Técnicas Criptográficas

José Manuel E. Valença *

8 de Junho de 2009

*Departamento de Informática, Universidade do Minho, Campus de Gualtar Braga

1.Fundamentos Matemáticos

Neste capítulo introdutório faz-se uma revisão dos fundamentos matemáticos essenciais à generalidade das técnicas criptográficas.

Vamos focar algumas noções de computabilidade, falaremos dos domínios relevantes ao processo computacional e algumas noções de Teoria das Probabilidades. Falaremos de algorimos e da sua relação com a formalização da segurança das técnicas criptográficas. Iniciamos porém, com alguma notação relevante ao estudo da dimensão dos objectos computacionais com que iremos lidar.

Neste curso $\mathbb{N}$ designa o domínio dos números naturais. $\mathbb{B} = \{0, 1\}$ representa o domínio finito dos bits e $\mathbb{B}^n$ o domínio finito das palavras de n bits. O único elemento de $\mathbb{B}^0$ designa-se por **string vazia** e é representado por ε. $\mathbb{B}^*$ e $\mathbb{B}^\infty$ representam domínios de bit-strings, finitas e infinitas, respectivamente.

Dentro dos reais vamos resignar por $\mathbb{I}$ o intervalo unitário real $[0, 1]$ e por $\mathbb{R}_+ = [0, +\infty]$ a "compactificação" dos reais positivos; isto é, os reais ≥ 0 adicionados de um ponto especial $+\infty$.

A cada natural x associamos um **comprimento** $|x|$ que é o menor número de bits necessários para a representação binária de x; isto é $|x| = \lfloor \log_2(x + 1) \rfloor$.

1.1 Notação Assimptótica

Frequentemente lidamos com funções reais de argumento inteiro positivo $f : \mathbb{N} \longrightarrow \mathbb{R}_+$ cujo comportamento não é conhecido com rigor mas que, ainda assim, pode-se ver contido dentro de determinados limites.

Muitas vezes o argumento n da função é o tamanho de um determinado objecto finito (tipicamente um número natural ou uma string finita) e basta conhecer a "ordem de grandeza" dos resultados $f(|x|)$ quando x percorre um qualquer domínio "computável".

Como estas funções podem ser ilimitadas, o estudo do comportamento destas funções exige que acrescentemos o símbolo ∞ a $\mathbb{N}$. Como símbolo ∞ deve verificar

$$\infty \leq \infty \qquad \text{e} \qquad n < \infty$$

O símbolo é incluído em $\mathbb{Q}$ e $\mathbb{R}$ com a imersão de $\mathbb{N}$ nestes domínios. $f(\infty) = \infty$ denota uma função ilimitada.

Usaremos o quantificador $(\exists_\infty\, x)$ para representar a asserção *existe um número infinito de valores* x; $(\exists_\infty)$ também se escreve como **i.o.** ("infinitely often").

Usaremos $(\forall_\infty\, x)$ como o quantificador que indica *para todos os valores de* x *com um número finito de excepções*; a sua denominação alternativa é **a.e.** ("almost everywhere").

1 Noção (Notação O)

- $f(n) = O(g(n))$ *sse existe* $C > 0$ *tal que* $\forall_\infty n \cdot f(n) \leq C\, g(n)$
- $f(n) = o(g(n))$ *sse* $\lim_{n\to\infty} f(n)/g(n) = 0$.
- $f(n) = \Theta(g(n))$ *sse* $f(n) = O(g(n)) \wedge g(n) = O(f(n))$
- $f(n) = O^{\log}(g(n))$ *sse* $\forall_\infty n \cdot f(n) \leq g(n) + O(1)$.
- $f = \Omega(g)$ *é equivalente a* $g = O(f)$, *e* $f = \Omega^{\log}(g)$ *equivalente a* $g = O^{\log}(f)$.

Notas

1. A notação pode ser usada para exprimir várias tipos de propriedades das funções $n \mapsto f(n)$.Por exemplo $f(n) = c + o(1)$ é apenas um modo de escrever $\lim_{n\to\infty} f(n) = c$; ou, $f(n) = O(1)$ é apenas outro modo de afirmar que f é uma constante positiva.

2. A comparação de funções usando $O(\cdot)$ é a mais frequentemente usada. Diz-nos que, em valor absoluto, $f(n)$ está limitada superiormente por uma função $C\, g(n)$ com a "forma" de g . Diz também que $|f(n)/g(n)|$ está limitado superiormente à constante C .

   A comparação $o(\cdot)$ é mais exigente; diz que este quociente tem de tender para zero. Por isso $f(n) = o(g(n))$ implica $f(n) = O(g(n))$ mas a implicação inversa não se verifica necessariamente.

3. Usando a definição de $\Omega(\cdot)$ a definição de $\Theta(.)$ equivale à afir- . mação de que $|f(n)/g(n)|$ está limitado superiormente por uma constante $C > 0$ e inferiormente por uma constante $c > 0$ para todo n suficientemente grande

   $$c\,|g(n)| \leq |f(n)| \leq C\,|g(n)| \qquad \text{ou} \qquad c \leq |f(n)/g(n)| \leq C$$

   A figura 1 representa $f(n) = \Theta(n\,g(n))$; ou seja,

   $$c\,n \leq |f(n)/g(n)| \leq C\,n$$

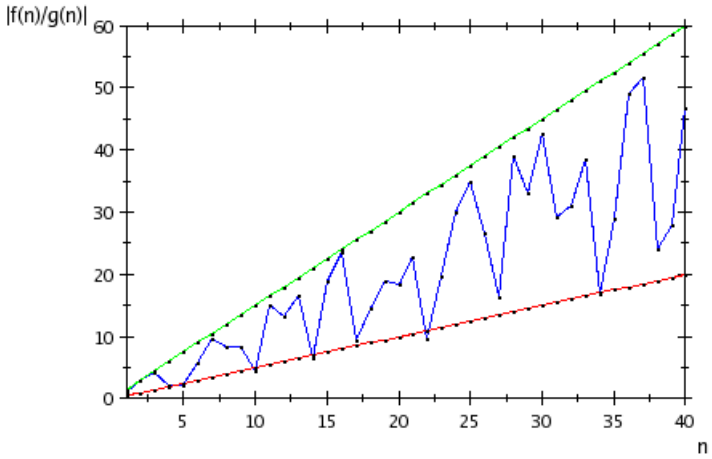
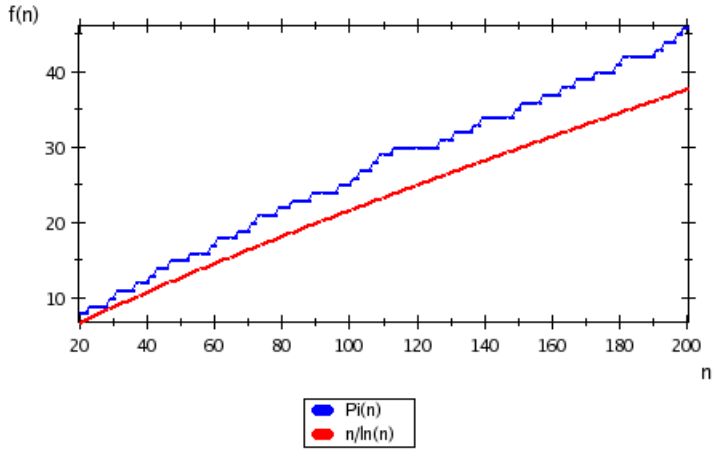


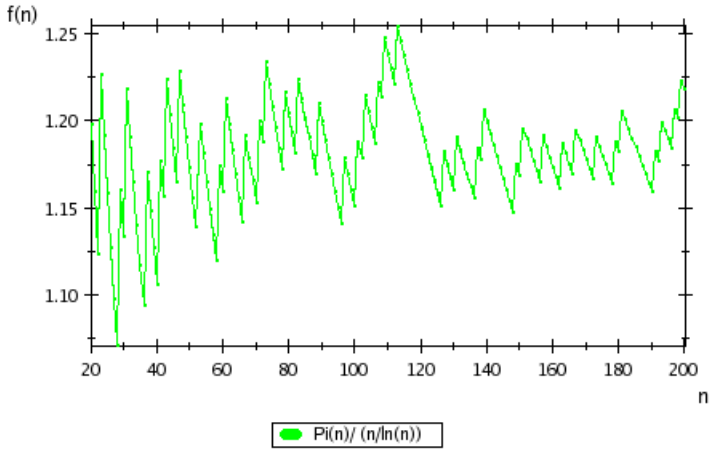
Figura 1: $f(n) = \Theta(n\,g(n))$.

EXEMPLO 1: O valor $\pi(n)$ define-se como *o número de números primos* $\leq n$. Não é possível construir uma função que calcule $\pi(n)$ de forma eficiente para todo argumento n mas é possível aproximar a função π por funções que podem ser calculadas de forma eficiente.

É normal aproximar $\pi(n)$ pela expressão $n/\ln n$. A primeira figura apresenta a variação destas duas funções

($\pi(\cdot)$ e a sua aproximação) na gama $n = 20..200$.





A segunda figura apresenta o quociente $\frac{\pi(n)}{n/\ln n}$. Note-se que o quociente está limitado acima e abaixo por duas constantes ~ 1. Por isso faz sentido afirmar que

$$\pi(n) \;=\; \Theta(n/\ln n)$$

No estudo dos algoritmos a notação assimptótica é usada, principalmente, para caracterizar a sua **complexidade**.

Uma medida de complexidade, frequentemente usada, é o número de *slots* temporais (por exemplo, ciclos de relógio) que uma máquina de referência usa para correr esse algoritmo. Pode-se também medir uma número de células de memória usadas nesse processo. No primeiro caso avalia-se a chamada **complexidade temporal** do algoritmo enquanto que no segundo caso avalia-se a **complexidade espacial** do mesmo.

Em qualquer dos casos temos uma função do tipo $\tau : \mathbb{N} \longrightarrow \mathbb{R}_+$ que mede essa complexidade. O argumento da função representa, quase sempre, uma medida da incerteza nos argumentos do algoritmo. Em Criptografia é normal usar-se o número de *bits* que são necessários para determinar o argumento do algoritmo. Deste modo, no estudo da complexidade, $\tau(n)$ conta o número médio de *slots* temporais ou o número médio de células de memória para um argumento do algoritmo especificado com n *bits*. [1]

Alguns casos particulares da ordem de uma função $\tau : \mathbb{N} \longrightarrow \mathbb{R}_+$

2 NOÇÃO (ORDEM DE COMPLEXIDADE)
Diz-se que τ *tem* ***ordem***

- ***polinomial*** *quando* $\tau(n) = O(p(n))$ *, para algum polinómio positivo* $p(n)$[2]*,*
- ***polinomial inversa*** *quando* $\tau(n)^{-1} = O(p(n))$ *, para um polinómio positivo* $p(n)$[3]*,*

[1] Em alternativa, a complexidade pode contar o número *máximo* ou o número *mínimo* de *slots* temporais ou unidades de memória.
[2] Como casos particulares temos as ordens **linear** ($\tau(n) = O(n)$), **quadrática** ($\tau(n) = O(n^2)$), **cúbica** ($\tau(n) = O(n^3)$), etc. . .
[3] Versões "inversas" das ordens anteriores: **inversa linear** ($\tau(n)^{-1} = O(n)$), **inversa quadrática** ($\tau(n)^{-1} = O(n^2)$), etc. . .

- ***exponencial*** *quando* $\tau(n)$ *não é polinomial,*
- ***desprezável*** *quando* $\tau(n)$ *não é polinomial inversa,*
- ***sub-exponencial*** *quando* $\tau(n) = O(2^{o(n)})$.

A ordem sub-exponencial é algo intermédio entre a ordem polinomial e exponencial e é frequentemente escrita numa notação específica.

3 NOÇÃO
Define-se $L[p, c](n) = O(2^{c\, n^p\, (\log_2 n)^{1-p}})$, *para* $p \in [0, 1]$ *e* $c > 0$.

É fácil verificar que:

1. A ordem $L[0, c](n) = O(n^c)$ é a ordem polinomial.
2. A ordem $L[1, c](n) = O(2^{c\, n})$ é a ordem completamente exponencial.

Assim a ordem $L[p, c](n)$ aproxima-se da ordem polinomial quanto mais pequeno for p e aproxima-se da ordem exponencial quanto maior for p (limitado, obviamente, a 1).

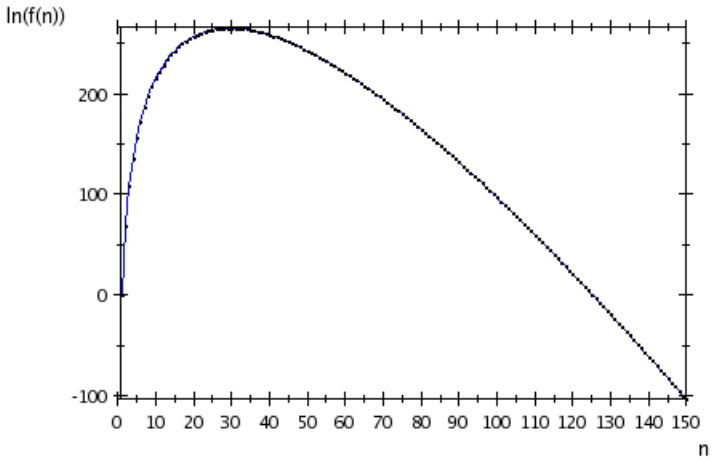
□

Funções $p(n)$ da forma n^2, ou $1 + n^4 + n^{256}$, são exemplos de funções **polinomiais positivas**. Funções da forma $p(n)/q(n)$, em que $p(n)$ e $q(n)$ são polinomiais positivas, dizem-se **racionais positivas**.

Um paradigma de função desprezável é $\delta(n) = 2^{-q(n)}$, com $q(n)$ um p.p. de grau > 0.

O mesmo se pode dizer da função $\delta(n) = q(n)/n!$.

Esta figura ilustra o comportamento de $n^{100}/n!$ em escala logarítmica; após um máximo perto de $n = 30$, o logaritmo da função tende rapidamente para $-\infty$ o que indica que $n^{100}/n!$ tende rapidamente para zero.



1.2 Funções, Computabilidade e Recursividade[4]

Criptografia lida essencialmente com objectos **computáveis** e, por isso, é muito importante estabelecer, desde o início, o nosso entendimento sobre esses conceitos.

Tradicionalmente associa-se computabilidade de uma função f à capacidade de uma máquina de Turing simular o seu comportamento. Como vamos lidar, essencialmente, com funções parciais $f, g : \mathbb{N} \longrightarrow \mathbb{N}$ convém introduzir alguma notação prévia.

1. $f(x) \simeq y$ é um **predicado** que indica que f é definido em x e o seu valor é y. $f(x) \not\simeq y$ é a sua negação: $f(x)$ não é definido ou, se for, é diferente de y.
2. $f(x)\uparrow$ e $f(x)\downarrow$ são **predicados** que indicam, respectivamente, que f não é definido em x e que f é definido em x.
3. $\mathrm{dom}(f) = \{\, x \mid f(x)\downarrow \,\}$ é o **domínio** de f e $\mathrm{rng}(f) = \{\, y \mid \exists x \cdot f(x) \simeq y \,\}$ é o **contra-domínio** de f.
4. Se $\forall x \cdot f(x)\downarrow$ (equivalentemente, quando $\mathrm{dom}(f) \equiv \mathbb{N}$) a função f diz-se **total**. Funções totais são também designadas por **sequências**.
5. $\digamma f$ é equivalente a $(\forall_\infty x)[f(x)\uparrow]$; ou seja, é finito o número de argumentos x onde $f(x)$ é definido.

[4] Este capítulo e o seguinde (dedicado a conjuntos) deriva essencialmente, e com pequenas adaptações, da obra *Classical Recursion Theory*, *P.G. Odifreddi, North-Holland, Volume 1 (1992) e Volume 2 (1999)*.

6. $f(x)\!\downarrow\!\downarrow \;=\; (\forall y < x)[f(y)\!\downarrow]$ é o predicado que indica que $f(y)$ é definido para todos os valores $y < x$.
7. Funções parciais admitem uma ordem parcial $\lesssim$ definida por

$$f \lesssim g \quad \text{sse} \quad (\forall x)\,[f(x)\!\downarrow \;\Rightarrow\; g(x) \simeq f(x)]$$

   A função parcial $\emptyset$ (também representada por $\perp$) é o mínimo desta ordem parcial; é a função que é indefinida para todo o possível argumento: $(\forall x \in \mathbb{N})\,[\emptyset(x)\!\uparrow]$.
8. $f \simeq g$ compara duas funções; diz-nos que num argumento arbitrário x ambos os lados ou são ambos indefinidos ou são definidos e têm o mesmo valor. Isto é

$$f \;\simeq\; g \qquad f \lesssim g \quad \wedge \quad g \lesssim f$$

9. Uma função parcial é uma **função característica** se o seu único valor definido é 0; isto é,

$$f(x)\!\downarrow \;\Rightarrow\; f(x) \simeq 0$$

   No contexto do estudo das funções computáveis identificamos **conjuntos** de $\mathbb{N}$ com funções características De facto uma função característica f determina um conjunto por compreensão $\{\, x \mid f(x) \simeq 0 \,\}$. Dessa forma a classificação que iremos fazer dos conjuntos reflecte a classificação nas funções. Por exemplo diremos que um conjunto pertence a uma classe $\mathcal{C}$ se e só se a função característica que o determina pertence a essa classe.
10. Uma **relação** é uma função total r cujos únicos resultados possíveis são 0 e 1. Isto é,

$$(\forall x)\,[r(x) \simeq 0 \;\vee\; r(x) \simeq 1]$$

Cada relação identifica-se também com o conjunto $\{\, x \mid r(x) \simeq 1 \,\}$. A função constante 0 determina o conjunto vazio e, por isso, essa relação é denotada por $\emptyset$.

Inversamente cada conjunto A pode sempre ser visto como uma relação r tal que $r(x) \simeq 1 \Leftrightarrow x \in A$.

11. Um dos problemas essenciais na construção de funções é a representação dos pares de inteiros. Vamos assumir a existência de um operador binário $.\|.: \mathbb{N} \times \mathbb{N} \longrightarrow \mathbb{N}$ que codifica pares de inteiros em inteiros e é sobrejectivo. Vamos assumir também que existem duas **funções projecção** sobrejectivas $\pi_1, \pi_2 : \mathbb{N} \longrightarrow \mathbb{N}$ que verificam $\forall x, y, z \in \mathbb{N}$

$$\pi_1(x\|y) \simeq x \quad , \quad \pi_2(x\|y) \simeq y \quad , \quad \pi_1(z)\|\pi_2(z) \simeq z \tag{1}$$

12. O **grafo** de uma função f é a relação r que verifica

$$r(x\|y) \simeq 1 \quad \Leftrightarrow \quad f(x) \simeq y \tag{2}$$

Dado que cada relação determina um conjunto por compreensão, cada função f determina também um conjunto através do seu grafo: $\{\, z \mid f(\pi_1(z)) \simeq \pi_2(z) \,\}$. Portanto, neste curso, a designação "grafo" refere-se (dependendo do contexto) quer à relação em (126) como o conjunto que lhe está associado.

13. A construção do grafo de uma função f é uma construção que permite representar qualquer função por uma relação. Inversamente é possível construir uma função recursiva f, a partir de uma relação r, que verifique

$$f(x)\downarrow \quad \Leftrightarrow \quad (\exists y)\,[\, r(x\|y) \simeq 1 \,] \tag{3}$$

e que tenha r como grafo; isto é, verifique (126).

Esta construção é designada por **valor mínimo** ou **recursividade**-μ e a função escreve-se $f(x) \simeq \mu y \cdot r(x\|y)$. Intuitivamente $\mu y \cdot r(x\|y)$ procura o menor valor de y que verifica $r(x\|y) \simeq 1$. Portanto

$$\mu y \cdot r(x\|y) \simeq z \quad \Leftrightarrow \quad [r(x\|z) \simeq 1] \wedge (\forall y < z)\, [r(x\|y) \simeq 0] \tag{4}$$

□

Diferentes classes de máquinas de Turing cracterizam diferentes noções de computabilidade:

Máquinas de Turing Determinístas (DTM´s)

Caracterizadas por uma "memória longa" (sequência duplamente infinita de células contendo bits ou marcas que seleccionam "conteúdo"), uma "posição", um conjunto de "estados" e um conjunto de instruções para, em função do estado e do conteúdo da célula selecionada pela posição, determina um novo estado e altera memória e posição.

O triplo *estado*×*posição*×*conteúdo* designa-se por **configuração**. A função transição mapeia configurações em configurações. Uma configuração é inicial (ou final) se contém o estado inicial (ou final).

4 Noção

Uma máquina de Turing M **para** *em x, e escreve-se $M(x)\downarrow$, se partindo de uma configuração inicial com conteúdo x alcança uma configuração final.*

5 NOÇÃO

Uma máquina de Turing determinística M **simula** *a função parcial* $f : \mathbb{N} \longrightarrow \mathbb{N}$ *com compexidade* $T(n), S(n)$ *– e escreve-se* $f \lesssim_{T,S} M$ *– se, partindo de uma configuação inicial com conteúdo* x *onde* f *seja definido, se alcança, em não mais de* $T(|x|)$ *passos e não ocupando mais do que* $S(|x|)$ *células, uma configuração final com conteúdo* $f(x)$.

A máquina M **computa** f *– e escreve-se* $M \simeq_{T,S} f$ *– se simula* f *e, adicionalmente,* $M(x)\downarrow \Rightarrow f(x)\downarrow$.

Note-se que a definição de simulação nada diz sobre as situações onde $f(x)$ não é definido; aqui a máquina pode ou não alcançar um estado final. A definição de computação já exige que a máquina pare exactamente para os mesmos argumentos onde f está definido.

Quando os recursos T e S estão implícitos, não são explicitamente colocados na notação e escreve-se simplesmente $f \lesssim M$ ou $f \simeq M$.

Máquinas de Turing Não-Determinísticas (NDTM)

Com a mesma constituição que as DTM´s mas com a opção de, em cada configuração – par (*estado*,*posição*) – haver transições não para uma mas para um número finito de configurações.

A noção de computabilidade é a mesma: partindo de uma configuração com conteúdo x, a computação termina quado uma das possibilidades de transição atingir o estado final. A máquina é consistente se, nessas circusntâncias, o conteúdo for sempre o mesmo.

Máquina de Turing Probabilística (PTM)

O resultado de cada transição é uma variável aleatória caracterizada por uma determinada distribuição probabilística. Em cada passo a máquina lê um bit fornecido por uma bit-strings infinita e aleatória; a nova

configuração é função desse bit e da configuração actual.

A noção de computabilidade é agora diferente. A máquina de Turing simula funções "a menos de um erro".

6 Noção

Uma máquina de Turing probabilística M **simula** *a função $f : \mathbb{N} \to \mathbb{N}$ com compexidade $T(n), S(n), \varepsilon(n)$ se, partindo de uma configuração inicial e com conteúdo x arbitrário no domínio de f, se alcança, em não mais de $T(|x|)$ passos, não ocupando mais do que $S(|x|)$ células e com uma probabilidade de falha que não excede $\varepsilon(|x|)$, uma configuração final de conteúdo $f(x)$. A máquina M* **computa** *f se simula f e pára em x só se f converge em x.*

Em função do tipo de máquina seleccionada e dos recursos admissíveis (tempo de processamento $T(n)$, espaço ocupado $S(n)$ e margem de erro $\varepsilon(n)$) podem-se definir várias classes de computabilidade. Por exemplo

TM classe das funções computáveis por máquinas de Turing (determinísticas ou não-determinísticas) sem quaisquer limitações de recursos.

PPTM funções compuáveis por máquinas de Turing probabilísticas com margem de erro desprezável.

PTIME funções computáveis por máquinas de Turing determinísticas em tempo $T(n)$ polinomial.

PPTIME funções computáveis por máquinas de Turing probabilísticas em tempo $T(n)$ polinomial e com margem de erro desprezável.

NPTIME funções computáveis por máquinas de Turing não-determinísticas em tempo polinomial.

. . . classes análogas podem-se definir com restricções no espaço em vez do tempo; várias combinações destas classes são possíveis.

Um dos resultados mais importantes da computabilidade é a enumeração das máquinas de Turing e a existência de uma máquina de Turing universal.

Em primeiro lugar pode-se provar que qualquer uma das classes de computabiliadade atrás referidas são enumeráveis. Por isso é possível indexar as diferentes máquinas dentro da classe criando uma sequência $\mathcal{C} = \{M_n\}$ dos seus elementos.

7 Teorema
Seja $\mathcal{C} = \{M_n\}$ uma enumeração de máquinas de Turing e sejam $\{f_n\}$ funções parciais que estas máquinas computam: $M_n \simeq f_n$. Seja f a função que verifica $f(n\|x) = f_n(x)$. Então existe uma máquina de Turing, dita **universal**, *$\mathcal{U}$ tal que $\mathcal{U} \simeq f$.*

Adicionalmente, se cada M_n computar f_n em $T(n)$ passos, a máquina $\mathcal{U}$ computa f em $O(T(n)\ \log T(n))$ passos.

□

Independentemente da classe de funções que se considere computáveis, e que depende essencialmente da constituição das máquinas que as simulam, é necessário classificar as funções pela forma como elas próprias são construídas. Surge, assim, toda a importância da noção de **recursividade**.

8 Noção (Funções Primitivas e Primitivas Recursivas)
São **funções primitivas** *a constante* 0, *a função "sucessor"* $x \mapsto x+1$, *e as projecções* π_1, π_2.

Uma função f *é* **definida por recursividade primitiva** *a partir das funções* g *e* h *quando, para todo* $x, y \in \mathbb{N}$,

$$f(x\|0) \simeq g(x) \quad , \quad f(x\|y+1) \simeq h(x\|y\|f(x\|y)) \tag{5}$$

Uma função f *diz-se* **primitiva recursiva** *se é uma função primitiva, se é a composição de duas outras funções primitivas recursivas ou se é definida por recursividade primitiva a partir de funções* g, h *primitivas recursivas.*

Funções primitivas recursivas que têm só dois resultados possíveis, 0 *e* 1, *designam-se por* **relações primitivas recursivas**.

Como primeiro resultado das funções primitivas recursivas, tem-se

9 Facto
A função f *é primitiva recursiva primitiva se e só se a sua* **função história** $\hat{f}$, *que verifica para todo* $x \in \varpi$,

$$\hat{f}(x\|0) \simeq f(x\|0) \quad , \quad \hat{f}(x\|y+1) \simeq f(x\|y+1) \,\|\, \hat{f}(x\|y) \tag{6}$$

for primitiva recursiva.

É necessário uma construção adicional para ser possível definir completamente a noção de recursividade. Para isso usamos o chamado **princípio do número mínimo** que, essencialmente diz,

> *Se uma propriedade P ocorre em algum x então existe um único $y \leq x$ tal que P ocorre em y e não ocorre em nenhum $z < y$.*

$$P(x) \;\Rightarrow\; (\exists_1\, y \leq x)[P(y) \wedge (\forall z < y)\, \neg P(z)] \tag{7}$$

Denota-se por μP (ou, mais explicitamente, $\mu x \cdot P(x)$) o valor y que verifica (7). Caso não exista nenhum x onde a propriedade $P(x)$ ocorra diz-se que μP é indefinido e representa-se esse facto por $\mu P{\uparrow}$. Note-se que a capacidade para determinar esse valor μP depende crucialmente do facto de a propriedade ser sempre definida pelo menos até ao valor y.

Uma algoritmo possível para determinar μP consistiria em percorrer todos os números naturais (desde o 0) até encontrar o primeiro y onde P ocorra. Porém de P for indefinido num valor $z < y$, como é computacionalmente indicidível verificar se $f(z)$ é definido, nunca seria possível ao algoritmo alcançar o valor y. Portanto o algoritmo só pode funcionar se P for definido para todos os $z < y$.

Como caso particular de propriedades considere-se, para qualquer função parcial f e qualquer natural l,

$$\mathcal{Z}[f](x) \;=\; f(x){\downarrow\downarrow} \wedge f(x) \simeq 0 \tag{8}$$

$$\mathcal{Z}[l, f](x) \;=\; x \leq l \wedge \mathcal{Z}[f](x) \tag{9}$$

Notas

1. $\mathcal{Z}$ e $\mathcal{Z}[l,\cdot]$ mapeiam uma função parcial arbitrária f em predicados. Como predicados são casos especiais de funções parciais, $\mathcal{Z}$ é uma "função de ordem superior" dado transformar funções em funções.
   No contexto da teoria da recursividade, estes objectos de ordem superior designam-se por **funcionais**. Enquanto as funções actuam sobre naturais e dão resultados naturais, as funcionais actuam sobre funções (ou predicados, ou conjuntos, etc.) e dão, como resultado, objectos com esta mesma natureza.
2. $\mathcal{Z}[l,f]$ é uma "variante limitada" de $\mathcal{Z}[f]$. A diferença fundamental entre elas está na decidibilidade de predicados do tipo $(\exists x)[\mathcal{Z}[f](x)]$. Normalmente este tipo de fórmulas não são decidíveis mesmo no caso simples em que f é uma função recursiva primitiva.
   Sendo f definido por recursividade primitiva, será uma função total; o predicado $(\exists x)[\mathcal{Z}[f](x)]$ é, simplesmente, $(\exists x)[f(x)=0]$. Se for válido é computável verificar esse facto; no entanto, se não for válido, já não é computável verificá-lo.
   Já $(\exists x)\,[\,\mathcal{Z}[l,f](x)\,]$ coincide com $(\exists x \le l)[\,f(x)=0\,]$; por isso é sempre computável, quer seja válido ou não.

10 NOÇÃO (RECURSIVIDADE)

A função parcial h define-se por **recursividade-μ** *não-limitada, a partir da função parcial f se*

$$h(y) \simeq \mu\mathcal{Z}[f_y] \qquad \textit{em que} \quad f_y(x) = f(y\|x) \tag{10}$$

h define-se por **recursividade limitada-μ** *se, para algum l, em* (117) *$\mathcal{Z}[f]$ é substituído por $\mathcal{Z}[l,f]$.*

A classe das **funções parciais recursivas**, *representada por $\mathcal{RP}$, é a menor classe que contém as funções primitivas e é fechada por composição, recursividade primitiva e recursividade-μ não-limitada.*

Dito de outra forma, uma função f é **parcial recursiva** se e só se cai dentro de uma das quatro possibilidades enunciadas: (1) é uma função primitiva, (2) ou é a composição de duas funções recursivas, (3) ou é obtida por recursividade primitiva a partir de duas funções recursivas, (4) ou é obtida por recursividade-μ não-limitada a partir de uma função recursiva.

11 NOÇÃO

1. *Funções parciais recursivas f cujo domínio é finito (isto é, onde se verifica $\digamma f$) designam-e por* ***funções finitas***
2. *Funções parciais recursivas cujo único valor definido é* 0 *designam-se por* ***funções características****.*
3. *As funções $f \in \mathcal{RP}$ que são totais (isto é, verificam $(\forall x)[f(x)\downarrow]$) designam-se simplesmente por* ***recursivas****. A respectiva classe é representada por $\mathcal{R}$.*
4. *As funções recursivas que têm apenas resultados* 0 *ou* 1 *designam-se por* ***relações recursivas****. A sua classe designa-se por $\mathcal{RS}$* [5]*.*

Relativa a uma relação recursiva arbitrária ρ usa-se a seguinte notação:

(i) os predicados $[\rho(x) \simeq 1]$ e $[\rho(x) \simeq 0]$ escrevem-se, respectivamente, como $\rho(x)$ e como $\neg\rho(x)$.

(ii) $\mu\rho$ é uma abreviatura de $\mu[\rho(x) \simeq 1]$.

[5] De "recursive set".

(iii) ρ^+ e ρ^- denotam as funções parciais cujo único valor definido é 0 e que verificam

$$\rho^+(x)\downarrow \Leftrightarrow \rho(x) \quad , \quad \rho^-(x)\downarrow \Leftrightarrow \neg\rho(x) \tag{11}$$

É simples provar que ambas são funções parciais recursivas.

Uma propriedade essencial destas sub-classe de funções:

12 Facto
A sub-classe das funções características é fechada por composição, recursividade primitiva e recursividade-μ não limitada. A sub-classe das relações recursivas é fechada por composição e recursividade primitiva.

□

A recursividade traduz uma definição de funções pela forma como são construídas. As máquinas de Turing representam funções pela forma como são computadas. Um dos resultados fundametais da Matemática liga estes dois conceitos.

13 Teorema (Computabilida Básica)
As classe $\mathcal{R}$ e TM *coincidem.*

Essencialmente o teorema diz-nos que as funções parciais recursivas são precisamente as mesmas funções que são

computáveis por máquina de Turing não-probabilísticas sem restricções de recursos. Para vários outros modelos da computação (autómatos, λ-Calculus, etc.) é possível definir equivalências análogas.

No entanto estes resultados dizem pouco sobre a verdadeira natureza das funções computáveis; isto porque as máquinas de Turing sem restricções de recursos são modelos pouco realistas da computação; normalmente "computabilidade" subentende "computabilidade efectiva", no sentido em que só são interressantes os modelos computacionais de dispositivos realistas que possam computar resultados com recursos realistas. Por isso saber que uma função é recursiva pouco diz sobre a sua aptidão para ser simulada por um tal dispositivo. A afirmação *se a função é recursiva é efectivamente computável* é, desta forma, falsa; existem muitas funções recursivas que não são efectivamente computáveis.

O inverso, dizer que *toda a função efectivamente computável é recursiva* ou, equivalentemente, *toda a função que não é recursiva não pode ser efectivamente computável* é a famosa **tese de Turing**.

Esta é uma questão bastante mais complexa, que não pode ser colocada ao nível exclusivo da ciência matemática porque está fortemente ligada à espistemologia (e mesmo à psicologia): o que é saber calcular, que percepção se tem dos possíveis resultados do cálculo, etc. Por isso está, obviamente, fora do contexto deste curso.

Voltamos, assim, à nossa questão inicial: quais são as funções que devemos considerar "computáveis"? Neste curso vamos seguir a abordagem normalmente assumida em Criptografia e, a menos que seja explicitamente estipulado em contrário, vamos convencionar

Funções Computáveis
São **efectivamente computáveis** as funções **PPTM**; isto é, as funções parciais computadas, em tempo polinomial e com margem de erro desprezável, por uma máquina de Turing probabilística.

□

Enumeração e Normalização das Funções Parciais Recursivas

Dado que a classe $\mathcal{RP}$ das funções parciais recursivas é definida indutivamente a partir de um número finito de constantes e um número finito de regras, é sempre possível codificar uma função num número inteiro seguindo, simplesmente, a forma indutiva da sua construção. Assim, a construção das funções p.r. reflecte-se numa codificação $\mathcal{I}\colon \mathcal{R} \longrightarrow \mathbb{N}$; a aplicação $\mathcal{I}$ codifica a função f num inteiro p a que chamaremos **índice** (ou **programa**) de f.

O essencial desta codificação é a sua capacidade em recuperar qualquer função, a partir do seu programa, usando um "interpretador universal". Em primeiro lugar para as funções recursivas

14 Proposição (Avaliação das Funções Recursivas)
Existe uma avaliação recursiva $\varphi\colon \mathbb{N} \longrightarrow \mathbb{N}$ *tal que, para todo* $f \in \mathcal{R}$ *e todo* $p \in \mathbb{N}$

$$\mathcal{I}(f) = p \qquad \Longleftrightarrow \qquad (\forall x)\,[\,\varphi(p\|x) \simeq f(x)\,] \tag{12}$$

A função recursiva φ é um "interpretador universal"[6] das funções totais recursivas: recebe o programa p e o argumento x e produz o mesmo resultado $f(x)$.

O conhecimento da função f permite obter o programa p. Porém não é possível decidir recursivamente se um inteiro arbitrário p é um programa para alguma função total; isto é, a fórmula $(\forall p \in \mathbb{N})(\exists f \in \mathcal{R})[\mathcal{I}(f) \simeq p]$ não é computavelmente decidível.

Esta assimetria é levantada quando se considera funções parciais. Agora qualquer inteiro poderá ser programa para uma função parcial mesma que ela seja a função trivial $\emptyset$. Adicionalmente a função avaliação φ (que é recursiva) pode ser normalizada para uma construção que usa duas funções primitivas recursivas.

15 Teorema (Normalização das Funções Parciais Recursivas)
Existe uma função primitiva recursiva $\mathcal{U}$, *e uma relação primitiva recursiva* $\mathcal{S}\colon \mathbb{N} \to \mathbb{N}$, *que determinam uma função parcial recursiva* φ *por*

$$\varphi(x) \simeq \mathcal{U}(\mu \mathcal{S}_x) \qquad \text{sendo} \quad \mathcal{S}_x(y) = \mathcal{S}(x \| y) \tag{13}$$

de tal forma que a sequência $\{\varphi_n(x) = \varphi(n \| x)\}$ *enumera* $\mathcal{RP}$.

Dizer que $\{\varphi_n\}$ enumera $\mathcal{RP}$ significa que toda a função parcial recursiva f está associada a um programa n tal que $f \simeq \varphi_n$. Inversamente, cada programa n determina a sua função p.r. φ_n.

[6] Os programadores de linguagens funcionais reconhecem em φ a função `eval` do Lisp ou o operador `$` do Haskell.

16 NOÇÃO

Dois programas p, q *são* **extensionalmente equivalentes**, *e escreve-se* $p \cong q$, *quando* $\varphi_p \simeq \varphi_q$.

Comparando com a proposição 14 nota-se uma restricção importante: a "avaliação" φ tem, aqui, uma forma particular determinada por duas funções que são totais e primitivas recursivas.

Como primeira consequência, a propriedade $\mathcal{Z}[\mathcal{S}_x](y)$ simplifica-se no simples teste $\mathcal{S}(x\|y) = 0$ e $\varphi(x)$ será definido se e só se $(\exists y)\,[\mathcal{S}(x\|y) = 0]$. A forma genérica de φ diz-nos que a recursividade-μ só necessita de ser usada uma vez: todas as funções recursivas podem ser calculadas usando apenas composição e recursividade primitiva e uma única utilização (eventual) da recursividade-μ não-limitada.

O seguinte resultado é uma consequência simples da normalização.

17 FACTO

Se f, g *forem duas funções totais recursivas (a.k.a., "sequências recursivas"),e* f *é injectiva, então existe uma função parcial recursiva* ϕ *tal que* $\mathrm{dom}(\phi) = \mathrm{rng}(f)$ *e* $\phi(f(n)) \simeq g(n)$, *para todo* n.

Prova Define-se $\phi(x) = g(\mu n \cdot f(n) \simeq x)$. É trivial verificar que ϕ satisfaz as propriedades requeridas.

A única utilização da recursividade-μ não-limitada, definida no teorema 15, sugere uma variante limitada da função φ substituindo a recursividade ilimitada por recursividade limitada.

18 NOÇÃO

Sejam $\mathcal{U}, \mathcal{S}$ as funções primitivas recursivas definidas no teorema 15. Para cada natural l, a **enumeração das aproximações** *é a sequência de funções $\left\{\varphi_n^l(x) = \varphi^l(n\|x\right\}$ com φ^l definida por*

$$\varphi^l(x) \simeq \mathcal{U}(\mu\mathcal{Z}[l, \mathcal{S}_x]) \qquad sendo \quad \mathcal{S}_x(y) = \mathcal{S}(x\|y) \tag{14}$$

As propriedades essenciais da aproximação φ^l são

$$\varphi^l \lesssim \varphi \quad , \quad \varphi^l(x) \simeq \varphi(x) \Leftrightarrow (\exists y < l)\, \mathcal{S}(x\|y) \tag{15}$$

A consequência mais importante é que $\varphi^l(x)\downarrow \Leftrightarrow (\exists y < l)\, \mathcal{S}(x\|y)$ é uma relação recursiva primitiva.

□

A noção de indexação das funções parciais recursivas, funções recursivas ou relações recursivas, pode ser extendida da seguinte forma.

19 NOÇÃO (ÍNDICES)

Um **sistema aceitável de índices** *em $\mathcal{RP}$ (ou $\mathcal{R}$) é uma enumeração $\{\psi_e\}$ de elementos de $\mathcal{RP}$ (ou elementos de $\mathcal{R}$) que verifica as seguintes propriedades:*

- ***enumeração**: existe um programa particular a (de "avaliação") tal que, para todos e, x, $\psi_a(e\|x) \simeq \psi_e(x)$.*
- ***parametrização**: existe uma função recursiva η tal que, para todo x, y, $\psi_{\eta(x\|e)}(y) \simeq \psi_e(x\|y)$.*

Notas Num sistema aceitável de índices $\{\psi_e\}$, a propriedade de "enumeração" diz-se que a avaliação é um programa a que recebe, como argumento, o emparelhamento $(e\|x)$ do par *programa+argumento* e reproduz o mesmo resultado que o progrma e origina no argumento x.

$$\psi(a\|(e\|x)) \simeq \psi(e\|x)$$

A propriedade da "parametrização" diz-nos que é possível decompor um argumento da forma $(x\|y)$ da função ψ_e, de tal forma que, com esse valor x e o programa original e, se constrói um novo progrma $i = \eta(x\|e)$ que, aplicado ao argumento que falta y, dá o resultado original.

$$\psi(\eta(x\|e)\|y) \simeq \psi(e\|(x\|y))$$

Nomeadamente, combinando estas duas observações, vê-se que $\psi_e(x) \simeq \psi(a\|(e\|x)) \simeq \psi_{\eta(e\|a)}(x)$, para todo x, e. Portanto, para todo e, $\eta(e\|a) \cong e$.

As propriedades essenciais dos sistemas aceitáveis de índices resume-se em:

20 TEOREMA

1. *$\{\psi_e\}$ é um sistema aceitável de índices em $\mathcal{RP}$ se e só se existem funções recursivas f e g tais que, para todo e, $\psi_e \simeq \varphi_{f(e)}$ e $\varphi_e \simeq \psi_{g(e)}$, em que $\{\varphi_e\}$ é a enumeração standard das funções parciais recursivas.*

2. *Todo o sistema aceitável de índices satisfaz a propriedade do ponto fixo: dada uma qualquer função recursiva f existe um índice e tal que $\psi_e \simeq \psi_{f(e)}$.*

Recordemos que a designação **funcional** está reservada para as aplicações que transformam funções (ou relações, ou conjuntos, etc.) em objectos com a mesma natureza. A indexação vai permitir definir funcionais em R, como **funções extensionais** nos índices. Formalmente

21 NOÇÃO

*Seja $\mathcal{C} = \{\psi_e\}$ um sistema aceitável de índices. Uma função f é $\mathcal{C}$-**extensional** sse*

$$\psi_a \simeq \psi_b \implies \psi_{f(a)} \simeq \psi_{f(b)}$$

*Uma funcional F é $\mathcal{C}$-**efectiva** em $\mathcal{RP}$, quando existe uma função recursiva f que é $\mathcal{C}$-extensional e verifica*

$$F[\psi_e] \simeq \psi_{f(e)}$$

*A funcional F é $\mathcal{C}$-**efectiva** em $\mathcal{R}$, quando é a restricção a $\mathcal{R}$ de uma funcional efectiva em $\mathcal{RP}$.*

Sendo F $\mathcal{C}$-efectiva e g uma qualquer função parcial recursiva, o predicado $g \in F$ determina se a função g está no seu contradomínio; isto é,

$$g \in F \quad \Leftrightarrow \quad (\exists e)\ \{\, g \simeq F[\psi_e] \,\} \quad \Leftrightarrow \quad (\exists e)\ \left\{\, g \simeq \psi_{f(e)} \,\right\}$$

Uma sequência de funcionais efectivas $\{F_n\}_{n\in\varpi}$ *é* **uniforme** *se existe uma função recursiva extensional* f *tal que*

$$F_n[\psi_e] \simeq \psi_{f(n\|e)}$$

Quando o sistema de índices é a enumeração standard de $\mathcal{RP}$, designaremos as funções simplesmente por **extensionais** e as funcionais por **efectivas**. A menos que o sistema de índices não seja explicitamente referido, estas serão as funcionais que iremos usar.

Dado que a enumeração standard de $\mathcal{RP}$ contém um função universal φ, verifica-se $\varphi_{f(e)}(x) \simeq \varphi(f(e)\|x)$. Assim, aí F será uma funcional efectiva quando existe uma função recursiva e extensional f tal que, para todo x, e,

$$F[\varphi_e](x) \simeq \varphi(f(e)\|x) \tag{16}$$

Se tivermos uma sequência uniforme de funcionais efectivas $\{F_n\}_{n\in\varpi}$ será, para todo x, e, n,

$$F_n[\varphi_e](x) \simeq \varphi(f(n\|e)\|x) \tag{17}$$

Funcionais efectivas permitem resolver o problema seguinte: *como caracterizar as classes de funções parciais que possam ser construídas de forma computacionalmente relevante*.

Recordemos que uma função parcial u é finita se o seu domínio é finito. Obviamente, toda a função finita é parcial recursiva. Interessam, particularmente, as funções finitas u geradas por funcionais efectivas.

22 Noção (Classes efectivamente enumeráveis de funções)
Cada funcional efectiva F determina uma classe de funções, que designaremos por $F^{\tilde{}}$, formado por todas as funções parciais recursivas que são mais definidas que uma função finita $u \in F$. Isto é

$$f \in F^{\tilde{}} \quad \Leftrightarrow \quad (\exists e)\,[\, f \simeq \varphi_e \,] \wedge (\exists u \in F)\,[\, F u \wedge u \lesssim f \,] \tag{18}$$

Uma classe de funções $\mathcal{A}$ diz-se **efectivamente enumerável (e.e.)**[7] *se é da forma $F^{\tilde{}}$ para alguma funcional efectiva F.*

Uma sequência de classes, $\{\mathcal{A}_n\}$, é **uniformemente recursiva** *(ou só* **uniforme***) quando cada $\mathcal{A}_n$ é da forma $F_n{}^{\tilde{}}$ para uma sequência uniforme $\{F_n\}$ de funcionais efectivas.*

□

O sistema standard de índices permite definir, também, uma noção de aleatoriedade. A intuição está em procurar, dado um y, o menor programa e que, sob um "input" fixo (neste caso, o inteiro 0) calcula y.

23 Noção
O **índice de Kolmogorov** *de y é o inteiro $\kappa(y)$ tal que*

$$\varphi_{\kappa(y)}(0) \simeq y \quad , \quad \varphi_e(0) \not\simeq y \quad \textit{par todo} \quad e < \kappa(y) \tag{19}$$

[7] Também designada por **completamente recursivamente enumerável** ou Σ^0_1-classe.

Existe pelo menos uma função que, no argumento 0, produz y; nomeadamente a função $x \mapsto x + y$; assim $\kappa(y)$ é sempre definido. Por outro lado, seria tentador definir $\kappa(y)$ por recursividade-μ a partir da função recursiva parcial $g_y(e) \simeq \varphi(e\|0) - y$. No entanto a recursividade μ, apesar de possível, não produz o resultado pretendido já que $g_y(e)$ pode não ser definida para valores $e < \kappa(y)$. Por isso

24 Facto
A função $y \mapsto \kappa(y)$ é total mas não é recursiva.

25 Noção
Um inteiro y diz-se **compressível** *quando $y > \kappa(y)$. Os inteiros não compressíveis dizem-se κ-*__aleatórios__.

Se virmos y como o código que representa um objecto num determinado domínio (por exemplo, strings de bits), a ideia de compressibilidade tem a haver com a possibilidade de existir um programa menor do que y que, aplicado a um *input* standard 0, reproduza o mesmo objecto. A noção de aleatoriedade que está associada à noção de compressibilidade, define como aleatórios precisamente os objectos que não são compressíveis.

Oráculos

Suponhamos que acrescentamos às constantes da definição usual de recursividade (noção 10, página 18) uma relação Φ (função total com valores em $\{0, 1\}$). A classe de funções resultante é representada por $\mathcal{RP}^{\Phi}$ e designam-se por funções recursivas parciais com **oráculo** Φ

26 Noção (Recursividade com Oráculos)
$\mathcal{RP}^{\Phi}$ é a menor classe de funções que contém Φ, as funções primitivas e é fechada por composição, recursividade primitiva e recursividade-μ ilimitada.

A noção de oráculo deriva, originalmente, das máquinas de Turing. Numa máquina de Turing com oráculo Φ, cada passo pode, ou não, recorrer ao oráculo Φ fornecendo-lhe o conteúdo da configuração actual para ajudar a determinar a próxima configuração.

Algo semelhante se passa numa máquina probabilística onde cada passo recorre a um bit de informação externo para ajudar a determinar a próxima configuração. Numa máquina com oráculo o bit não é externo mas resulta da aplicação de Φ ao conteúdo actual e, principalmente, este recurso não é necessariamente usado em todos os passos: é um recurso cujo uso é controlado.

Esta diferença traduz-se na definição de simulação e computação no caso geral de uma M.T. probabilística.

27 Noção
Uma máquina de Turing com oráculo M^{Φ} **simula** *a função f com compexidade $T(n), S(n), \varepsilon(n), O(n)$ se, partindo de uma configuração inicial e com conteúdo x arbitrário no domínio de f, se alcança, em não mais de $T(|x|)$ passos, não ocupando mais do que $S(|x|)$ células, com uma probabilidade de falha que não excede $\varepsilon(|x|)$ e consultando o oráculo não mais que $O(|x|)$ vezes, uma configuração final de conteúdo $f(x)$. A máquina M* **computa** *f se simula f e pára em x só se f converge em x.*

O uso de oráculos tem uma importância muito grande nas classes de complexidade das funções computáveis. Por exemplo, o seguinte resultado ilustra bem essa importância.

28 Teorema
Existem oráculos recursivos Φ *e* Ψ *tais que* $\mathrm{P}^{\Phi} = \mathrm{NP}^{\Phi}$ *e* $\mathrm{P}^{\Psi} \neq \mathrm{NP}^{\Psi}$.

Como sabemos, a asserção $\mathrm{P} \neq \mathbf{NP}$ é uma das grandes questões não decididas da Matemática. Este resultado dá respostas parciais recorrendo a oráculos recursivos.

1.3 Conjuntos, classes e sua computabilidade

Como foi referido anteriormente vamos identificar os conjuntos de naturais, para os quais queremos asociar propriedades de computabilidade, com domínios de funções características. Note-se que, no contexto deste estudo, funções características são sempre recursivas. As propriedades desses conjuntos são, portanto, as propriedades desta classe de funções.

29 PROPOSIÇÃO
A sub-classe de $\mathcal{RP}$ formada pelas funções características, é um reticulado distributivo com um limite inferior $\emptyset$ e limite superior $\mathbf{0}$. Os elementos f que têm complemento $\overline{f}$ são aqueles para os quais existe um relação recursiva ρ tal que $f \simeq \rho^+$ e $\overline{f} \simeq \rho^-$.

Notas A função trivial $\emptyset$ é um caso particular de função característica. A constante $\mathbf{0}$ é uma outra instância da mesma classe de funções. O domínio da primeira é o conjunto vazio $\emptyset$ enquanto que o domínio da segunda é todo o conjunto ϖ.

Dadas funções características f, g, é sempre possível definir funções características "união" $f \cup g$ e "intersecção" $f \cap g$ que verificam

$$(f \cup g)(x)\downarrow \;\Leftrightarrow\; f(x)\downarrow \vee q(x)\downarrow \qquad , \qquad (f \cap g)(x)\downarrow \;\Leftrightarrow\; f(x)\downarrow \wedge q(x)\downarrow$$

Pode-se pensar numa função parcial $\overline{f}$, cujo único resultado definido seja 0, e que verifique

$$\overline{f}(x)\downarrow \quad \Leftrightarrow \quad f(x)\uparrow$$

Esta construção, porém, não gera necessariamente uma função recursiva. Obviamente, se $\overline{f}$ for recursiva, será uma função característica.

Neste contexto, a caracterização das funções primitivas transfere-se para sub-conjuntos dos naturais.

30 Noção (Conjuntos recursivamente enumeráveis e conjuntos recursivos)
Um conjunto A é **recursivamente enumerável (r.e.)** *se é o domínio de uma função característica f; neste contexto*

$$x \in A \quad \Leftrightarrow \quad f(x)\downarrow$$

Um conjunto A diz-se **recursivo** *quando ele e o seu complemento $\overline{A}$ são ambos recursivamente enumeráveis. Equivalentemente, quando A é r.e. e existe uma relação recursiva ρ tal que $x \in A \Leftrightarrow \rho(x)$.*

O conjunto recursivo A, que verifica $(\forall x)[x \in A]$, designa-se por ϖ; extensionalmente ϖ coincide com $\mathbb{N}$. O conjunto recursivo A, para o qual o predicado $(\exists x)[x \in A]$ não é válido, designa-se por $\emptyset$.

Existe uma assimetria fundamental entre a verificação da asserção $x \in A$ quando x pertence ou não ao conjunto. Se A for só recursivamente enumerável, verificar $x \in A$ requer uma quantidade finita de informação enquanto que verificar $x \notin A$ pode necessitar de informação infinita; isto deriva, mais uma vez, da indecidibilidade da verificação de que uma função é definida num argumento particular. No entanto, se A for um conjunto recursivo, verificar $x \in A$ ou $x \notin A$ requer, em ambos os casos, informação finita.

Uma das primeiras consequências da enumeração das funções recursivas parciais é

31 Teorema (Grafos)
Uma função f é parcial recursiva se e só o seu grafo for recursivamente enumerável. Uma função f é recursiva se e só se o seu grafo for recursivo.

Como consequência da proposição 29 e do facto 12 tem-se

32 Teorema (Consistência Estrutural dos Conjuntos r.e.)

1. *Os conjuntos recursivamente enumeráveis formam um reticulado distributivo* **ReSet** *com o limite superior* $\emptyset$ *e limite superior* ϖ. *Os conjuntos recursivos* **RSet** *são, neste reticulado, os elementos que têm complemento (equivalentemente,* **RSet** *é a maior álgebra booleana contida em* **ReSet***).*
2. *Se* A *é recursivamente enumerável e* f *é uma função parcial recursiva, então tanto* $f(A)$ *como* $f^{-1}(A)$ *são recursivamente enumeráveis. Se* A *é recursivo e* f *é recursivo então* $f^{-1}(A)$ *é recursivo.*
3. *Se* A *é finito então é recursivo. Se* A *é infinito e recursivamente enumerável, então contém um subconjunto infinito que é recursivo.*

O item (2) implica que, sendo A e f ambos recursivos, o conjunto $f^{-1}(A)$ é recursivamente enumerável; no entanto não força ele seja necessariamente recursivo.

Conjuntos, que sendo infinitos, não contêm qualquer sub-conjunto infinito que seja recursivo, dizem-se **imunes**. O item (3) diz-nos que nenhum conjunto recursivamente enumerável é imune. Por isso, conjuntos imunes podem não parecer úteis; porém, como veremos adiante, existem conjuntos imunes extremamente relevantes à Criptografia.

□

A classe $\mathcal{RP}$ pode, no entanto, ser demasiado lata para caracterizar "computabilidade efectiva". Por isso é frequente seleccionar uma qualquer classe de funções efectivamente computáveis $\mathcal{C}$, tomá-la como referência e considerar

apenas as funções características que caem nessa classe.

Nestas circunstâncias pode-se modificar a classificação de conjuntos e dizer

33 Noção (Conjunto computávelmente enumeráveis e computáveis)
Um conjunto A é **computavelmente enumerável (c.e.)** *se é o domínio de uma função característica efectivamente computável. O conjunto é* **computável** *se ele e o seu complemento são computavelmente enumeráveis.*

Porém a classe $\mathcal{C}$ das funções consideradas efectivamente comutáveis tem de ser suficientemente lata para que se verifique um resultado de consistência estrutural análogo ao teorema 32. Nomeadamente, $\mathcal{C}$ tem de conter suficientes elementos para que a classe dos conjuntos computavelmente enumeráveis forme um reticulado distributivo com limite inferior $\emptyset$ e limite superior $\mathbf{0}$ que contenha a imagem e a pré-imagem, por uma função em $\mathcal{C}$, de qualquer conjunto computavelmente enumerável.

Diagonalização

Considere-se a enumeração standard das funções parciais recursivas $\{\varphi_n\}$ introduzida no teorema da normalização (teorema 15, página 23). Seja φ a função universal; isto é a função parcial recursiva tal que, para todo x

$$\varphi_n(x) \simeq \varphi(n\|x)$$

Seja r uma qualquer função recursiva: *como caracterizar o conjunto* $\left\{\varphi_{r(n)}\right\}$ *das funções enumeradas por* r*?*

Nomeadamente $x \mapsto \varphi_{r(x)}(x)$, onde o argumento determina simultaneamente o programa e o argumento ao qual a função é aplicada, é uma função parcial recursiva porque, pela normalização, $\varphi_{r(x)}(x) \simeq \varphi(r(x)\|x)$. Se esta função é ou não recursiva é um problema que iremos discutir.

Uma outra questão fundamental no estudo dos conjuntos recursivamente enumeráveis: *existe algum conjunto recursivamente enumerável que não seja recursivo*? Veremos, em seguida, que a resposta é afirmativa e que ela se baseia nas propriedades de uma função deste tipo.

Este tipo de questões, e as técnicas a elas associadas, designam-se por **diagonalização** e são essenciais ao estudo de problemas essenciais da computabilidade. Exemplos de técnicas de diagonalização aparecem nos seguintes resultados.

34 Teorema

1. *Não existe nenhuma função recursiva r que enumere o conjunto das relações recursivas.*
2. *Existe um conjunto $\mathcal{K}$, definido por*

$$x \in \mathcal{K} \quad \Leftrightarrow \quad \varphi_x(x)\downarrow$$

que é recursivamente enumerável e não é recursivo.

Esboço de prova

1. Suponhamos que existia r recursiva tal que $x \mapsto \varphi_{r(x)}$ enumerava todas as relações recursivas. Então pode-se definir uma nova relação recursiva $u(x) \simeq 1 - \varphi_{r(x)}(x) = 1 - \varphi(r(x)\|x)$ e, por isso, existe algum n tal que $u \simeq \varphi_{r(n)}$. Obtém-se uma contradição porque, pela definição, tem-se $u(n) = 1 - \varphi_{r(n)}(n)$ e, pela enumeração, tem-se $u(n) = \varphi_{r(n)}(n)$.

2. A "função diagonal" $\delta : x \mapsto \varphi_x(x) = \varphi(x\|x)$ é parcial recursiva; como $x \in \mathcal{K} \Leftrightarrow \delta(x)\downarrow$ o conjunto $\mathcal{K}$ é recursivamente enumerável. Não é recursivo porque, se fosse, existiria a função característica do seu complemento e, portanto, um programa n tal que $x \notin \mathcal{K} \Leftrightarrow \varphi_n(x)\downarrow$ para todo x. Particularizando para $x = n$ resultaria uma contradição

$$n \in \mathcal{K} \quad \Leftrightarrow \quad \varphi_n(n)\downarrow \quad \Leftrightarrow \quad n \notin \mathcal{K}$$

O resultado no item (1) é frequentemente apresentada aproveitando a ligação entre as funções recursivas e as máquinas de Turing e associando a parcialidade das funções ao problema da paragem das máquinas de Turing. Nesse contexto é apresentado como

> *Não existe nenhuma máquina de Turing que, sob input x, decida se a máquina de Turing universal pára nesse input x.*

□

Representação

Uma propriedade essencial dos conjuntos está na forma como os seus elementos são construídos como resultados de funções; essa "representação" dos conjuntos é uma consequência imediata do teorema da normalização.

35 NOÇÃO (CONJUNTOS REPRESENTÁVEIS)
O conjunto A é ***recursivamente representável*** *(ou só,* ***representável)****) se é vazio ou então é o contradomínio de uma função recursiva. Tal função designa-se por* ***representação*** *de A.*

Equivalentemente, um conjunto A é **representável** *se for o contradomínio de uma função parcial recursiva.*

Para mostrar a equivalência entre as duas formas partimos de um resultado intermédio

36 LEMA *Seja f uma função parcial recursiva. Então, existe uma função recursiva cujo contradomínio coincide com o contradomínio de f.*

Esboço de prova

Seja a um elemento arbitrário do contradomínio de f e seja e um programa, na enumeração standard das funções parciais recursivas, tal que $f \simeq \varphi_e$. Defina-se

$$f_a(z) \simeq \begin{cases} y & \text{se } \varphi_e^l(x) \simeq y \quad \text{sendo } \pi_1(z) \simeq l \ , \ \pi_2(z) \simeq x \\ a & \text{em caso contrário} \end{cases}$$

Como $x \mapsto \varphi^l(e\|x)\downarrow$ é uma relação primitiva recursiva, esta função é recursiva cujo contra-domínio coincide com o de f.

Existe agora uma relação básica entre as noções de enumeração recursiva e representatividade.

37 TEOREMA (REPRESENTAÇÃO)

1. *Um conjunto A é recursivamente enumerável sse for recursivamente representável.*
2. *Um conjunto $A \neq \emptyset$ é recursivo sse for representável por uma função recursiva, monótona e não-decrescente.*

Esboço de prova

1. Se A for recursivamente enumerável com a função característica c, então é o contradomínio da função parcial recursiva $t(x) \simeq x+c(x)$. A função t é parcial recursiva com contradomínio coincidente com A. Usando o lema 36 constrói-se uma função recursiva f com o mesmo contradomínio.

   Inversamente se A é o contradomínio de uma função recursiva r, constrói-se uma função característica f por

   $$f(x) \simeq 0 \cdot \mu y.[r(y) = x]$$

2. Neste caso é simples construir uma função recursiva usando a recursividade-μ para procurar, por ordem crescente, os elementos do conjunto. A seguinte função é recursiva, é monótona não decrescente e enumera os elementos de A.

   $$r(0) = \mu x.[x \in A] \quad , \quad r(n+1) = \begin{cases} \mu P_n & \text{se } (\exists x)\, P_n(x) \\ r(n) & \text{em caso contrário} \end{cases}$$

   $$\text{sendo} \qquad P_n(x) = [x \in A \ \wedge\ x > r(n) \ \wedge\ x \leq n + r(n)]$$

   Note-se que, porque $(x \in A)$ é uma relação recursiva (já que A é recursivo), também $P_n(x)$ e $(\exists x)\, P_n(x)$ são relações recursivas.

□

Indexação

Tal como para as funções parciais recursivas, é possível indexar os conjuntos recursivamente enumeráveis. Note-se que, dada uma qualquer função parcial recursiva φ_e é possível construir uma função característica $c_e \simeq 0 \cdot \varphi_e$ que

tem precisamente o mesmo domínio. Por isso é possível ver os conjuntos r.e. como domínios de funções parciais recursivas. Representaremos por $\mathcal{W}_e$ o domínio da função parcial recursiva de índice e.

Porém podem existem, normalmente, muitas funções parciais diferentes que têm exactamente o mesmo domínio A; como cada uma destas funções tem um índice diferente, a cada conjunto r.e. A está associado um conjunto de **r.e. índices**. Em termos de informação, basta o conhecimento de um desses índices para determinar o conjunto.

No caso particular em que o conjunto A é recursivo, e para além dos r.e. índices que possui como qualquer conjunto r.e., está associado aos índices das várias relações recursivas ρ que o determina; estes são os **índices característicos**.

Finalmente quando A é um conjunto finito tem, também, um único **índice canónico** (A) que é o inteiro a cuja representação binária tem o valor 1 precisamente nos elementos de A; isto é, $(A) = \sum_{i \in A} 2^i$.

Designaremos por D_a o conjunto finito cujo índice canónico é a; por convenção $D_0 = \emptyset$. Portanto $(D_a) = a$ e $D_{(A)} = A$. Designaremos por $\mathcal{S}_a$ a classe dos superconjuntos de D_a que são recursivamente enumeráveis

$$\mathcal{W}_x \in \mathcal{S}_a \Leftrightarrow D_a \subseteq \mathcal{W}_x \tag{20}$$

É essencial atender ao facto que $\mathcal{S}_a$ não contém qualquer superconjunto de D_a que não seja recursivamente enumerado. Esta classe representa todas as computações que "continuam" o conjunto codificado por a.

Pode-se extender o conceito de classe de conjuntos representando "continuações computáveis"; para tal, em vez de considerarmos um só índice a, vamos fazer a percorrer um conjunto A que seja recursivamente enumerável.

38 Noção (Classe efectivamente enumerável de conjuntos)
Uma classe de conjuntos recursivamente enumeráveis $\mathcal{A}$ é **efectivamente enumerável**[8]*, se existe um conjunto recursivamente enumerável A, chamado* **representação** *de $\mathcal{A}$, tal que*

$$\mathcal{W}_x \in \mathcal{A} \quad \Leftrightarrow \quad (\exists a)\,[a \in A \wedge D_a \subseteq \mathcal{W}_x\,] \tag{21}$$

Equivalentemente, se existe uma funcional efectiva F tal que

$$\mathcal{W}_x \in \mathcal{A} \quad \Leftrightarrow \quad \varphi_x \in F^{\tilde{}} \tag{22}$$

Uma visão alternativa das classes efectivamente enumeráveis é dada pelo seguinte resultado.

39 Facto
Uma classe de conjuntos recursivamente enumeráveis $\mathcal{A}$ é efectivamente enumerável se e só se $\{\, x \mid \mathcal{W}_x \in \mathcal{A}\,\}$ é um conjunto recursivamente enumerável.

Uma noção mais fraca é

[8] Também designada por **completamente recursivamente enumerável** ou por **Σ_1^0-classe**.

40 Noção

Uma classe $\mathcal{A}$ *de conjuntos recursivamente enumeráveis diz-se* **recursivamente enumerável** *se existe uma função recursiva* r *que enumera um índice de cada um dos seus elementos; isto é,* $\mathcal{A} = \left\{\mathcal{W}_{r(e)}\right\}_{e\in\varpi}$.

Em (21), sem perda de generalidade, pode-se considerar que os elementos da representação A codificam conjuntos finitos disjuntos dois a dois. Isto é, A verifica

$$(a \in A) \wedge (b \in A) \quad \Rightarrow \quad D_a \cap D_b = \emptyset \tag{23}$$

41 Noção (Array)

Um conjunto recursivamente enumerado A *que verifica* (23) *designa-se por* **array** [9].

42 Facto

Uma classe é efectivamente enumerado sse existe um array A *que a represente.*

□

Existe uma relação forte entre conjuntos finitos e naturais que é estabelecida pela correspondência $a \leftrightarrow D_a$ dada pela indexação canónica. Isto faz com que algumas noções relevantes a objectos de nível 0 (naturais) se extendam para alguns objectos de nível 1 (conjuntos de naturais). Por exemplo,

[9] No Odifreddi (pag. 228) este conceito é designado por "strong disjoint array".

43 NOÇÃO

Se A é um conjunto finito então o seu ***índice de Kolmogorov*** $\kappa(A)$ *é o índice de Kolmogotov do seu índice canónico; isto é,* $\kappa(D_a) \doteq \kappa(a)$.

A é ***compressível*** *quando o seu índice canónico é compressível e é* ***κ-aleatório*** *se não for compressível.*

Conjuntos "quase" recursivos

Vimos que a diferença essencial entre um conjunto recursivo (ou computável) e um conjunto r.e (ou c.e.) e é que, no primeiro caso, o predicado $x \in A$ é recursivamente verificável, quer seja ou não válido, enquanto que, no 2º caso, o predicado só é recursivamente verificável se for válido.

Existem conjuntos que são recursivamente enumeráveis e que, não sendo recursivos, são "quase recursivos". Temos, pelo menos, duas classes intermédias importantes: a dos **conjuntos semi-recursivos** e a dos **conjuntos simples**.

44 NOÇÃO (CONJUNTOS SEMI-RECURSIVOS)

Uma função recursiva σ é uma ***função de escolha*** *se, para todos x, y se verifica sempre $\sigma(x\|y) = x$ ou $\sigma(x\|y) = y$. Um conjunto A é* ***semi-recursivo*** *se existe uma função de escolha σ tal que, para todos x, y*

$$(x \in A) \Rightarrow \sigma(x\|y) \in A \qquad e \qquad (y \in A) \Rightarrow \sigma(x\|y) \in A$$

Note-se que se A for recursivo, com informação finita, é possível decidir se $x \in A$ e $y \in A$ e, desta forma, decidir $f(x\|y) \in A$. Porém, se A for só recursivamente enumerável, informação finita pode não ser suficiente para decidir $x \notin A$ e $y \notin A$; portanto pode não haver informação suficiente para decidir $f(x\|y) \in A$. Desta forma se A for recursivo é semi-recursivo mas se for só r.e. pode não ser semi-recursivo. Ao invés, se for semi-recursivo é recursivamente representável.

□

Outra noção "intermédia" é a de **conjunto simples**. Conjuntos simples ligam-se à noção de compressibilidade e, por isso, à de aleatoriedade. Para se ver como, temos de recorrer a uma outra noção: a de **conjunto retráctil**.

Suponhamos que A é recursivamente enumerável por uma função total (não necessariamente recursiva) crescente r; isto é $A = \{a_x = r(x)\}_{x \in \varpi}$. O conjunto diz-se **retráctil** se existe uma função parcial recursiva σ tal que

$$\sigma(a_{x+1}) \simeq a_x \quad , \quad \sigma(a_0) \simeq a_0$$

45 FACTO

Se o conjunto A é recursivo é retráctil. Se A e o seu complemento $\overline{A}$ são retracteis, então A é recursivo. Se A é retráctil e não é recursivo, é imune [10].

46 NOÇÃO (CONJUNTOS SIMPLES)

Um conjunto é ***simples*** *se é recursivamente enumerável e o seu complemento é imune.*

[10] Recordemos que imunes são os conjuntos infinitos que não contém qualquer subconjunto infinito recursivamente enumerável.

O seguinte resultado exprime um conceito fundamental da aleatoriedade de Kolmogorov

47 Teorema (Kolmogorov)
O conjunto dos números κ-compressíveis (não-aleatórios) é simples.

Isto significa que o conjunto dos números aleatórios é infinito e não contém nenhum sub-conjunto infinito que seja recursivamente enumerável; para além disso, os não-aleatórios formam um conjunto recursivamente enumerável.

Reducibilidade de Turing

48 Noção
Para uma função total g, a classe de todas as funções **recursivas em** g *é a menor classe de funções que contém g e é fechada por composição, recursividade primitiva e recursividade μ limitada.*

Se f é recursivo em g representamos esse facto por $f \leqslant_T g$ e diz-se que é **Turing redutível** *a g. Se $f \leqslant_T g$ e $g \leqslant_T f$ diz-se que f e g são* **Turing equivalentes** *e escreve-se $f \equiv_T g$.*

As classes de equivalência de funções totais definidas pela relação $\equiv_T$ designam-se por **graus de Turing**.

Os graus de Turing colocam na mesma classe as funções que são recursivas em relação umas às outras usando recursividade limitada; portanto, duas funções no mesmo grau podem-se considerar redutíveis uma à outra em termos de complexidade computacional.

O resultado seguinte ajuda a perceber a estrutura destes objectos computacionais.

49 Proposição
Cada grau de Turing $\boldsymbol{\alpha}$ contém pelo menos um conjunto e contém qualquer conjunto recursivo.

Esboço de prova
Dada um representante arbitrário $f \in \boldsymbol{\alpha}$, seja g o seu grafo. Claramente, como f é uma função total, existe uma construção de g recursiva em f e, inversamente, existe uma construção de f que é recursiva em g (teorema 31, página 34). Como cada grafo é uma relação e, por isso, determina um conjunto. Por outro lado, são Turing equivalentes a f todas as funções totais recursivas. Nomeadamente todas as relações recursivas e, deste modo, todos os conjuntos recursivos.

Recordemos o conjunto $\mathcal{K} = \{\, x \mid \varphi_x(x)\downarrow \,\}$ que, como vimos no teorema 34, página 37, é recursivamente enumerável mas não é recursivo. Então

50 Teorema (Post)
Se A é um conjunto recursivamente enumerável então $A \leqslant_T \mathcal{K}$.

Se tomarmos os graus que contêm, pelo menos, um conjunto recursivamente enumerável (os chamados **graus recursivamente enumeráveis**) existem dois que têm importância especial:

(i) o grau representado por $\mathbf{0}$ que é formado exclusivamente pelas funções recursivas (equivalentemente, pelos conjuntos recursivos). A proposição 49 diz-nos que qualquer grau $\boldsymbol{\alpha}$ contém $\mathbf{0}$.

(ii) o grau representado por $\mathbf{0}'$ que é o menor grau que contém o conjunto $\mathcal{K}$. Pelo teorema 50 qualquer grau que contenha um conjunto recursivo está contido em $\mathbf{0}'$.

1.4 Domínios

...God made the integers, all else is the work of man...[11]

...God made the bit-strings, all else is the work of programmers...[12]

Tomemos por referência uma qualquer classe $\mathcal{C}$ de funções consideradas como efectivamente computáveis. Tipicamente escolhe-se PPTIME.

No contexto deste curso designamos por **domínio** qualquer colecção de objectos que podem ser posto em correspondência biunívoca com um sub-conjunto dos números naturais $\mathbb{N}$ ou uma sub-classe do respectivo "power-set" $\wp(\mathbb{N})$.

51 NOÇÃO (DOMÍNIO ENUMERÁVEL)
Um domínio X que pode ser posto em correspondência biunívoca com um sub-conjunto recursivo $R \subseteq \mathbb{N}$ designa-se por ***domínio numerável****. Se existir uma representação computável $\psi : \mathbb{N} \rightarrow X$, o domínio X diz-se* ***enumerável****.*

Um domínio que está em correspondência biunívoca com todo $\mathbb{N}$ ***identifica-se*** *com ϖ.*

[11] LEOPOLD KRONECKER- 1823/1891
[12] senso comum!

Um sub-conjunto $A \subseteq X$ diz-se **recursivamente enumerável** *(***recursivo***) se está em correspondência biunívoca com um conjunto recursivamente enumerável (recursivo) de ϖ.*

Todo o sub-conjunto $Y \subseteq \mathbb{N}$ é ordenável pela ordem nele induzido pela ordem natural dos inteiros. Todo o domínio enumerável $X \cong Y$ é ordenável pela ordem $\preceq$ nele induzido pela sua correspondência com Y.

Exemplo 2(Enumerações):

Para definir um domínio enumerável basta identificar o "primeiro" elemento do domínio e o "sucessor" de um elemento genérico $x \in X$.

Um domínio enumerável, definido deste modo, é o conjunto $\mathbb{Q}$ dos números racionais. É bem conhecida a enumeração por "diagonalização" que ao racional n/d faz suceder o racional $(n+1)/(d-1)$, caso $d > 1$, ou o racional $1/(n+1)$ caso $d = 1$.

A enumeração dos pares de naturais $\mathbb{N} \times \mathbb{N}$ é análoga à dos racionais: o primeiro elemento do domínio é o par $(0,0)$ e a função "sucessor" mapeia o par (x,y) no par $(x+1, y-1)$, caso seja $y > 0$, ou no par $(1, x+1)$ caso seja $y = 0$.

52 Noção (Representação e Codificação)
Se X é um qualquer domínio enumerável, uma função sobrejectiva $r\colon \mathbb{N} \to X$ é uma **representação** *de X.*

A família de conjuntos $\left\{r^{-1}(x)\right\}_{x\in X}$ é uma partição de $\mathbb{N}$; i.e. os conjuntos são mutuamente disjuntos e cobrem todo o domínio Os inteiros $n \in r^{-1}(x)$ designam-se por r-**códigos** de x.

Uma função $s\colon X \longrightarrow \mathbb{N}$ que mapeie cada $x \in X$ num dos seus r-códigos[13] designa-se por r-**codificação** de X.

bit-strings

O domínio enumerável que usaremos mais frequentemente é, porém, o domínio das bit-strings finitas $\mathbb{B}^*$. Importante (mas não enumerável) é, também, o domínio $\mathbb{B}^\infty$ das bit-strings infinitas. Notações alternativas para estes domínios são, respectivamente, $2^{<\varpi}$ e 2^{ϖ}.

Apesar de, como veremos em seguida, $2^{<\varpi}$ e 2^{ϖ} se identificarem, respectivamente, com os naturais e os conjuntos de naturais, exigem algumas noções e notações que lhe são específicas.

53 NOÇÃO
Se x é uma string finita, existe um único n tal que $x \in \mathbb{B}^n$; esse inteiro é o ***comprimento*** *de x e é representado por $|x|$. Se u é uma string infinita, convenciona-se que $|u| = +\infty$. Se u é uma string finita ou infinita, para todo $0 \leq i < |u|$, a componente de ordem i de u representa-se por u_i.*

[13] Note-se que a codificação é uma "inversa-à-direita" de r; isto é verifica-se $r(c(x)) = x$, para todo $x \in X$. Como consequência a cardinalidade de X é sempre menor ou igual à cardinalidade de $\mathbb{N}$; donde, nenhum domínio não enumerável pode ser representável.

A **concatenação** de uma string finita x com uma string finita ou infinita υ é um bit-strings representada por $x\,\upsilon$ tal que $|x\,\upsilon| = |x| + |\upsilon|$ e

$$(x\,\upsilon)_i = \begin{cases} x_i & \text{se} \quad i < |x| \\ \upsilon_{i-|x|} & \text{se} \quad i \geq |x| \end{cases}$$

Representa-se por $x \leq \upsilon$ quando existe uma terceira string s tal que $x\,s = \upsilon$. Neste caso diz-se que x é um **prefixo** de υ; um prefixo de comprimento n diz-se um **n-prefixo**. Inversamente, dada uma string finita ou infinita υ, representa-se por $\downarrow\upsilon$ o conjunto dos prefixos de υ; isto é, $\downarrow u = \{\, x \mid x \leq u \,\}$.

Representa-se por $\uparrow u$ o conjunto de todas as strings infinitas que têm u como prefixo.

$$\uparrow x \;=\; \{\, u \in \mathbb{B}^\infty \mid x \leq u \,\} \;=\; \{\, x\,\upsilon \mid \upsilon \in \mathbb{B}^\infty \,\} \tag{24}$$

Um sub-conjunto $L \subseteq 2^{<\varpi}$ designa-se por **linguagem**. A linguagem L diz-se **livre de prefixos** quando nenhum dos seus elementos é prefixo de um outro elemento da mesma linguagem. Isto é, para todos $x \neq y \in L$, nunca ocorre $x \leq y$; equivalentemente $\uparrow x$ e $\uparrow y$ são sempre disjuntos.

Por exemplo, a string finita ou infinita υ determina a linguagem $\downarrow\upsilon$; neste caso, a linguagem tem cardinalidade igual ao comprimento da string. Da mesma forma, cada linguagem L gera uma outra linguagem, representada como $\downarrow L$, formada pelos prefixos de elementos de L; isto é, $\downarrow L = \{\, x \mid \exists y \in L : x \leq y \,\} = \bigcup_{y \in L} \downarrow y$.

Nota

É elementar provar que, dadas duas linguagens L e H, se for $L \subseteq H$, então terá de ser $\downarrow L \subseteq \downarrow H$. Porém, o inverso não é necessariamente válido: pode ocorrer $\downarrow L \subseteq \downarrow H$ sem que $L \subseteq H$ ocorra.

Considere-se, por exemplo, as linguagens $L = (1\,1)^*$ e $H = 1\,(1\,1)^*$. A primeira é formada por todas as sequências de 1´s de comprimento par; a segunda é idêntica mas as strings têm comprimento ímpar. As linguagens são distintas mas têm o mesmo conjunto de prefixos; $\downarrow L = \downarrow H = 1^*$.

Se L é uma linguagem livre de prefixos, representamos por $\uparrow L$ a colecção de strings infinitas que contêm algum elemento de L como prefixo; isto é

$$\upsilon \in \uparrow L \quad \Leftrightarrow \quad (\exists^1 x \in L)\,[\,x \leq \upsilon\,] \tag{25}$$

Note-se que, porque L é livre de prefixos, os diversos $\uparrow x$, com $x \in L$, são disjuntos. Assim, cada $u \in \uparrow L$ tem, quanto muito, um prefixo contido em L.

54 FACTO

Se L é uma linguagem livre de prefixos então $\sum_{x \in L} 2^{-|x|} \leq 1$.

Esboço de Prova

Seja l_k o número de elementos de L de tamanho k. O número de strings de comprimento n que têm um prefixo de tamanho $k < n$ em L é, portanto, $2^{n-k}\, l_k$. Portanto $l_n \leq 2^n - \sum_{k<n} l_k\, 2^{n-k}$; o que implica $\sum_{k=0}^{n} l_k 2^{-k} \leq 1$, independentemente do valor de n. Tomando o limite $n \to \infty$, e notando que $\sum_{x \in L} 2^{-|x|} = \sum_{k=0}^{\infty} l_k 2^{-k}$, conclui-se que $\sum_{x \in L} 2^{-|x|} \leq 1$.

Representação das bit-strings finita

Para vermos que $\mathbb{B}^*$ é enumerável e identificável com $\mathbb{N}$ basta representar os elementos deste conjunto numa árvore binária enumerando os seus nodos como se indica na figura 2.

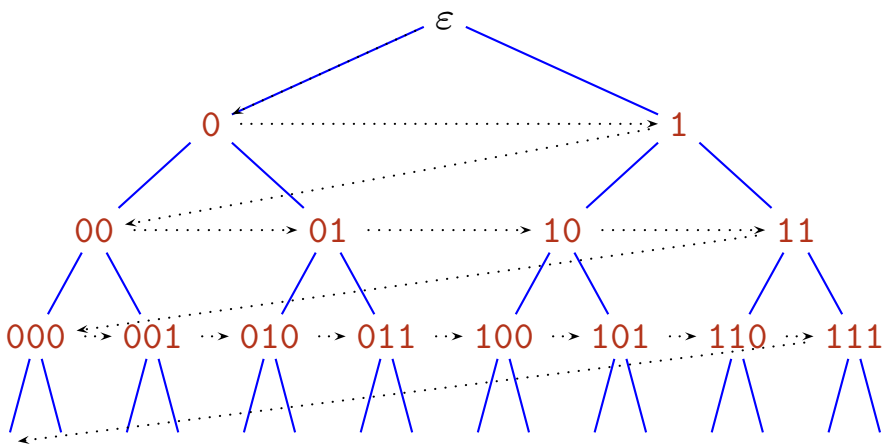


Figura 2: Enumeração das bit-strings finitas.

A função representação $r : \mathbb{N} \longrightarrow \mathbb{B}^*$, e a respectiva codificação $s : \mathbb{B}^* \longrightarrow \mathbb{N}$, constroem-se a partir das funções biunívocas $r_n : [0, 2^n - 1] \longrightarrow \mathbb{B}^n$, que constituem a representação standard em n bits dos naturais, e as respectivas inversas $s_n = {r_n}^{-1}$ que codificam strings finitas em naturais.

Então

$$r(x) \;=\; r_{|x|}\Big(x - (2^{|x|} - 1)\Big) \qquad , \qquad s(u) \;=\; (2^{|u|} - 1) + s_{|u|}(u) \tag{26}$$

Considere-se a representação e codificação standard de bit-strings finitas defindas em (26). Dado que $2^{<\varpi}$ e ϖ se identificam, a cada linguagem $L \subseteq 2^{<\varpi}$ corresponde univocamente um conjunto de naturais formado pelos códigos das strings em L: a cada linguagem L está associado um único conjunto $A_L \in 2^{\varpi}$ dada por

$$A_L \;=\; \{\, s(u) \;\mid\; u \in L \,\}$$

Inversamente, a cada conjunto finito $A \in 2^{\varpi}$ corresponde uma única linguagem $L_A \subseteq 2^{<\varpi}$ formado pelas bit-strings representadas pelos elementos de A; isto é,

$$L_A \;=\; \{\, r(a) \;\mid\; a \in A \,\}$$

Desta forma o conjunto de todas as linguagens $2^{2^{<\varpi}}$ identifica-se com o conjunto de todos os conjuntos de naturais $\wp(\mathbb{N})$, e ambos são designados por 2^{ϖ} .

As caracterizações usuais dos conjuntos de naturais transferem-se directamente para o domínio das linguagens. Por exemplo

55 NOÇÃO

A linguagem L é ***recursivamente enumerável*** *(****recursiva****) se o conjunto A_L é recursivamente enumerável (recursivo). A linguagem L é* ***finita*** *quando A_L é finito. L é* ***compressível*** *(κ-aleatório) se é finita e A_L é compressível (κ-aleatório).*

A identificação entre conjuntos de naturais e linguagens pode ser extendida a outras classes de objectos. Nomeadamente

56 NOÇÃO

- *Cada conjunto A identifica-se com a relação α tal que $\alpha(x) \simeq 1 \Leftrightarrow x \in A$.*
- *Cada relação α identifica-se a bit-strings infinita u que verifica $\alpha(x) \simeq 1 \Leftrightarrow u_x = 1$.*
- *Cada conjunto A identifica-se com a string u que verifica $x \in A \Leftrightarrow u_x = 1$.*
- *Cada linguagem L identifica-se com a string infinita associada ao conjunto A_L.*

Estes vários domínios representam facetas concretas do domínio abstracto 2^{ϖ}; para cada uma destas facetas pode-se ver uma noção de "truncatura".

57 NOÇÃO

Dado $A \in 2^{\varpi}$, a ***truncatura*** *$A \restriction n$ denota o prefixo de comprimento n da string infinita que representa A .*

Para $x \in \varpi$, o predicado $(\exists n)\,[A \restriction n \simeq x]$ é representado por $x \prec A$. A classe $\{ A \in 2^{\varpi} \mid x \prec A \}$ é representada por $\uparrow x$. O conjunto $\{ x \in \varpi \mid x \prec A \}$ é representado por $\downarrow A$.

Vendo $A \in 2^{\varpi}$ como string infinita, a truncatura $A \restriction n \in \varpi$ é a string formada pelos primieros n bits de A mas também pode ser vista como o natural $s(A \restriction n)$ que a codifica. Vendo $A \in 2^{\varpi}$ como um conjunto, $A \restriction n$ identifica-se também com o sub-conjunto $\{ x \mid x \in A \wedge x \leq n \}$.

Reais

Indistintamente 2^{ϖ} identifica o domínio das bit-strings infinitas, o domínio dos conjuntos de naturais e o domínio das linguagens. Falta um quarto domínio: o intervalo real unitário $\mathbb{I} = [0, 1]$.

Comecemos pelas bit-strings finitas e a sua relação com uma determinada classe de números recaionais.

58 Noção (Racional de Lebesgue)
Uma string finita $x \in \mathbb{B}^*$ (equivalentemente, um natural x) determina um racional

$$x^{\tilde{}} = \sum_{k<|x|} x_k \, 2^{-k-1} \tag{27}$$

Números racionais desta forma (somas finitas de fracções 2^{-k}) chamam-se ***racionais de Lebesgue***.

É importante notar que os racionais de Lebesgue são limitados superiormente por 1 e que, se for $u \leq v$ verifica-se $u^{\tilde{}} \leq v^{\tilde{}}$. Portanto a função $\cdot^{\tilde{}} \colon \mathbb{B}^* \to \mathbb{Q}_+$ é monótona crescente e limitada a 1.

A noção de racional de Lebesgue permite construir uma forma "inversa" do facto 54 (página 53). Esse resultado diz-nos que, para uma linguagem livre de prefixos $L = \{x_n\}$, a sequência $\{d_n\}$, com $d_n = |x_n|$, verifica $\sum_n 2^{-d_n} \leq 1$. O resultado "inverso" está expresso no seguinte teorema.

59 TEOREMA (LEVIN, SCHNORR,CHAITIN, MILLER)
Seja $\{d_n\}$ uma sequência computavel de naturais tal que $\sum_n 2^{-d_n} \leq 1$. Então existe uma sequência computável $\{x_n\}$, livre de prefixos, tal que $|x_n| = d_n$, para todo n. Adicionalmente, se $\{d_n\}$ for recursiva, então $\{x_n\}$ enumera uma linguagem recursiva livre de prefixos.

Intuição

A construção da sequência $\{x_n\}$ é feita via racionais de Lebesgue. Considere-se, para cada n, $z_n \in 2^{<\varpi}$ a menor string tal que $(z_n)^{\tilde{}} = 1 - \sum_{i \leq n} 2^{-i}$; seja $t_n = |z_n|$. O último bit de z_n (i.e. $(z_n)_{t_n - 1}$ tem de ser 1 (caso contrário z_n não seria a menor string que satisfaz esta relação).

Através dos bits dos diversos z_n que são 1, a prova deste teorema constrói indutivamente uma sequência de linguagens livres de prefixos $L_0 \subset L_1 \subset \cdots L_n \subset \cdots$ de tal forma que $x_i \in L_n$, para todo $i \leq n$.

Tomando agora uma bit-strings infinita u, pode-se construir uma série de Cauchy no intervalo $[0, 1]$

$$u^{\tilde{}} \quad = \quad \sum_{k=0}^{\infty} u_k \, 2^{-k-1} \tag{28}$$

Note-se que $u^{\sim}$ é um real bem definido porque é o limite se uma sequência crescente de racionais limitada por 1. É também o limite dos racionais de Legesgue, determinados pelos prefixos $u \restriction n$ de u.

$$u^{\sim} = \lim_{n \to \infty} (u \restriction n)^{\sim} \tag{29}$$

Para simplificar a notação designemos por $x_n \in 2^{<\varpi}$ a string finita $x_n = u \restriction n$ que se obtém truncando u a n bits; $\downarrow u$ é a linguagem formada por todos estes prefixos. Uma característica da linguagem $\downarrow u$ é o ser *completamente ordenada*; isto é, $x, y \in \downarrow u$ implica $x \leq y$ ou $y \leq x$. Se, adicionalmente, a linguagem for recursivamente enumerável, ou recursiva, as diversas truncaturas $x_n = u \restriction n$ formam "aproximações computáveis" da string u; assim, u e o real de Lebesgue $u^{\sim}$ que determina são, de algum modo, "computáveis".

Por isso faz sentido definir,

60 NOÇÃO (REAL RECURSIVO (R.R.))
Um **real recursivo** *é um sub-conjunto recursivo* $R \subseteq \mathbb{Q}_+$ *que tem um limite superior. Equivalentemente é o limite de uma função recursiva, limitada e monótona não-decrescente* $r : \mathbb{N} \longrightarrow \mathbb{Q}_+$.

Um **real recursivo de Legesgue (r.L.)** *é uma linguagem* L *recursiva e totalmente ordenada; isto é,* $x, y \in L$ *implica* $x \leq y$ *ou* $y \leq x$.

Notas
Na definição de real recursivo, note-se que R (por ser recursivo) pode ser enumerado por uma função monótona e não decrescente. Esta

função determina uma sequência não-decrescente e limitada de racionais positivos; por definição de real, esta sequência determina um número real.

Claramente um real recursivo de Lebesgue é um real recursivo; como L é uam linguagem recursiva cujos elementos formam uma ordem total, pode ser enumerada por uma função recursiva, monótona, não-decrescente (teorema 37, página 39). Os elementos de L, assim enumerado, originam uma sequência não-decrescente de racionais de Lebesgue.

Vimos como associar strings infinitas $u \in 2^{\varpi}$ a reais $u^{\tilde{}}$ no intervalo unitário $[0,1]$. Como é que esta aplicação $\cdot^{\tilde{}}\colon 2^{\varpi} \longrightarrow [0,1]$ transforma classes de strings infinitas?

Tomemos, como exemplo, uma string finita $x \in 2^{<\varpi}$ e a classe $\uparrow x = \{\, x\,v \mid v \in 2^{\varpi} \,\}$ formada por todas as strings infinitas $u = x\,v$ que têm x como prefixo. Pela definição (28) tem-se

$$u^{\tilde{}} \;=\; x^{\tilde{}} \,+\, 2^{-|x|}\, v^{\tilde{}}$$

Quando v percorre todo o 2^{ϖ}, $v^{\tilde{}}$ percorre o intervalo $[0,1]$. Por isso, $u^{\tilde{}}$ percorre o intervalo $[x^{\tilde{}}\ ,\ x^{\tilde{}} + 2^{-|x|}]$. Isto justifica

61 FACTO
As classes em 2^{ϖ} da forma $\uparrow x$ estão em correspondência biunívoca com os intervalos $\mathbb{I}_x = [x^{\tilde{}}\ ,\ x^{\tilde{}} + 2^{-|x|}]$.

Os intervalos reais $\mathbb{I}_x$ referidos no resultado anterior designam-se por **intervalos de Lebesgue**. É relevante frisar que o intervalo é compacto e que o seu tamanho é $2^{-|x|}$, sendo $|x|$ o comprimento da string x que o determina.

□

Eventos

Vimos que conjuntos A se identificam com strings infinitas. Falta agora ver a forma como classes de conjuntos se identificam com colecções de strings infinitas.

Para isso é necessário introduzir algumas noções:

62 Noção (σ-Álgebra)
*Seja X um domínio. Uma família Σ de sub-conjuntos de X, fechada por uniões contáveis e complementos, designa-se por σ-**álgebra**. Os elementos de uma σ-álgebra designam-se por **eventos**.*

Notas Porque Σ é fechado por uniões contáveis, contém o conjunto vazio (união vazia); porque é fechado por complementos contém o próprio domínio X por ser o complemento de $\emptyset$. Finalmente, o complemento de uma união contável é uma intersecção contável; por isso Σ é também fechado pela intersecção contável de eventos.

O exemplo imediato de σ-álgbera $\Sigma_{\mathcal{L}}$ formada pelas uniões ou intersecções contáveis de intervalos de Lebesgue. Designaremos por **intervalos** os elementos desta σ-álgebra específica. Formalmente $\Sigma_{\mathcal{L}}$ é a menor σ-álgebra que contém todos os intervalos de Lebesgue.

□

Genericamente eventos generalizam a noção de intervalo. Por isso é importante definir algo que dê a noção de tamanho desses eventos de forma análoga à noção de tamanho nos intervalos.

63 NOÇÃO (MEDIDA EM σ-ÁLGEBRAS)
Se Σ é uma σ-álgebra, uma função $\mu\colon \Sigma \longrightarrow \mathbb{R}_+$ é uma **medida** *se verifica:*

1. $\mu(\bigcup_i \alpha_i) = \sum_i \mu(\alpha_i)$ *para qualquer enumeração $\{\alpha_i\}$ de eventos mútuamente disjuntos.*
2. *Se $\{\alpha_i\}$ é uma sequência decrescente ($\alpha_{i+1} \subseteq \alpha_i$ para todo i), então $\mu(\bigcap \alpha_i) = \lim_{i\to\infty} \mu(\alpha_i)$.*

A noção de medida traduz a ideia de que um "intervalo", que é a união de "intervalos" disjuntos, tem uma medida que é a soma da medida das parcelas; portanto a medida é uma função crescente nos eventos. Como caso particular, o evento vazio tem medida nula.

Quando os eventos formam uma cadeia descendente, as respectivas medidas devem decrescer. Impomos, adicionalmente, que o evento limite destas cadeia tenha uma medida que é o limite das medidas dos elementos da sequência.

Considere-se $\Sigma_{\mathcal{L}}$; isto é, a menor σ-álgebra que contém todos os intervalos de Lebesgue. A medida μ é, neste caso, a função que verifica $\mu(\mathbb{I}_x) = 2^{-|x|}$. Portanto μ associa a cada intervalo o seu tamanho.

Na perspectiva do domínio genérico 2^{ϖ}, pode-se construir uma generalização desta σ-álgebra.

64 Noção (σ-álgebra de Lebesgue)
No domínio 2^{ϖ} *a* ***σ-álgebra de Lebesgue***, *representada por* $\mathcal{L}$, *é a menor* σ*-álgebra que contém todos as classes da forma* $\uparrow x$ *com* $x \in \varpi$.

Nas σ-álgebras de Lebesgue, fixando um inteiro $n > 0$, têm importância particular os eventos que são formados por uniões contáveis de eventos elementares $\uparrow x$ em que $|x| \geq n$; estes eventos designam-se por ***n-eventos***.

Eventos da forma $\bigcap_{x \in A} \uparrow x$, quando A é um conjunto recursivamente enumerável, designam-se por **recursivamente enumeráveis**. Igualmente, se A for recursivo, o evento designa-se por **recursivo**.

65 Noção (Medidas e Comprimentos de Lebesgue)
Dado um real $1 \geq \epsilon > 0$, *uma* ϵ-***medida de Lebesgue*** *é uma medida em* $\mathcal{L}$ *tal que* $0 < \mu(\uparrow x) \leq 2^{-\epsilon\,|x|}$ *para todo* $x \in \varpi$. *Quando, para todo* $x \in \varpi$, *se tem* $\mu(\uparrow x) = 2^{-|x|}$, *a medida diz-se* ***standard***.

Fixando uma medida μ, *um evento* α *diz-se* ***nulo*** *se* $\mu(\alpha) = 0$. *O* ***comprimento de Lebesgue*** *do evento* α , *representado por* $\lambda(\alpha)$, *é* $+\infty$ *se* α *for nulo ou, caso não seja, é* $-\log_2 \mu(\alpha)$.

Se L *é uma linguagem livre de prefixos, designa-se por* ***medida*** *de* L *à medida de Lebesgue do evento* $\uparrow L$.

É importante nota-se que esta definição de σ-álgebra de Lebesgue é feita para o domínio abstracto 2^{ϖ} e serve para qualquer um dos domínios concretos a ele associados: bit-strings infinitas, conjuntos, relações e linguagens.

As classes elementares $\uparrow x$ são também definidas de forma abstracta; x pode ser um string de bits finita ou então ser um natural. A classe $\uparrow x$ representa, indistintamente, todas as strings infinitas com prefixo x (vendo x como string) ou todos os conjuntos que contém D_x (vendo x como natural), etc.

Notas Fixando um qualquer tamanho n, tem-se $2^{\varpi} = \bigcup_{|x|=n} \uparrow x$; donde $\mu(2^{\varpi}) \leq 2^n \cdot 2^{-\epsilon\, n}$, para todo n. Se admitissemos a hipótese de ser $\epsilon > 1$, a medida seria reduzida à situação trivial em que todos os eventos têm medida zero.

A medida standard de um evento $\uparrow x$ é sempre $2^{-|x|}$; por isso, nessa medida, tem-se sempre $\lambda(\uparrow x) = |x|$; isto é, o comprimento do evento elementar $\uparrow x$ coincide com o tamanho do objecto $x \in \varpi$ que o determina.

Se a medida não for standard, apenas se pode afirmar que $\mu(\uparrow x) \leq 2^{-\epsilon\,|x|}$, para um $\epsilon > 0$ que depende apenas de μ (e não de x). Neste caso pode-se afirmar que $\lambda(\uparrow x) \geq \epsilon\,|x|$.

EXEMPLO 3: Interpretando 2^{ϖ} como o domínio $\mathbb{B}^{\infty}$ das bit-strings infinitas, vamos ver que o conjunto singular $\Omega = \{u\}$ (com $u \in \mathbb{B}^{\infty}$) é um evento e, além disso, é um evento nulo.

Seja $x_n = u \restriction n$ a truncatura de u a n bits, e considere-se os eventos elementares $U_n = \uparrow x_n$. O evento U_n contém todas as strings que, até ao n-ésimo bit, coincidem com u. Note-se que $\Omega = \bigcap_n U_n$ e, porque é a intersecção contável de eventos, é também um evento.

Em qualquer medida de Lebesgue, tem-se $\mu(U_n) \leq 2^{-\epsilon\,|x_n|} = 2^{-\epsilon\, n}$. Pela definição de medida, notando que a sequência $\{U_n\}$ é decrescente e $\epsilon > 0$, tem-se

$$\mu(\Omega) \;=\; \lim_{n\to\infty} \mu(U_n) \;\leq\; \lim_{n\to\infty} 2^{-\epsilon\, n} \;=\; 0$$

Portanto Ω é nulo e o respectivo comprimento é $\lambda(\Omega) = +\infty$.

Recuperando a visão abstracta do domínio 2^{ϖ}, convém ver algumas formas particulares de eventos:

Eventos da forma $\Omega = \{A\}$, com $A \in 2^{\varpi}$, designam-se por **eventos singulares**. O exemplo anterir mostra que esses eventos, para uma qualquer medida de Lebesgue, são nulos.

Uniões contáveis de eventos singulares $\mathcal{A} = \bigcup_i \Omega_i$, com $\Omega_i = \{A_i\}$ e $A_i \in 2^{\varpi}$, designam-se por **eventos discretos**. Obviamente, para qualquer medida, $\mu(\mathcal{A}) = \sum_i \mu(\Omega_i) = 0$. Por isso os eventos discretos são também nulos.

EXEMPLO 4(CLASSE DOS CONJUNTOS FINITOS): Considere-se, por exemplo, a classe $\mathcal{F}$ formada por todos os conjuntos finitos. Como os conjuntos finitos estão em correspondência biunívoca com os seus índices canónicos, $\mathcal{F}$ é enumerável. Obviamente $\mathcal{F} = \bigcup_{u \in \mathcal{F}} \{u\}$ é um evento discreto e, por isso, é nulo.

Aproximaçãoes às classes efectivamente enumeráveis

No estudo da computabilidade estamos particularmente interessados nas classes de conjuntos efectivamente enumeráveis na perspectiva que representam possíveis "continuações computáveis" de uma determinada computação.

Estamos também interessados em avaliar, de alguma forma, o "tamanho" dessas classes usando o mecanismo dos eventos e das medidas a eles associados.

66 Noção

Seja $\mathcal{A} \subseteq 2^{\varpi}$ uma classe de conjuntos. Um evento α que verifique $\mathcal{A} \subseteq \alpha$ designa-se por **cobertura** *de $\mathcal{A}$. Se α for um n-evento designa-se por* **n-cobertura**.

67 Noção

Fixe-se uma medida de Lebesgue μ e uma classe $\mathcal{A} \subseteq 2^{\varpi}$. $\mathcal{A}$ é **nula**, *e escreve-se $\lambda(\mathcal{A}) = +\infty$, se tem uma cobertura nula. $\mathcal{A}$ não-nulo tem* **comprimento** *não-inferior a um real $r \in \mathbb{R}_+$, e escreve-se $\lambda(\mathcal{A}) \geq r$, se tem uma cobertura de comprimento r.*

É importante focar o seguinte ponto: os eventos, normalmente, não traduzem computações; são apenas usados para "cobrir" as classes efectivamente enumeráveis (essas sim, representam computações) e, deste modo, aferir as suas dimensões.

Para ilustrar este ponto, tome-se um qualquer natural x e compare-se a classe efectivamente enumerável $\mathcal{S}_x$ (ver (20) na página 41) e o evento elementar $\uparrow x$.

- Os conjuntos $A \in \mathcal{S}_x$ são os conjuntos recursivamente enumerados que contêm o conjunto D_x.
- A classe $\uparrow x$ é formada por todos os conjuntos A em que os elementos de A de comprimento $\leq |x|$ foram um sub-conjunto que coincide com D_x.

Para além de ambas as classes conterem D_x, existe muito pouco de comum entre elas. Alguns pontos a reter:

Na classe $\mathcal{S}_x$ estão todos os conjuntos finitos que contêm D_x; porém, qualquer classe de conjuntos finitos é (como vimos atrás) enumerável; portanto, é um evento discreto e, consequentemente, nulo. Por isso, numa medida do "tamanho" de $\mathcal{S}_x$, só são relevantes os seus conjuntos infinitos. O mesmo se pode dizer quando se toma uma classe efectivamente enumerável $\mathcal{A}$ que, como sabemos, é uma união contável de classes da forma $\mathcal{S}_x$. Para avaliar o seu "tamanho" só são relevantes os conjuntos infinitos.

Dado que x não é necessariamente um prefixo dos conjuntos em $\mathcal{S}_x$, surge a questão de saber quais são esses prefixos. Nomeadamente só são relevantes os prefixos dos $\mathcal{W}_e$ que sejam infinitos. Define-se, assim, $\downarrow\mathcal{S}_x$ como o conjunto de todos os prefixos de conjuntos recursivamente enumeráveis infinitos que contêm D_x.

$$z \in (\downarrow\mathcal{S}_x) \quad \Leftrightarrow \quad (\exists e)\,[\, z \prec \mathcal{W}_e \ \wedge\ \mathcal{W}_e \text{ não finito} \ \wedge\ \mathcal{W}_e \supseteq D_x \,] \tag{30}$$

68 NOÇÃO

Sendo $\mathcal{A}$ a classe efectivamente enumerável representada pelo "array" A, os **prefixos** *de $\mathcal{A}$ formam o conjunto, representado por de $\downarrow\mathcal{A}$, que verifica*

$$z \in (\downarrow\mathcal{A}) \quad \Leftrightarrow \quad (\exists x)\,[\, x \in A \ \wedge\ z \in (\downarrow\mathcal{S}_x)\,] \tag{31}$$

Recordemos que uma n-cobertura α de $\mathcal{A}$ é um n-evento que cobre $\mathcal{A}$; isto é, tem a forma $\bigcup \uparrow z$, com $|z| \geq n$.

69 Noção
A **n-cobertura mínima** *de* $\mathcal{A}$*, representado por* $\mathcal{A}^{(n)}$ *, é o* n*-evento gerada pelos prefixos de* $\mathcal{A}$

$$\mathcal{A}^{(n)} = \bigcup_{|z|\geq n \wedge z \in (\downarrow \mathcal{A})} \uparrow z \tag{32}$$

É elementar verificar que sequência de eventos $\left\{\mathcal{A}^{(n)}\right\}$ é decrescente e todos os elementos da sequência contêm $\mathcal{A}$; isto é, para todo n, $\mathcal{A} \subseteq \mathcal{A}^{(n+1)} \subseteq \mathcal{A}^{(n)}$. Por isso, faz sentido

70 Noção
A **cobertura mínima**[14] *de* $\mathcal{A}$ *é o evento* $\hat{\mathcal{A}} = \bigcap_n \mathcal{A}^{(n)}$*. Fixando uma medida de Lebegue* μ*, o real* $\mu(\hat{\mathcal{A}})$ *representa-se por* $H^{\mu}(\mathcal{A})$ *e designa-se por* μ**-medida exterior de Hausdorff** *de* $\mathcal{A}$*.*

71 Facto
Cada evento $\mathcal{A}^{(n)}$ *é recursivamente enumerável. No entanto* $\hat{\mathcal{A}}$ *pode não ser recursivamente enumerável.*

Note-se que

$$H^{\mu}(\mathcal{A}) = \mu(\hat{\mathcal{A}}) = \lim_{n\to+\infty} \mu(\mathcal{A}^{(n)}) = \lim_{n\to+\infty} \sum_{|z|\geq n \wedge z \in (\downarrow \mathcal{A})} \mu(\uparrow z) \tag{33}$$

[14] Também designada por **cobertura de Hausdorff**.

É importante ter em atenção que H^{μ} não é, normalmente, uma medida. É uma avaliação do "tamanho" de uma classe através da medida dos eventos que a cobrem.

□

Domínios algébricos e co-algébricos

Os domínios enumeráveis são, frequentemente, **algébricos** (também designados por **indutivos**) no sentido em que os elementos do domínio podem ser construídos indutivamente a partir de outros elementos do mesmo domínio.

Exemplo 5:
Pode-se construir os elementos de $\mathbb{N}$ pela construção de Peano, partindo da constante 0 e da função "sucessor" $\mathrm{suc}\colon x \mapsto x+1$. Por indução são gerados, com estes dois construtores, todos os elementos de $\mathbb{N}$.

Do mesmo modo as bit-strings finitas são geradas por três construtores: a constante string vazia ε e duas funções $\mathrm{inc1}\colon u \mapsto u\,1$ e $\mathrm{inc0}\colon u \mapsto u\,0$ (a primeira acrescenta o bit 1 e a segunda acrescenta o bit 0 ao argumento).

Domínios indutivos incluem as listas X^* de elementos de um domínio enumerável X, e as árvores com nodos em X. Domínios enumeráveis formados por conjuntos de X, ou "bags" já não são, normalmente, algébricos.

É fácil verificar que todo o domínio indutivo é primitivamente representável.

Enquanto que $\mathbb{B}^*$ e $\mathbb{N}$ são paradigmas de domínios enumeráveis, os domínios dos seus sub-conjuntos (respectivamente $\wp(\mathbb{B}^*)$ e $\wp(\mathbb{N})$) são paradimas de domínios não-enumeráveis.

EXEMPLO 6:
A prova de que $\wp(\mathbb{N})$ não é enumerável é um exemplo típico de prova por redução ao absurdo através de um argumento de diagonalização: assume-se que existe uma enumeração por conjuntos $\{A_k\}$ que cobre todo $\wp(\mathbb{N})$; define-se um conjunto "diagonal" A que contém o inteiro k se e só se o conjunto A_k não contém k; claramente a "diagonal" A não pode pertencer à enumeração e, portanto, esta não pode cobrir $\wp(\mathbb{N})$.

Os domínios não enumeráveis (os anteriores e ainda as bit-strings infinitas $\mathbb{B}^\infty$, o intervalo real $[0,1]$, os domínios de funções $\mathbb{B}^\mathbb{N}$ e $\mathbb{N}^\mathbb{N}$, etc.) são, quanto muito, **co-algébricos** (ou **co-indutivos**) no sentido em que os elementos do domínio podem ser indutivamente observados; isto é, indutivamente o resultado de uma observação é construído a partir de resultados de observações análogas. No entanto não existe uma função computável única capaz de construir um elemento do domínio a partir de outros elementos do mesmo domínio ou de outros domínios.

O seguinte exemplo procura ilustrar estes pontos.

EXEMPLO 7:
Considere-se a "função" $r : \omega \mapsto \omega^\sim$, "definida" em (28) que relaciona bit-strings infinitas com reais.

Não se trata de uma função computável já que constrói apenas os elementos de uma sequência de Cauchy. É um "processo", e não um "algoritmo", que vai construindo sequencialmente os racionais r_k que convergem para o "resultado". Sequencialmente, no passo k, o processo lê o bit de ω_k e calcula o racional $r_k = r_{k-1} + \omega_k\, 2^{-k-1}$. Nunca se chega a atingir o "resultado" limite; a sequência de racionais $\{r_k\}$ é o verdadeiro "output" deste processo.

Suponhamos que se quer construir uma "função inversa" que tome um número real no intervalo $\mathbb{I} = [0,1]$ e gere a bit-strings que lhe corresponde. Mais uma vez não existe nenhuma função computável mas um eventual processo. Considere-se a função $h : \mathbb{I} \longrightarrow \mathbb{B} \times \mathbb{I}$

definida do seguinte modo

$$h(x) \quad = \quad \begin{cases} (0\ ,\ 2\,x) & \text{se} \quad x < 1/2 \\ (1\ ,\ 2\,x-1) & \text{se} \quad x \geq 1/2 \end{cases} \tag{34}$$

Esta função gera, a partir de um valor inicial x, uma sequência de bits $b_0, b_1, b_2, \cdots$ através da função de recorrência

$$x_0 = x \qquad , \qquad (b_i\ ,\ x_{i+1}) \ = \ f(x_i)$$

De facto não existe um processo computável de gerar esses bits uma vez que a função h não é computável. De facto não existe sequer uma forma computável para determinar se um real arbitrário é ou não inferior a $1/2$.

Como os reais não são enumeráveis, não é possível, genericamente, definir funções computáveis que construam reais. No entanto é possível aproximar um determinado real por uma sequência de racionais que "cresça" para um determinado limite [15].

Resta saber se os elementos dessa sequência podem ser computados. Surge assim a noção de "real recursivo" que essencialmente traduz o facto de a representação $r(\cdot)$ ser uma função recursiva que, para determinar um real $x \geq 0$, calcula uma sequência crescente de aproximações racionais $\{r(n)\}$ convergindo para x; i.e. $x = \lim \uparrow r(n)$.

A noção de real r.L. é, no entanto, mais geral e pode assumir várias facetas identificando objectos "computacionais" de natureza diversa: linguagens, testes, etc. Caso não exista ambiguidade usaremos a mesma designação (real

[15] Aliás esta é a definição formal de real.

recursivo de Lebesgue) para designar a linguagem L da definição, a função recursiva r que a enumera e o real $\lim \uparrow r(n)\tilde{}$.

1.5 Aleatoriedade

A aleatoriedade é um dos conceitos fundamentais em Criptografia uma vez que a noção de "segredo" está, intuitivamente, ligada à incapacidade de um atacante distinguir informação protegida de não-informação; isto é, informação aleatória.

Aleatoriedade de Kolmogorov

A primeira abordagem à aleatoriedade foi apresentada através do "índice de Kolmogorov". Recordemos a noção de índice de Kolmogorov (noção 23, página 29) de um inteiro y como o inteiro $\kappa(y)$ tal que

$$\varphi_{\kappa(y)}(0) \simeq y \quad , \quad \varphi_e(0) \not\simeq y \quad \text{par todo} \quad e < \kappa(y)$$

Vimos que κ era uma função total mas não recursiva.

Pode-se estender esta noção usando funções parciais que recorrem a um oráculo O e definir $\kappa(y|O)$ como o inteiro que verifica

$$\varphi^O_{\kappa(y)}(0) \simeq y \quad , \quad \varphi^O_e(0) \not\simeq y \quad \text{par todo} \quad e < \kappa(y)$$

Como sabemos φ^O_e é a o elemento de ordem e na indexação das funções parciais recursivas que recorrem à relação O como função primitiva.

O inteiro $y \in \varpi$ é **κ-compressível** quando $\kappa(y) < y$. Os inteiros que não são κ-compressíveis são **κ-aleatórios**.

Estas noções extendem-se para strings finitas, definindo $\kappa(r(y)) = \kappa(y)$ ($r : \mathbb{N} \longrightarrow \mathbb{B}^*$ é a representação standard de strings finitas), e para conjuntos finitos, definindo $\kappa(D_y) \doteq \kappa(y)$.

O teorema de Kolmogorov (teorema 47, página 46) permite dar uma primeira caracterização dos objectos (inteiros, ou strings finitas, ou conjuntos finitos) κ-compressíveis e, por complemento, os objectos κ-aleatórios. O teorema diz-nos os inteiros κ-compressíveis formam um conjunto simples; isto é, um conjunto recursivamente enumerável cujo complemento é imune (infinito e nenhum dos seus sub-conjuntos infinitos é recursivamente enumerável).

□

Note-se que todos estes resultados são independentes do sistema de índice usado mas o índice de Kolmogorov vai depender, em princípio do sistema de índices usado. Por isso, para definir uma noção de aleatoriedade que seja universal, convém restringir o sistema de índices usado.

A enumeração das funções parciais recursivas tem uma variante quando se

72 Teorema
Existe um sistema de índices $\{\varphi_e\}$ que enumera todas as funções parciais recursivas com domínio livre de prefixos e uma função universal φ com domínio livre de prefixos que verifica $\varphi_e(x) \simeq \varphi(e\|x)$.

Esboço de prova A prova deste teorema é uma consequência da normalização das funções parciais recursivas (teorema 15, página 23) e da representação dos sistemas aceitáveis de índeices (teorema 20, página 26).

Neste resultado estamos a ver φ como uma função parcial recursiva de strings finitas em strings finitas; a condição de o seu domínio ser livre de prefixos pode-se escrever

$$\varphi(x)\downarrow \wedge\ \varphi(y)\downarrow \wedge\ x \neq y \quad \Rightarrow \quad \uparrow x \cap \uparrow y = \emptyset \tag{35}$$

As funções φ que vericam as condições do resultado anterior, dizem-se **universais livres de prefixos (u.l.p.)**; as indexações que determinam dizem-se **livres de prefixos**.

Por o domínio de φ ser livre de prefixos, tem-se $\sum_{\varphi(z)\downarrow} 2^{-|z|} \leq 1$ (ver facto 54, página 53). Esta observação tem uma forma "inverso"

73 FACTO
Dada uma qualquer sequência recursiva de $\{d_n\}_n$ que verifique $d = \sum_n 2^{-d_n} \leq 1$, existe uma função parcial recursiva φ_e cujo domínio é uma linguagem livre de prefixos com medida d.

Prova Seja $\{x_n\}$ a sequência recursiva livre de prefixos gerada pela sequência $\{d_n\}$ (teorema 59, página 58). A função $\varphi_e(z) = \mu n \cdot [z = x_n]$ é parcial recursiva e o seu domínio é a linguagem livre de prefixos enumerada por $\{x_n\}$. Como $|x_n| = d_n$, para todo n, tem-se $\sum_n 2^{-|x_n|} = \sum_n 2^{-d_n} = d$.

□

74 NOÇÃO (COMPLEXIDADE DE KOLMOGOROV)
*Seja φ uma função universal livre de prefixos. Define-se φ-**complexidade de Kolmogorov**[16] de um $y \in \varpi$,*

[16] Na generalidade dos textos esta comprexidade é normalmente designada "prefix free Kolmogorov complexity". No entanto, devido

representado por $K_\varphi(y)$, *como* $\min\{\,|x| \mid \varphi(x) \simeq y\,\}$. *Dado um conjunto* O *(um oráculo) define-se* $K_\varphi(y|O) = \min\{\,|x| \mid \varphi^O(x) \simeq y\,\}$.

Algumas propriedades de K

75 Teorema

1. *A menos de uma constante* $O(1)$ *independente de* y, *tem-se* $K_\varphi(y) = \min\{\,|e| \mid \varphi_e(0) \simeq y\,\} + O(1)$.
2. *Para qualquer linguagem* L *verifica-se* $\sum_{y\in L} 2^{-K_\varphi(y)} \leq 1$.

Esboço de prova O resultado (1) deriva da indexação de funções. Este é um resultado que estabelece a relação entre a complexidade de Kolmogorov e o índice de Kolmogorova para indexações livres de prefixos.

O resultado (2) deriva do facto de que o conjunto $\{\,x \mid \varphi(x) \simeq y \wedge y \in L\,\}$ ser livre de prefixos já que o domínio de φ é livre de prefixos (ver facto 54, página 53).

Por outro lado temos uma consequência imediata do teorema de Kolmogorov particularizando a indexação para uma que seja livre de prefixos.

76 Teorema

Para toda a constante $c \geq 0$, *o conjunto* $\{\,y \mid K_\varphi(y) \leq |y| - c\,\}$ *é simples.*

ao modo como é aqui definida a indexação das funções parciais recursivas, esta complexidade é sempre "prefix free" e não tem qualquer interesse introduzir outra que o não seja.

Esboço de prova
O conjunto $\{ y \mid K_\varphi(y) \leq |y| - c \}$ é um sub-conjunto do conjunto das objectos compressíveis que, pelo teorema de Kolmogorov, é simples. É fácil verificar que o sub-conjunto também tem de ser simples.

Por outro lado, usando as relações entre os sistemas aceitáveis de índices (ver teorema 20, página 26) prova-se

77 TEOREMA
A menos de uma constante, $K_\varphi(y)$ (ou $K_\varphi(y|O)$) não depende da função universal livre de prefixos φ.

Nestas circunstâncias, entendendo que $K(y)$ e $K(y|O)$ são sempre definidos a menos de uma constante independente de y, podemos prescindir do índice φ em K_φ.

Intuição
A indexação das funções parciais recursivas usa intensivamente a construção de pares $(x\|y)$. Uma questão importante é o de saber quais são os limites ao tamanho do par $x\|y$ em função dos tamanhos de x e y.

Intuitivamente sabemos que, para ser possível recuperar x e y a partir de $x\|y$, tem de existir neste valor informação sobre o tamanho do primeiro elemento do par. A codificação óptima de $|x|$ exige alguma informação constante C (que depende apenas do algoritmo de emparelhamento) e $||x||$ bits. Por isso,

78 FACTO
Existe uma codificação óptima de pares x, y tal que

$$|x\|y| \leq C + ||x|| + |x| + |y|$$

Considere-se agora o índice e da projecção π_1. Temos $\varphi(e\|(y\|0)) \simeq \pi_1(y\|0) \simeq y$. Pela propriedade de parametrização dos índices, existe uma função recursiva λ tal que $\varphi(\lambda(y\|e)\|0) \simeq \varphi(e\|(y\|0)) \simeq y$. Por isso $\kappa(y) \leq \lambda(y\|e)$.

Basta agora ver que $|\lambda(z)| \leq |z| + O(1)$ para concluirmos que $|\kappa(y)| \leq C + |y\|e| \leq C + ||y|| + |y| + |e|$. Logo $|\kappa(y)| \leq O(1) + ||y|| + |y|$, absorvendo na constante $O(1)$ quer a constante C como $|e|$.

Baseado neste tipo de intuição, e nas propriedades de enumeração e parametrização dos índices, prova-se que

79 FACTO

A complexidade de Kolmogorov K é "sub-aditiva"; isto é,

$$K(x\|y) \;\leq\; O(1) + K(x) + K(y)$$

Para exprimir propriedades genéricas da complexidade de Kolmogorov é conveniente alguma notação prévia:

80 NOÇÃO

Vamos designar por **défice de informação** *de y a quantidade*

$$k_d(y) \quad = \quad K(y) - |y| - K(|y|) \tag{36}$$

e por **défice inicial de informação** *a quantidade*

$$K_d(y) \quad = \quad K(y) - |y| \;=\; k_d(y) + K(|y|) \tag{37}$$

A informação própria de y (aquela que a mensagem y transmite) está nos seus bits e no seu tamanho. Por isso entendemos que ela é $|y| + K(|y|)$.

Desta forma $k_d(y)$ representa um "défice" de informação: a diferença entre a informação $K(y)$ no "input", indispensável para que uma máquina de Turing produza y, e a informação própria do "ouput" y. A quantidade K_d exprime o mesmo tipo de défice mas tomando como referência a informação nos bits de y.

Ao longo deste curso vai ser muito importante caracterizar a **incerteza** com que podemos, dado um conjunto de possibilidades, seleccionar uma dessas possibilidades. A próxima secção será dedicada ao estudo desta noção mas, para já, é necessário avançar com algumas noções prévias.

81 NOÇÃO (INCERTEZA DE HARTLEY E UNIFORME)
Designamos por **incerteza de Hartley** *(ou, caso não exista ambiguidade, simplesmente* **incerteza**[17]*) de um conjunto finito A, e representamos por $\vartheta(A)$, e definida por $\vartheta(A) = \lceil \log_2 [\![A]\!] \rceil$, sendo $[\![\cdot]\!]$ a função recursiva que conta o número de elementos de um conjunto finito. Define-se $\vartheta(\emptyset)$ como $+\infty$.*

A **incerteza uniforme** *à funcional $\vartheta[\cdot]$ que a cada conjunto A (finito ou não) associa a função $\vartheta[A](n) \simeq \vartheta(A \restriction n)$.*

Propriedades genéricas da complexidade de Kolmogorov exprimem a sua relação com a noção de incerteza uniforme.

[17] A próxima secção, dedicada ao conceito de incerteza, vai introduzir outras formas para exprimir esta noção.

82 Teorema (Contagem de Chaitin)
Para todo n, todo k, e para uma constante $O(1)$ independente de n e k,

$$\max_{|y|=n} k_d(y) = \pm O(1) \tag{38}$$

$$n - \vartheta[\{\, y \mid k_d(y) \leq -k \,\}](n) \geq k - O(1) \tag{39}$$

O teorema da contagem presupõe que se restringe os y´s a objectos de tamanho n e diz-nos duas coisas:

(i) Para todo n, o máximo do déficite $k_d(y)$ dos y´s de tamanho n, é uma constante C independente de n.

(ii) O predicado $P(y) = k_d(y) \leq -k$ avalia se o défice de informação de y é inferior a $-k$ bits; isto é equivalente a dizer que o "supro informação" é igual ou superior a k bits.
Restringindo os valores y´s ao comprimento n, o conjunto de possibilidades tem incerteza n. Assim a diferença $n - \vartheta_n[y.k_d(y) \leq -k])$ mede a quantidade de informação que é preciso fornecer para ter a certeza que a propriedade $[k_d(y) \leq -k]$ se verifica nesse conjunto de possibilidades. Essencialmente (ii) diz-nos que, a menos de uma constante positiva, é necessário fornecer (de novo) os mesmos k bits de informação.

□

A complexidade de Kolmogorov $K(y)$ é uma medida que pode ser usada em conjuntos finitos mas não em conjuntos infinitos. Se A é um conjunto infinito, temos de atender à complexidade $K(A \restriction n)$ das diferentes truncaturas de A.

Esta ideia e o teorema da contagem está na base da seguinte definição:

83 Noção (Aleatoriedade de Kolmogorov-Levin-Chaitin)
Um conjunto A é ***KLC-aleatório*** *se existe um $c \in \varpi$ tal que, para todo n, $K(A\restriction n) > n - c$. O conjunto A é* ***KLC-aleatório relativo a O*** *se, para algum c, se verifica $K(A\restriction n|O) > n - c$ para todo n.*

Em alternativa pode-se apresentar a definição de aleatoriedade, introduzindo a família de conjuntos U_c definidos como

$$U_c \quad = \quad \{\, y \mid K(y) \leq |y| - c \,\} \tag{40}$$

Note-se que, pelo teorema 76, todos estes conjuntos são simples. Então a noção 83 reescreve-se facilmente.

84 Facto
A é KLC-aleatório sse existe uma constante $c \in \varpi$ tal que, para todo n, $A\restriction n \notin U_c$. Equivalentemente, A não é KLC-aleatório sse está contido em $\bigcap_c \uparrow U_c$.

Prova
A prova da primeira parte é uma simples reescrita da definição 83 e da definição dos conjuntos U_c. Para a segunda parte, temos que A não é KLC-aleatório sse, para todo c, existe n tal que $(\exists n)[K(A\restriction n) \leq n - c]$; ou seja, $(\exists n)[A\restriction n \in U_c]$ ou ainda, equivalentemente, $A \in \uparrow U_c$. Como A tem de pertencer a todos $\uparrow U_c$ concluímos que tem de pertencer à sua intersecção.

Os conjuntos U_c determinam eventos $\uparrow U_c$ que são "raros". Esse facto deriva do seguinte resultado.

85 Facto

Seja U_c definido em (40)*; então* $\sum_{y\in U_c} 2^{-|y|} \leq 2^{-c}$.

Esboço de prova

Para $y \in U_c$ tem-se $c - |y| \leq -K(y)$; logo $2^{c-|y|} \leq 2^{-K(y)} \leq 1$. Consequentemente $\sum_{y\in U_c} 2^{-|y|} \leq 2^{-c}$.

Combinando o teorema da contagem com a esta definição de aleatoriedade, para um conjunto A ser KLC-aleatório tem existir constante $O(1)$ tal que, para todo n,

$$0 \leq K(A\restriction n) - n \pm O(1) \leq K(n)$$

Note-se que só se define complexidade de Kolmogorov a menos de uma constante; por isso, para medir a informação necessária para computar o prefixo $A\restriction n$, temos de usar $K(A\restriction n) \pm O(1)$.

A quantidade $K(A\restriction n) - n \pm O(1)$ mede, a menos de uma constante, o "défice" inicial de informação que é preciso colocar na computação relativa à informação nos bits do prefixo. Este défice é sempre positivo e está limitado superiormente pela complexidade do comprimento n.

Essa é a intuição associada à complexidade de Kolmogorov: para gerar qualquer prefixo dum conjunto aleatório é sempre preciso por mais informação do que a que se consegue retirar. No entanto aquilo que é importante, em termos de aleatoriedade, é a "velocidade" com que esta défice cresce com n.

Esta relação entre a noção de KLC-aleatoriedade razão de crescimento de $K(A\restriction n) - n$, exprime-se em termos das chamadas funções ϖ-similares.

86 Noção
Uma função total f é ϖ-similar quando verifica $\sum_n 2^{-f(n)} < \infty$. Em caso contrário diz-se ϖ-limitada.

Note-se que adicionar ou subtrair uma constante a f não altera o facto de f ser ϖ-similar ou não. Adicionalmente é simples provar que, se f é ϖ-similar e g é ϖ-limitada, então $(\exists_\infty n)\,[f(n) > g(n)]$.

87 Teorema (Miller & Yu)
Se A é um conjunto KLC-aleatório, então a função de n, $K(A\restriction n) - |A\restriction n|$ é ϖ-similar.

Como complemento a este teorema temos algo que nos ajuda a responder a uma dúvida básica: *será que existem conjuntos aleatórios?* O seguinte teorema dá uma resposta a esta questão.

88 Teorema (Miller)
Para todo ϖ-similar f existe um conjunto KLC-aleatório A tal que, para todo n, $K(A\restriction n) - n \leq f(n) + O(1)$.

Uma outra propriedade importante da complexidade de Kolmogorov deriva da existência dos chamados **conjuntos de Kraft-Chaitin** que, de uma forma que iremos esclarecer, "aproximam" conjuntos aleatórios.

89 NOÇÃO (CONJUNTO DE KRAFT-CHAITIN)
Uma sequência recursiva de pares $R = \{d_n \| y_n\}$ *que verifiquem* $\sum_n 2^{-d_n} \leq 1$ *designa-se por* ***conjunto de Kraft-Chaitin***.

Os pares $r_n = d_n \| y_n$ são designados por **requests**; o elemento d_n designa-se por **incerteza** de y_n e o elemento y_n por **resposta** a $r(n)$.

Pode-se ver $R = \{d_n \| y_n\}_n$ como uma enumeração de respostas à selecção de um n condicionado por incertezas d_n. É como uma função recursiva $n \mapsto y_n$ mas associando incertezas d_n à selecção do argumento e com a restrição adicional de os pares incerteza+resposta serem recursivamente enumeráveis. Desta forma os $w_n = 2^{-d_n}$ são uma **probabilidade** de um acto de selecção do índice/argumento n.

EXEMPLO 8: Alguns exemplos particularmente simples de conjuntos de Kraft-Chaitin

1. O domínio abstract ϖ determina um conjunto de Krfat-Chaitin $\varpi' = \{d_n \| n\}_{n \in \varpi}$ com $d_n = n + k$, para um qualquer $k > 0$ independente de n. De facto $\sum_{y \in \varpi} 2^{-d_n} = 2^{-k} \sum_{n \in \varpi} 2^{-n} = 2^{-k+1} \leq 1$.

2. Seja A um conjunto recursivo; então o conjunto dos seus prefixos $\downarrow A = \{A \restriction n\}_{n \in \varpi}$ é também recursivo. A sequência $n \mapsto d_n = n + k$ é recursiva e a sequência de pares $A' = \{d_n \| A \restriction n\}_{n \in \varpi}$ é recursiva. Como (tal como no caso anterior), $\sum_n 2^{-d_n} \leq 1$, conclui-se que A' é um KC-conjunto.

3. Genericamente seja Y um qualquer conjunto recursivamente enumerável e seja $\alpha : \varpi \longrightarrow$ uma enumeração recursiva dos seus elementos. Suponhamos que é possível encontrar uma função recursiva d tal que $K(\alpha(n)) \leq d(n)$, para todo n. Então $R = \{d(n)\|\alpha(n)\}_n$ é um conjunto de Kraft-Chaitin.
   De facto, porque α e d são ambas recursivas, o conjunto $\{d(n)\|\alpha(n)\}_{n\in\varpi}$ é recursivamente enumerável. Por outro lado, usando o teorema 75, $\sum_n 2^{-d(n)} \leq \sum_n 2^{-K(\alpha(n))} \leq \sum_{y\in R} 2^{-K(y)} \leq 1$.

4. Em particular, considere-se, para $c \in \varpi$, o conjunto $U_c = \{\, y \mid K(y) \leq |y| - c \,\}$. Vimos (teorema 76) que U_c é um conjunto recursivamente enumerável. Considere-se a função recursiva $\tau(y) = |y| - c$; seja $d(n) = \tau(\alpha(n))$ sendo α uma enumeração de U_c. Como, para todo $y \in U_c$, se verifica $K(y) \leq \tau(y)$ o exemplo anterior diz-nos que $\{\tau(y)\|y\}_{y\in U_c}$ é um conjunto de Kraft-Chaitin.

O exemplo 8 diz-nos que a existência de uma função recursiva d e um conjunto recursivamente enumerável Y tal que $K(y) \leq d(y)$, para todo $y \in Y$, origina um conjunto de Kraft-Chaintin da forma $\{d(y)\|y\}_{y\in Y}$. O resultado inverso é um teorema importante.

90 TEOREMA (LEVIN, SCHNORR, CHAITIN)
Se $R = \{d_n\|y_n\}$ for um conjunto de Kraft-Chaitin, então existe uma sequência recursiva, livre de prefixos $\{x_n\}_n$ e uma função parcial recursiva ϕ tais que, para todo n, $|x_n| = d_n$ e $\phi(x_n) \simeq y_n$.

Como consequência, para todo n e para uma constante $O(1)$ independente de n, verifica-se $K(y_n) \leq d_n + O(1)$.

Essencialmente o resultado diz-nos que, para cada "request" $r_n = d_n \| y_n$ existe um programa e_n com o comprimento da incerteza d_n, que é capaz de gerar a resposta y_n; ou seja, $\varphi(e_n \| 0) \simeq y_n$ e $|e_n| \geq d(n)$. Note-se que a construção de Kaft-Chaitin impõe que o conjunto de pares $\{d_n \| y_n\}_n$ seja computacionalmente enumerável o que é algo que não precisa de acontecer, por exemplo, no conjunto $\{K(y_n) \| y_n\}_n$.

Esboço de prova

Pelo teorema 59 (página 58), existe uma sequência recursiva, livre de prefixos $\{x_n\}$ tal que $|x_n| = d_n$; portanto a função $h(x) \simeq \mu n \cdot [x = x_n]$ é parcial recursiva. Como $\{y_n\}$ é uma sequência recursiva, a função $\phi(x) = y_{h(x)}$ é parcial recursiva e verifica $\phi(x_n) \simeq y_n$, para todo n.

Seja φ uma qualquer função universal cujo domínio é livre de prefixos. Seja e o índice de ϕ em φ. Então $y_n \simeq \phi(x_n) = \varphi(e \| x_n)$; logo $K_\varphi(y_n) \leq |e| + |x_n| = |e| + d_n$.

91 Noção

Um conjunto de Kraft-Chaitin $R = \{d_n \| y_n\}$ diz-se **suficientemente aleatório** *quando $\sum_n 2^{-\sigma_n} < \infty$ sendo $\{\sigma_n\}$ a sequência definida por $\sigma_n = K(y_n) + d_n - |y_n|$.*

Um conjunto-KC R suficientemente aleatório, distribui a aleatoriedade por duas componentes: nas incertezas d_n e nas respostas y_n. A incerteza na geração de y_n é dada por $K(y_n)$; a incerteza na "escolha" de y_n é representada por d_n; somando as duas incertezas e subtraindo a informação específica de y_n (dada por $|y_n|$) temos a diferença σ_n. A "aleatoriedade suficiente" impõe que a sequência $\{\sigma_n\}$ seja ϖ-similar.

Esta noção é sugerida pelo teorema de Miller & Yu (teorema 87, na página 83) onde se toma um conjunto A e a sequência das suas truncaturas, $y_n = A \upharpoonright n$; fazendo $\sigma_n = K(y_n) - |y_n|$ e sendo A aleatório, o teorema diz-nos que $\{\sigma_n\}$ é uma sequência ϖ-similar.

Note-se que, neste caso, a sequência $\{y_n\}$ não é recursiva e portanto não se trata, formalmente, de um conjunto de Kraft-Chaitin. Por outro lado não são necessárias incertezas d_n: a aleatoriedade de A é já suficiente para garantir que $\{\sigma_n\}$ seja ϖ-similar. Se A não for aleatório as incertezas d_n que tornam $\{\sigma_n\}$ ϖ-similar medem a "falta de aleatoriedade" de A.

Essa abordagem é ilustrada nos exemplos seguintes.

EXEMPLO 9: Considere-se os conjuntos de Kraft-Chaitin apresentados no exemplo 8 e procuremos averiguar quais os que são suficientemente aleatórios:

1. Considere-se o KC-conjunto $\mathcal{N} = \{n + k \| n\}_n$ com $k > 0$. Então $\sigma_n = K(n) + (n+k) - |n| = K(n) + (n+k) - \lfloor \log_2(n+1) \rfloor \geq K(n)$, para todo $n \in \varpi$. Consequentemente $\sum_n 2^{-\sigma_n} \leq \sum_n 2^{-K(n)} < \infty$ e, portanto, $\mathcal{N}$ é suficientemente aleatório.
   É possível definir incertezas d_n menores que $n + k$ e que ainda geram conjuntos $\mathcal{N}$ suficientemente aleatórios. Por exemplo, sabe-se que, para $s > 1$, a série $\zeta(s) = \sum_{n>0} n^{-s}$ converge. De facto, quando s percorre os complexos esta é a famosa zeta-função de Riemman que converge[18] para todos $\Re(s) > 1$. Defina-se uma

[18] Um dos resultados mais supreendentes da zeta-função de Riemann é **fórmula de produtos de Euler** que diz que, para $\Re(s) > 1$,

variante da função comprimento da forma seguinte: $|n|_s = \lfloor s \log_2(n+1) \rfloor$. Então, para um real $s > 1$, a série $\sum_n 2^{-|n|_s}$ converge; de facto tem-se $2^{-|n|_s} \leq (n+1)^{-s}$ e, portanto, $\sum_n 2^{-|n|_s} \leq \zeta(s)$.
Se $s > 1$ for um real recursivo, a função $d_s(n) = |n|_s + k$ é recursiva e é possível escolher $k > 0$ de tal forma que se verifique $\sum_n 2^{-d_s(n)} \leq 1$: basta escolher $k \geq \log_2(\zeta(s))$.
Designemos por $\mathcal{N}_s$ o conjunto de Kraft-Chaitin $\{d_s(n)\|n\}_{n\in\varpi}$ com d_s assim definida.
Com tais incertezas tem-se $\sigma_n = K(n) + d_s(n) - |n| = K(n) + k + |n|_s - |n|$. Para $s > 1$, tem-se sempre $|n|_s - |n| \geq 0$; logo $\sigma_n \geq K(n)$ para todo n e, consequentemente, $\sum_n 2^{-\sigma_n} < \infty$. Com estas incertezas $\mathcal{N}_s$ continua a ser sufientemente aleatório.

2. Seja A um conjunto recursivo e $A'_k = \{n+k \parallel A \restriction n\}_n$; no exemplo 8 viu-se que, para $k > 0$, A'_k é um conjunto de Kraft-Chaitin. Temos $\sigma_n = K(A \restriction n) + (n+k) - n = K(A \restriction n) + k$; logo $\sum_n 2^{-\sigma_n} = 2^{-k} \sum_n 2^{-K(A\restriction n)} < \infty$. Consequentemente A'_k é suficientemente aleatório.

3. Seja $Y = \{y_n\}$ um conjunto recursivamente enumerável; vimos que a existência de uma função recursiva d tal que, para todo n, $K(y_n) \leq d(n) + k$ assegura, com constantes k apropriadas, que $Y' = \{d_n \| y_n\}$ é um

verifica-se $\zeta(s) = \prod_p \frac{1}{1-p^{-s}}$ em que o poduto percorre todos os *números primos* p e converge uniformemente numa vizinhança de s.

Sabe-se que $\zeta(s)$ tem uma continuação meromórfica para todo o plano complexo com um único polo no ponto $s = 1$. Sabe-se que $\zeta(s)$ não tem nenhum zero para $\Re(s) > 1$ e os zeros com $\Re(s) < 0$ são os inteiros pares negativos. A famosa **hipótese de Riemman** conjectura que todos os restantes zeros de $\zeta(s)$ se situam sobre a recta $\Re(s) = 1/2$.

conjunto de Kraft-Chaitin. Isto não assegura, porém, que Y' seja suficientemente aleatório. Se adicionalmente se tiver $|y_n| \leq d_n + O(1)$, então $\sigma_n \geq K(y_n) + O(1)$ e, por isso, Y' já será suficientemente aleatório.

4. O conjunto $U_c = \{ y \mid K(y) \leq |y| - c \}$ é recursivamente enumerado; seja α uma qualquer enumeração recursiva de U_c. Vimos que $U'_c = \{|\alpha(n)| - c \| \alpha(n)\}_n$ é um conjunto de Kraft-Chaitin.

   Adicionalmente temos $\sigma_n = K(\alpha(n)) + (|\alpha(n)| - c) - |\alpha(n)| = K(\alpha(n)) - c$; portanto $\sum_n 2^{-\sigma(n)} = 2^c \sum_{y \in U_c} 2^{-K(y)} < \infty$. Consequentemente U'_c é suficientemente aleatório.

O seguinte resultado ilustra a intuição por detrás da noção de KC-conjunto suficientemente aleatório.

92 FACTO

Seja A um conjunto. Se existir d tal que $R = \{d(n) \| A \upharpoonright n\}_{n \in \varpi}$ é um KC-conjunto suficientemente aleatório, então existe um conjunto KLC-aleatório A' tal que $K(A' \upharpoonright n) \leq K(A \upharpoonright n) + d(n) + O(1)$, para todo n.

Prova Seja $\sigma(n) = K(A \upharpoonright n) + d(n) - n$; por hipótese R é suficientemente aleatório e, por isso, σ é ϖ-similar. Então, pelo teorema de Miller (teorema 88, pag. 83), existe um conjunto KLC-aleatório A' tal que $K(A' \upharpoonright n) \leq n + \sigma(n) + O(1)$; ou seja $K(A' \upharpoonright n) \leq K(A \upharpoonright n) + d(n) + O(1)$

Este resultado ilustra como um conjunto A não aleatório (nomeadamente, é recursivo) pode "aleatorizar-se" à custa de uma sequência de incertezas $d(n)$ que façam suficientemente aleatório o conjunto de Kraft-Chaitin definido pelas truncaturas de A; é possível "randomizar" um conjunto A, mesmo que as complexidades $K(A \upharpoonright n)$ sejam baixas,

juntando-lhes valores $d(n)$. A intuição é os $d(n)$ representam a incerteza adicional que falta a A para ser aleatório; de certa forma d mede a "falta de aleatoriedade" de A.

O que este resultado não diz é qual é a relação entre os elementos do conjunto "pouco aleatório" A e os do conjunto aleatório A'.

Aleatoriedade de Martin-Löf

A aleatoriedade de Martin-Löf baseia-se no conceito de "teste raro". Intuitivamente um teste raro sobre conjuntos é uma propriedade que pode ser verificada de forma computacionalmente eficiente sobre qualquer conjunto A, que falha quase sempre mas não é impossível; isto é, são "raros" os conjuntos que verificam a propriedade mas existem sempre alguns.

Assim, dado um conjunto A, se for possível determinar um teste raro que A verifique, então é porque se conhece informação sobre A a partir da qual é possível fazer essa selecção de teste; nestas circunstâncias, o conjunto não será aleatório. Se não for possível escolher um tal teste, é porque não é possível encontrar padrões de regularidade em A; ito é, A é aleatório.

Estas intuições formalizam-se na chamada **complexidade de Martin-Löf** e, para a formalizar, começamos por clarificar a nossa noção de "teste computável raro".

□

Começamos por analisar a noção genérica de **teste**.

Classes efectivamente enumeráveis de conjuntos representam "continuações computávies" de objectos previamente computados. Estas continuações são aproximações computáveis de objectos que, normalmente, não podem ser computados. Por isso faz sentido definir sequências de classes efectivamente enumeráveis representando sequências de aproximações sucessivas a objectos que podem ser descritos mas não computados.

93 NOÇÃO

Um ***teste*** $\mathcal{T} = \{\mathcal{A}_n\}$ *é uma sequência decrescente (*$\mathcal{A}_{n+1} \subseteq \mathcal{A}_n$*, para todo* n*) de classes efectivamente enumeráveis de conjuntos.*

Um teste $\{\mathcal{A}_n\}$ *é* ***uniformemente recursivo*** *se existe uma sequência uniforme de funcionais efectivas* $\{F_n\}$ *de tal forma que cada uma das classes* $\mathcal{A}_n$ *verifica* $\mathcal{W}_x \in \mathcal{A}_n \Leftrightarrow \varphi_x \in F_n\tilde{}$.

Uma classe $U \subseteq 2^{\varpi}$ ***satisfaz*** *o teste* $\mathcal{T}$ *quando* $U \subseteq \bigcap_n \mathcal{A}_n$. *Um conjunto* $A \in 2^{\varpi}$, *statisfaz o teste* $\mathcal{T}$ *quando a classe singular* $\{A\}$ *satisfaz esse teste.*

94 FACTO

A colecção de todos os testes uniformemente recursivos é enumerável.

Esboço de prova

Dado que cada sequência uniforme de funcionais efectivas é completamente determinda por uma função recursiva, a enumeração das funções recusivas induz a enumeração dos testes uniformemente recursivos.

No entanto testes genéricos, definidos desta forma não são mensuráveis porque as classes $\mathcal{A}_n$ não são eventos. Por isso faz sentido substituir as diferentes classes $\mathcal{A}_n$ por coberturas que possam ser mensuráveis.

95 Noção (Teste de Martin-Löf)
Fixemos uma medida de Lebesgue μ. Uma sequência de eventos recursivamente enumeráveis $\mathcal{U} = \{U_k\}$ é um **μ-teste de Martin-Löf (ML-teste)** *se, para todo k, $\lambda(U_k) > k$ e existe um teste uniformemente recursivo $\mathcal{T} = \{\mathcal{A}_k\}$ que é coberto por $\mathcal{U}$; isto é, para todo k, $\mathcal{A}_k \subseteq U_k$.*

Convém focar nas três componentes essencias desta definição:

- Cada classe $\mathcal{A}_k$ é efectivamente enumerável. Isto significa que existe uma "array" A_k que a representa e que os conjuntos na colecção são exactamente os recursivamente enumeráveis que contém um conjunto finito indexado por esse "array".
- Os teste de Martin-Löf são uniformemente recursivos. Isto implica, imediatamente, que existe uma enumeração de todos estes testes.
- Porque $\mathcal{A}_k$ é coberto por U_k tem-se $\lambda(\mathcal{A}_k) > k$. Assim a medida de Hausdorff de $H(\mathcal{A}_k)$ está limitada a $2^{-\lambda(\mathcal{A}_k)} < 2^{-k}$.

96 Facto
Se $\mathcal{T} = \{\mathcal{A}_k\}$ é um teste uniformemente recursivo tal que $\lambda(\mathcal{A}_k) > k$, para todo k, então existe um teste de Martin-Löf $\mathcal{U} = \{U_k\}$ que cobre $\mathcal{T}$.

Notas

O menor que evento que cobre cada $\mathcal{A}_k$ é a sua cobertura mínima $\hat{\mathcal{A}}_k$; assim, se for $\lambda(\mathcal{A}_k) > k$, tem-se também $\lambda\left(\hat{\mathcal{A}}_k\right) > k$.

No entanto não é possível garantir que estas coberturas mínimas sejam recursivamente enumeráveis.

Recordemos, porém, que a cobertura mínima é sempre construída como o limite de n-coberturas $\hat{\mathcal{A}}_k = \bigcap_n \mathcal{A}_k^{(n)}$, e que todas estas n-coberturas são recursivamente enumeráveis. Se for $\lambda\left(\hat{\mathcal{A}}_k\right) > k$ então tem de existir algum n tal que $\lambda\left(\mathcal{A}_k^{(n)}\right) > k$.

Definindo $U_k = \mathcal{A}_k^{(n)}$, para um n tal que $\lambda\left(\mathcal{A}_k^{(n)}\right) > k$ obtemos o desejado teste de Martin-Löf.

97 NOÇÃO (ALEATORIEDADE DE MARTIN-LÖF)
Fixe-se uma medida de Lebesgue μ. Uma classe $\mathcal{A} \subseteq 2^{\varpi}$ diz-se **μ-Martin-Löf nula (ML μ-nula)** *se existe algum ML μ-teste $\{\mathcal{U}_n\}$ tal que $\mathcal{A} \subseteq \bigcap_n \mathcal{U}_n$. Um conjunto A diz-se* **ML μ-aleatória** *quando a classe $\{A\}$ não for ML μ-nula; isto é, quando A não verifica qualquer ML μ-teste.*

Uma análise detalhada das noções de teste e aleatoriedade de Martin-Löf, assim como a relação destes conceitos com outras noções de aleatoriedade, pode ser vista no excelente artigo de Downey et al.[19].

Os conceitos de KLC-aleatoriedade e ML-aleatoriedade estão intimante ligados. De facto

[19] Rod Fowney, Denis R. Hirschfeldt, André Nies, Sebastain Terwijn, CALIBRATING RANDONNESS, The Bulletin of Symbolic Logic, Vol. 12, Nº 3, Setembro 2006.

98 Teorema (Schorr)

Seja μ a medida standard de Lebesgue; um conjunto é KLC-aleatório se e só se for ML μ-aleatório.

Esboço de Prova

Para vermos a intuição por detrás desta relação entre estas duas abordagens à aleatoriedade, considere-se um qualquer teste de Martin-Löf na medida standard de Lebesgue. O teste é completamente determinado pela classe uniformemente enumerável $\mathcal{U} = \{U_k\}$ de apresentações U_k, que são livres de prefixos, de tal forma que $\sum_{y \in U_k} 2^{-|y|} < 2^{-k}$.

Nesta representação, um conjunto A verifica o teste sse, para todo k existe um n tal que $A \upharpoonright n \in U_k$. Define-se, agora, um conjunto R tal que

$$d \| y \in R \quad \Leftrightarrow \quad k > 0 \wedge d = |y| - k \wedge y \in U_{2\,k}$$

É fácil verificar que R é recursivo (já que os U_k formam uma sequência uniformemente recursiva) e que o seu peso é

$$\sum_{k>0} \sum_{y \in U_{2k}} 2^{-(|y|-k)} = \sum_{k>0} 2^k \left(\sum_{y \in U_{2\,k}} 2^{-|y|} \right) < \sum_{k>0} 2^k\, 2^{-2\,k} < 1$$

Portanto R é um conjunto de Kraft-Chaitin e, pelo teorema 90, existe uma constante c tal que, para todo $k > 0$ e todo $y \in U_{2k}$, $K(y) \leq |y| - k + c$; equivalentemente $K(y) - |y| \leq c - k$.

Se A for KC-aleatório, e se for $y = A \upharpoonright n$, existe um k tal que $K(y) - |y| > c - k$. Por isso não podemos ter $A \upharpoonright n = y$ para nenhum $y \in U_{2k}$ e, consequentemente, o teste falha. Ou seja, *KLC-aleatoriedade implica ML-aletoriedade*.

A prova inversa é uma simples consequência do facto 84 cuja prova se repete aqui. Para cada k considere-se a linguagem U_k definda por

$$U_k = \{ y \mid K(y) \leq |y| - k \wedge (\forall z < y)[K(z) > |z| - k] \} \tag{41}$$

Pelo teorema 76 U_k é recursivamente enumerado; claramente U_k é livre de prefixos. Portanto a sequência de linguagens $\{U_k\}_{k\in\varpi}$ é uniformemente recursiva. Como a indexação é livre de prefixos, tem-se $\sum_{y\in U_k} 2^{-K(y)} \leq 1$; logo

$$\mu(\uparrow U_k) = \sum_{y\in U_k} 2^{-|y|} \leq \sum_{y\in U_k} 2^{-(K(y)+k)} \leq \sum_{y\in U_k} 2^{-K(y)} \cdot 2^{-k} \leq 2^{-k}$$

Portanto a sequência $\{\uparrow U_k\}$ é um teste de Martin-Löf. Se A for ML-aleatória tem-se $A \notin \bigcap_k \uparrow U_k$. Isto significa que existe um k tal que $A \notin \uparrow U_k$. Ou seja, nenhum prefixo de A pode pertencer a U_k; isto é $(\forall n)\,[A\upharpoonright n \notin U_k] \equiv (\forall n)\,[K(A\upharpoonright n) > n - k]$. Mas isto significa que A é KLC-aleatório. Portanto: *ML-aleatoriedade implica KLC-aleatoriedade*.

Aleatoriedade de Shannon-Schorr

O terceiro paradigma de aleatoriedade designa-se por **paradigma da imprevisibilidade** e invoca uma intuição muito óbvia: *se uma string infinita u é aleatória, o conhecimento dos seus n primeiros bits, não fornece qualquer informação que permita prever o bit seguinte*.

Esta intuição pode ser concretizada através de uma noção de "estratégia ganhadora de aposta": um jogador define uma regra que, conhecendo o passado x de resultados do jogo, lhe permite dividir o lucro que acumulou em x pelas hipóteses de jogo que podem continuar x.

No caso do jogo ser binário (lançar uma moeda ao ar, por exemplo), teremos uma relação da forma

$$\text{lucro}(x) \;=\; \text{aposta}(x\,0) \;+\; \text{aposta}(x\,1)$$

Como o resultado do jogo é binário, só faz sentido ter $\text{lucro}(x) \leq 2\,\text{aposta}(x)$ (o lucro está limitado ao dobro da aposta). A generalização é óbvia

99 Noção (Martingales)
Uma função $d\colon \varpi \to \mathbb{R}_+$ *é um* **super-martingale** *se, para todo* $x \in \varpi$,

$$2\,d(x) \geq d(1\|x) + d(0\|x)$$

Quando a igualdade $2\,d(x) = d(1\|x) + d(0\|x)$ *se verifica para todo* x, d *designa-se por* **martingale**.

d **sucede** *em* $A \in 2^{\varpi}$, *e escreve-se* $A \sqsubseteq d$, *se* $\lim_{n\to\infty} \sup d(A \restriction n) = \infty$; d *sucede numa classe* $\mathcal{A}$ *de conjuntos se sucede em todos conjuntos dessa classe.*

É muito importante notar-se que cada valor $d(A \restriction n)$ é um real positivo e, por isso, só são conhecidos os seus limites superiores racionais. Daí o limite $\lim_{n\to\infty} \sup d(A \restriction n)$ se aplicar aos seus maximizantes racionais.

Os martingales são super-martingales que representam "estratégias ganhadores eficientes": todo o lucro é investido em apostas e o lucro é sempre o dobro da aposta ganhadora. A sua importância deriva do seguinte teorema:

100 Teorema (Ville)
Uma classe $\mathcal{A} \subseteq 2^{\varpi}$ é Lebesgue nula (é coberta por um evento nulo) se e só se existe um martingale d que sucede em $\mathcal{A}$.

A definição genérica de super-martingale (ou de martingale) não dá qualquer informação sobre as características computacionais destes objectos. Obviamente, só faz sentido uma estratégia ganhadora que possa ser computada!

Em primeiro lugar os diversos reais $\{d(x)\}$ devem se reais recursivos. Isto significa que cada um destes valores é determinado por um conjunto recursivo e limitado de racionais; como existe uma correspondência biunívoca entre racionais positivos e naturais, cada $d(x)$ identifica-se com um conjunto recursivo de naturais: os códigos desses racionais. Exige-se algo mais: deve existir uma forma computacionalmente relevante de gerar todos os conjuntos $d(x)$. Estas considerações motivam a seguinte definição:

101 Noção (Martingale efectivo)
Um super-martingale (ou martingale) d diz-se **efectivo** *(ou* **computacionalmente enumerável***) de todo $d(x)$ é um real recursivo e existe uma função recursiva f tal que $d(x) = \mathcal{W}_{f(x)}$.*

Exemplo 10: Considere-se a seguinte função $d\colon 2^{<\varpi} \longrightarrow \mathbb{R}_+$

$$d(x) \;=\; \sum_{y \leq x} 2^{|y|-K(y)} \;+\; \sum_{y > x} 2^{|x|-K(y)} \tag{42}$$

Note-se, em primeiro lugar, que $d(x)$ é um real recursivo que depende recursivamente de x; basta notar que a 1ª parcela é uma soma finita que depende recursivamente de x, que a 2ª parcela $2^{|x|} \sum_{y>x} 2^{-K(y)} \leq 2^{|x|}$. Adicionalmente, calculando $d(x\,0)$ e $d(x\,1)$, é fácil verificar $2\,d(x) = d(x\,0) + d(x\,1)$. Portanto $d(\cdot)$ é um martingale efectivo.

Considere-se de novo os conjuntos $U_c = \{\, x \mid K(x) \leq |x| - c \,\}$. Uma vez que $d(x) \geq 2^{|x|-K(x)}$ (basta seleccionar em (42) só a parcela $y = x$), então, para todo $x \in U_c$, tem-se $d(x) \geq 2^c$. Tome-se agora um qualquer conjunto A não-aleatório; pela definição sabemos que, para todo c, existe um n tal que $A \upharpoonright n \in U_c$. Portanto, para todo c, existe um n tal que $d(A \upharpoonright n) \geq 2^c$; logo $\lim_n d(A \upharpoonright n) = \infty$; i.e. d sucede em A.

Alternativamente, considere-se agora um qualquer conjunto A aleatório; tome-se $x = A \upharpoonright n$, par um qualquer n. Note-se que $y \leq x$ sse y for da forma $A \upharpoonright k$ com $k \leq n$. Então tem-se $d(A \upharpoonright n) \geq \sum_{k \leq n} 2^{k-K(A\upharpoonright k)}$. Portanto $\lim_{n\to\infty} d(A \upharpoonright n) \geq \sum_{n\in\varpi} 2^{n-K(A\upharpoonright n)}$.

□

Temos agora o resultado definitivo que ajuda a fechar a comparação entre estes três paradigmas.

102 TEOREMA (SCHORR)
Um classe de $\mathcal{A}$ é Martin-Löf nula se e só se existe um super-martingale efectivo d que sucede em $\mathcal{A}$

Como corolário óbvio

103 Corolário *Um conjunto A será ML-aleatório (e também KLC-aleatório) se e só se não existe nenhum supermartingale efectivo que sucede em A.*

1.6 Comparar a Aleatoriedade

Na secção anterior consideramos vários pontos de vista sobre a aleatoriedade de um conjunto A e estudou-se as possíveis equivalências entre estas pontos de vista. A pergunta essencial era sempre: *dado um conjunto A, será que ele é aleatório?*.

Porém as simples respostas a esta pergunta dão, frequentemente, informação insuficiente. Normalmente quer-se ter alguma forma de "grau de aleatoriedade" que permita medir quão aleatório é A. Ou, pelo menos, poder comparar a aleatoriedade de dois conjuntos A e B, testando se um é mais ou menos aleatório que o outro ou se são incomparáveis em termos de aleatpriedade. Dar algumas respostas a estas questões é o objectivo desta secção.

Aleatoriedade e os graus de Turing

Parece intuitivo que conjuntos aleatórios não podem ser efectivamente computáveis. A intuição é mais difícil quando se lida com semi-decidibilidade: *"será que um conjunto aleatório pode ser recursivamente enumerável?"*. Esta questão pode-se colocar em termos de uma mais geral: *"será que existem conjuntos KLC-aleatórios A que sejam Turing redutíveis a algum conjunto recursivamente enumerado?"* e a sua resposta foi dada por Kučera

104 Teorema (Kučera)
Se A tem grau recursivamente enumerável (i.e. é Turing redutível a algum conjunto recursivamente enumerado) e é KLC-aleatório, então, forçosamente, $A \equiv_T \mathcal{K}$.

Recordemos que $\mathcal{K}$ é o conjunto recursivamente enumerável (mas não recursivo) dado por $x \in \mathcal{K} \Leftrightarrow \varphi(x\|x)\downarrow$. Vimos que não existência de uma enumeração recursiva para o complemente de $\mathcal{K}$ prende-se directamente com o problema da indecidibilidade da paragem em máquinas de Turing e, por isso, $\mathcal{K}$ pode ser visto como o "menos previsível" de todos os conjuntos recursivamente enumerados.

Este resultado diz-nos que qualquer conjunto aleatório A que tenha grau r.e. se comporta (em termos de eventuais reduções recursivas) exacmatente como $\mathcal{K}$. Nomeadamente, dado que qualquer conjunto recursivamente enumerável R verifica $R \leqslant_T \mathcal{K}$, também verificará $R \leqslant_T A$.

Comparação de Kolmogorov

Pode-se comparar a aleatoriedade de dois conjuntos A e B comparando os seus graus de compressibilidade através da comparação das complexidades de Kolmogorov dos seus diversos prefixos.

105 Noção (Redução de Turing)
*O conjunto A é K-**redutível** ao conjunto B, e escreve-se $A \leqslant_K B$, quando $K(A\restriction n) = O^{\log}(K(B\restriction n))$. Isto é, quando $(\forall_\infty n)\,[K(A\restriction n) \leq K(B\restriction n) + O(1)]$. Se $A \not\leqslant_K B$ e $B \not\leqslant_K A$, os conjunto dizem-se K-**incomparáveis** e escreve-se $A|_K B$.*

Esta relação de ordem permite, desde já, uma outra questão: *que construções de conjuntos preservam aleatoriedade?*

Usaremos a notação $A \uplus B$ para designar a união disjunta[20] de conjuntos definida como

$$A \uplus B \quad = \quad \{\, 2\,x \mid x \in A \,\} \cup \{\, 2\,x+1 \mid x \in B \,\} \tag{43}$$

O seguinte teorema relaciona a aleatoriedade de A e de B com a da sua união disjunta.

106 Teorema (Miller, van Lanbalgen, Kautz, Kucera)
Se $A \uplus B$ é KLC-aleatório, então ambos A e B são KLC-aleatórios. Adicionalmente A é KLC-aleatório relativo a B (e B é KLC-aleatório relativo a A) e ainda tanto A como B são K-incomparáveis com $A \uplus B$.

□

Outra abordagem à comparação entre conjuntos recorre à sua interpretação como reais e é devida, principalmente, ao trabalho de Solovay.

Recordemos que cada $u \in 2^{<\varpi}$ pode ser vista como um racional $\sum_{u_i=1} 2^{-i-1}$. Desta forma se, como strings, for $u \leq v$ (u é um prefixo de v), então na ordem usual dos racionais também se tem $u \leq v$. Equivalentemente, cada $x \in \varpi$ está associado ao conjunto finito D_x e ao racional $\sum_{i \in D_x} 2^{-i-1}$.

[20] Os textos de computabilidade usam, normalmente, a notação $A \oplus B$ para designar união disjunta; no entanto, neste curso, a notação $x \oplus y$ será usada para designar o `xor` bit a bit de duas strings. A string que representa $A \uplus B$ é, ao invés, formada pelo "merge" alternado dos bits de A e de B.

Um conjunto A pode ser visto como o real que é limite da sequência crescente de racionais $\{A\restriction n\}_n$. Recordemos também (ver noção 60, página 59) que um **real recursivo** é o limite de uma sequência recursiva crescente de racionais.

O real A é **fortemente recursivo** se a sequência $\{A\restriction n\}_n$ é recursiva. Note-se que qualquer real fortemente recursivo é, obviamente, recursivo mas o inverso não é verdade: existem reais recursivos que não são fortemente recursivos. Para ilustrar esta diferença, e como consequência do teorema 59 (página 58), temos o seguinte resultado.

No que se segue, φ é uma função universal livre de prefixos, tida como referência, e $\{\varphi_e\}_{e\in\varpi}$ é a enumeração, induzida por φ, das funções parciais com domínios livres de prefixos.

107 Proposição
A é um real recursivo se e só se for da forma $\sum_{\varphi_e(x)\downarrow} 2^{-|x|}$ para algum programa e. [21]

Esboço de prova
Se A for o limite de uma sequência recursiva e crescente de racionais $\{r_n\}$, seja $\{l_n\}$ a sequência crescente de racionais de Lebesgue que verifica $l_n \leq r_n < l_n + 2^{-n}$ e $|l_n| = n$. Claramente a sequência $\{l_n\}$ é recursiva e tem A como limite; a partir desta sequência é possível construir uma sequência recursiva $\{d_n\}$ tal que a série $\sum_n 2^{-d_n}$ tem exactamente o mesmo limite A. Usando o facto 73, página 75 conclui-se que A é a medida de um domínio livre de prefixos de uma função parcial recursiva φ_e. Inversamente, dada uma linguagem recursiva livre de prefixos L, a sua medida é claramente um real recursivo.

[21] Equivalentemente, se for a medida do domínio de uma máquina de Turing livre de prefixos.

Como consequência deste resultado faz sentido introduzir a seguinte notação:

108 DEFINIÇÃO
Para uma função universal livre de prefixos φ, $\Omega \doteq \sum_{\varphi(x)\downarrow} 2^{-|x|}$ *é o real recursivo definido por* φ *e que se designa por* **Ω-conjunto de Chaitin**. *Genericamente, para um índice* e, $\Omega_e \doteq \sum_{\varphi_e(x)\downarrow} 2^{-|x|}$ *é o real recursivo determinado pela função* φ_e .

A proposição 107 diz que todo o real recursivo é da forma Ω_e para algum índice e. Nomeadamente os reais recursivos são recursivamente enumeráveis.

Note-se que $\sum_{\varphi(x)\downarrow} 2^{-|x|}$ e $\sum_{\varphi_e(x)\downarrow} 2^{-|x|}$ representam as medidas das linguagem livres de prefixos que são os domínios de φ ou φ_e: tem-se $\Omega = \mu(\mathrm{dom}(\varphi))$ e, genericamente, $\Omega_e = \mu(\mathrm{dom}(\varphi_e))$. Se M_e for uma qualquer máquina de Turing que compute φ_e, o real Ω_e mede a probabilidade de uma bit-strings infinita arbitrária ser aceite por tal máquina. Por isso Ω_e é designada por **probabilidade de paragem** do programa e. Em particular Ω é a **probabilidade de paragem** da máquina de Turing universal livre de prefixos ou, simplesmente, a probabilidade de paragem.

Limitando a recursividade, obtém-se variantes dos reais de Chaitin. Sendo φ^l a função universal livre de prefixos obtida de φ com recursividade limitada a l passos, então, $\Omega^l \doteq \mu(\mathrm{dom}(\varphi^l))$ e $\Omega^l_e \doteq \mu(\mathrm{dom}(\varphi^l_e))$.

Na teoria clássica da computabilidade o grau r.e. de $\mathcal{K} = \{\, x \mid \psi_x(x)\downarrow \,\}$ não depende da enumeração aceitável

$\{\psi_e\}$ das funções parciais recursivas; isto porque existe um teorema que diz que qualquer outro sistema aceitável $\{\psi'_e\}$ determina um $\mathcal{K}'$ que é Turing-equivalente a $\mathcal{K}$ e, por isso, tem o mesmo grau.

Na aleatoriedade algorítmica, Ω depende da função universal φ tomada por referência. Por isso, para sermos precisos, deve-se dizer que determina uma probabilidade de paragem. Para que Ω determine a probabilidade de paragem, é necessário um resultado que estabeleça uma redução forte entre os diversos Ω definidas pelas diversos candidatos a funções universais livres de prefixos. Isto exige uma nova noção de redução algorítmica.

Antes, porém, convém apresentar um resultado que está na origem da importância atribuída ao real Ω.

109 Lema *Ω é KLC-aleatório.*

Esboço de prova

Pela definição de KLC-aleatoriedade necessitamos de provar que existe um $c > 0$ tal que, para todo n, $K(\Omega \restriction n) \geq n - c$. Equivalentemente, que existe um $c > 0$ tal que, para todo n e todo $x \in \mathrm{dom}(\varphi)$ com $|x| < n - c$, se tem $\varphi(x) \not\simeq \Omega \restriction n$.

Pelo teorema 104, página 100, Ω é Turing equivalente a $\mathcal{K}$; nomeadamente, isto significa que, para todo n existe um limite à recursividade l tal que o prefixo $\Omega \restriction n$ está no contradomínio de φ^l; i.e., para algum x tem-se $\varphi^l(x) \simeq \Omega \restriction n = \Omega^l \restriction n$. Vamos definir um programa e tal que $\varphi_e(x) \notin \mathrm{rng}(\varphi^l)$ para todo x tal que $|x| < n - c$ sendo $c = |e| + K(e)$; equivalentemente, quando $|e\|x| < n$, temos $|e\|x| \notin \mathrm{dom}(\varphi^l)$. Isto significa que $\Omega - \Omega^l \geq 2^{-|e\|x|} > 2^{-n}$ e, portanto, $\Omega \restriction n \neq \Omega^l \restriction n$. Assim $|x| < n - c$ implica que $\varphi(x) \neq \Omega \restriction n$.

Este lema mostra que o real recursivo Ω é aleatório[22]; inversamente gostariamos de mostrar que todo o real recursivo que seja aleatório é, de alguma forma, redutível a Ω. Para isso precisamos de uma noção de redução em que, nos reais, Ω tenha propriedades semelhantes às que $\mathcal{K}$ tem nos graus de Turing.

110 Noção (Redução de Solovay)
Diz-se que o real A é **Solovay-redutível** *ao real B (equivalentemente, B* **domina** *A), e escreve-se $A \leqslant_S B$, quando existe uma constante $C > 0$ e uma função computável[23] f tais que, para todo racional q, se verifica $A - f(q) < C\,(B - q)$.*

Essencialmente a definição diz-nos que, se existir uma sucessão crescente de racionais $\{q_n\}$ que tenha B como limite, é possível construir uma sucessão $\{f(q_n)\}$ que converge para A de forma tão ou mais rápida. Isto implica, nomeadamente, que A deve ser "menos aleatório" do que B. De facto Solovay provou

111 Lema (Solovay) *Se $A \leqslant_S B$ então, para todo n, verifica-se $K(A\restriction n) \leq K(B\restriction n) + O(1)$.*

O facto de reais recursivos se identificarem com probabilidades de paragem de máquinas de Turing livres de prefixos permite obter uma forma explicita para esta relação quando aplicada a reais recursivos.

[22] No que se segue, a menos que seja explícito o modelo de aleatoriedade usado, "aleatório" significa indistintamente KLC-aleatório, Martin-Löf aleatório ou Shannon aleatório.

[23] Calculada dentro de uma determinada classe de complexidade de máquinas de Turing; por exemplo, como é convencional em Criptografia, a classe PPT.

112 TEOREMA
Sejam Ω_a , Ω_b reais recursivos determinados por programas a e b respectivamente. Então verifica-se $\Omega_a \leqslant_S \Omega_b$ sse existe uma função total computável f e uma constante $C > 0$ tais que,

$$(\forall l) \quad \left[\Omega_a - \Omega_a^{f(l)} < C\,(\Omega_b - \Omega_b^l) \right] \tag{44}$$

113 COROLÁRIO *Se A e B são limites de sequências crescentes e computáveis de racionais $\{p_n\}$ e $\{q_n\}$, respectivamente, então $A \leqslant_S B$ sse existe uma função total computável f e uma constante $C > 0$ tais que*

$$(\forall n) \quad \left[A - p_{f(n)} < C\,(B - q_n) \right] \tag{45}$$

Todos estes resultados traduzem a noção fundamental da redução de Solovay $A \leqslant_S B$: a partir de uma qualquer aproximação computável a B é possível construir uma aproximação a A que converja tão ou mais rapidamente que a aproximação de B.

Adicionalmente, aplicada a reais recursivos, a redução de Solovay assume propriedades idênticas à da redutibilidade de Turing. Para tal necessitamos de uma noção análoga ao grau $\mathbf{0}'$ e um resultado análogo ao teorema de Post.

114 NOÇÃO
*Um real recursivo A diz-se Ω-**equivalente** (ou **Solovay-completo**) quando domina Ω (i.e. $\Omega \leqslant_S A$).*

O seguinte resultado resume a relação entre a aleatoriedade dos reais recursivos e a redução de Solovay.

115 Teorema
Ω domina todo real recursivo A. Adicionalmente,

(i) Todo o real recursivo Ω-equivalente é a probabilidade de paragem de uma função universal livre de prefixos; portanto, é KLC-aleatório.

(ii) Todo o real recursivo KLC-aleatório é Ω-completo.

Este resultado é fundamental ao nosso estudo já que estabelece uma correspondência biunívoca entre quaisquer duas de três classes de entidades: os reais recursivos Ω-completos, as probabilidades de paragem de máquinas universais de Turing livres de prefixos e reais recursivos KLC-aleatórios.

Um aspecto interessante é a relação entre a redução de Solovay e a adição de reais recursivos. O seguinte resultado resume as relações essenciais.

116 Facto
Sejam A e B reais recursivos arbitrários. Então

1. *Se for $A + B = \Omega$ então pelo menos uma das parcelas, A ou B, é Ω-equivalente.*
2. *Verifica-se $A \leqslant_S B$ sse existe um real recursivo X e uma constante c tais que $A + X = c\,B$.*

□

Uma das construções mais importantes em Criptografia é o de **geradores pseudo-aleatórios**; isto é, sequências de "bits" que, até um certo ponto, são indistinguíveis de uma sequência aleatória. Sequências diferentes são geradas por "seeds" diferentes.

A definição aqui apresentada é expressa em termos de reais-recursivos; o gerador é uma sequência, indexada pelos "seeds", de reais recursivos. A noção de "indistinguibilidade" é expressa limitando os níveis de recursividade com que as aproximações são feitas.

117 NOÇÃO
Uma classe efectivamente computável de reais recursivos $\left\{\Omega_{e(s)}\right\}$ *é um* **gerador pseudo-aleatório** *para a* **função de expansão** L, *se existe uma função computável* f *e uma constante* $C > 0$ *tais que, para cada* s,

$$\forall\, l \leq L(|s|) \quad \left[\; \Omega - \Omega^{f(l)} \quad \leq \quad C\,\left(\Omega_{e(s)} - \Omega^{l}_{e(s)}\right)\;\right] \tag{46}$$

Note-se que, para qualquer "seed" s, a quantificação nos níveis de recursividade l não é ilimitada; se o fosse para algum s, então esse $\Omega_{e(s)}$ seria sempre Ω-equivalente (teorema 115) e, portanto, realmente aleatório. A função L estabelece um limite nos níveis de recursividade que podem ser usados para comparar as razões de convergência para Ω e $\Omega_{e(s)}$ das respectivas aproximações.

□

Considere-se uma qualquer conjunto de Kraft-Chaitin $X = \{d_n \| x_n\}_{n\in\varpi}$; a essência do teorema de Chaitin é que existe um programa p que determina uma máquina de Turing livre de prefixos M_p cujo domínio é enumerado por uma sequência recursiva $\{u_n\}$ verificando $|u_n| = d_n$ e $M_p(u_n) \simeq x_n$, para todo n.

Nomeadamente peso $\sum_n 2^{-d_n}$ do KC-conjunto X coincide com o real recursivo Ω_p .

Obviamente que vários programas p conduzem a máquinas de Turing M_p que satisfazem estas condições; nada se diz sobre a sua natureza ou a sua complexidade; pode acontecer, por exemplo, que uma máquina M_p, para cumprir estas condições, tenha uma complexidade temporal que é exponencial com o comprimento do "input" [24]. No entanto, independentemente do programa p, o real recursivo Ω_p é sempre o mesmo e é determinado por X.

Por isso é importante identificarmos classes de complexidade nos conjuntos KC análogas às que são definidas para as máquinas de Turing. Assim define-se

118 NOÇÃO
Diremos que o conjunto KC $X = \{|u_i| \,\|\, x_i\}$ *é* ***computável*** *se existe uma máquina de Turing efectivamente computável* M_p *que verifique* $(\forall i)\,[\, M_p(u_i) \simeq x_i\,]$ *e* $M_p(u)\downarrow \Rightarrow (\exists i)\,[\, u = u_i\,]$.

[24] Neste caso o comportamento aleatório dos "inputs" não permite, realisticamente, inferir qualquer comportamento das respostas $\{x_i\}$.

Se se convencionar, como é usual em análise de segurança, que "efectivamente computável" significa que a máquina de Turing está na classe PPT, então um conjunto KC será computável quando existir um algoritmo probabilístico, com tempo polinomial que, com margem de erro desprezável, rejeite qualquer "input" que não pertença à sequência $\{u_i\}$ e compute, para todo i, x_i a partir de u_i.

Intuitivamente este algoritmo estabele uma ligação entre a aleatoriedade da sequência de respostas $\{x_i\}$ e dos "inputs" $\{u_i\}$: se a classe de complexidade for suficientemente restrita, a primeira nunca é mais aleatória do que a segunda.

Vimos que o peso $w(X)$ de um KC-conjunto $X = \{|u_i| \parallel x_i\}$ identifica-se com um real recirsivo Ω_p; inversamente qualquer real recursivo Ω_e pode ser identificado com o conjunto de Kraft-Chaitin da forma $\{\, |u| \| u\}_{u \in \mathrm{dom}(\varphi_e)}$.

Faz sentido, então, analisar a forma como a ordem $\leqslant_S$ de Solovay nestes pesos induz uma forma de comparação da aleatoriedade associada a KC-conjuntos.

119 NOÇÃO
Sejam X, Y conjuntos Kraft-Chaitin e Ω_p, Ω_q os reais recursivos que representam os respectivos pesos. X é ***redutível*** *a Y, e escreve-se $X \leqslant Y$, quando $\Omega_p \leqslant_S \Omega_q$.*

A redutibilidade de Solovay estabelece-se na forma como as aproximações a um real recursivo convergem. Quando os reais recursivos são pesos $w(X)$, as aproximações podem ser expressas de uma forma específica.

Defina-se $w(X;n) = \sum_{i\geq n} 2^{-|u_i|}$.

Note-se que $w(X) - w(X;n) = \sum_{i<n} 2^{-|u_i|}$ é um racional que aproxima o real $w(X) = \Omega_p$; assim $w(X;n)$ pode ser visto como o erro dessa aproximação: avalia o erro que se comete quando, em X, se ignoram "inputs" (e respostas) de ordem $\geq n$. Obviamente que $\lim_n w(X;n) = 0$.

Defina-se também $\lambda(X;n) = \max\{\, \lambda \in \varpi \mid \sum_{i\geq n} 2^{\lambda-|u_i|} \leq 1 \,\}$.

Tem-se sempre $\lim_n \lambda(X;n) = +\infty$. Pode-se interpretar $\lambda(X;n)$ como um limite inferior para os comprimentos de dos "inputs" u_i que determinam as respostas x_i de ordem $i \geq n$; dito de outro modo, $\lambda(X;n)$ mede a incerteza nas respostas x_i de ordem $i \geq n$.

$$\varphi_e(u) \simeq x_i \wedge i \geq n \quad \Rightarrow \quad |u| \geq \lambda(X;n)$$

Equivalentemente, a informação necessária para determinar as respostas de ordem inferior a n, está concentrada nos "inputs" u de comprimento limitado a $\lambda(X;n)$. Nesta perspectiva, quanto mais a informação global sobre X estiver "concentrada" nas respostas até n, maior será o "comprimento" $\lambda(X;n)$.

Considere-se agora um segundo conjunto Kraft-Chaitin $Y = \{|v_i|\|y_i\}$. Como em qualquer KC-conjunto tem-se também $\lim_n w(Y;n) = 0$ e $\lim_n \lambda(Y;n) = +\infty$. Comparando as razões de convergência, para zero dos pesos $w(X;n)$ e $w(Y;n)$ ou para $+\infty$ dos comprimentos $\lambda(X;n)$ e $\lambda(Y;n)$, é possível relacionar a aleatoriedade dos dois KC-conjuntos. Do corolário 256 conclui-se

120 Facto

Verifica-se $X \leqslant Y$ *se e só se existe uma função computável* f *tal que, para todo* n*,*

$$\lambda(X; f(n)) \geq \lambda(Y; n) + O(1)$$

A relação de redução $\leqslant$ é uma ordem parcial e, naturalmente, define classes de equivalências que serão designados por KC-**graus**; $[X]$ denota o KC-grau que contém X. Diz-se que X é Ω-**equivalente** quando $\Omega \leqslant X$; assim, o KC-grau formado pelos KC-conjuntos Ω-equivalentes é representada por $[\Omega]$.

□

A relação de redução de conjuntos Kraft-Chaitin pode ser transderida para sequência de respostas.

Note-se que vários KC-conjuntos $X = \{d_i \| x_i\}$ podem ter a mesma sequência $\alpha = \{x_i\}$ de respostas. Também, como as sequências de respostas de um KC-conjunto é obrigatoriamente recursiva, é natural que seja necessário lidar com sequências que não coincidem exactamente com as respostas de nenhum KC-conjunto; por exemplo, a sequência $\{A \upharpoonright n\}$ das truncaturas de um conjunto aleatório A não é recursiva.

Por isso convém estabelecer explicitamente uma relação entre sequências e KC-conjuntos.

121 Noção

Uma sequência $\alpha = \{x_n\}_{n \in \varpi}$ *é* **prevista** *pelo conjunto de Kraft-Chaitin* X *quando for uma subsequência da*

sequência de respostas de X.
O **fecho computável**, *representado por* $[\![\alpha]\!]$, *é a classe de todos os KC-conjuntos computáveis que prevêm* α.

Pode-se agora estabelecer uma relação de redução entre sequências (recursivas ou não).

122 NOÇÃO
A sequência α *é* **redutível** *à (ou menos* **aleatória** *do que a) sequência* $\beta = \{y_n\}_{n\in\varpi}$, *e escreve-se* $\alpha \leqslant \beta$, *quando, para todo o KC-conjunto computável* Y *que prevê* β, *existe um KC-conjunto computável* X *que prevê* α *e é redutível a* Y.
A sequência α *é* Ω-**completa** *quando todo o KC-conjunto computável que preveja* α *é* Ω-*completo.*

A definição anterior pode-se reescrever usando fechos computáveis.

$$\alpha \leqslant \beta \quad \Leftrightarrow \quad (\forall Y \in [\![\beta]\!])\ (\exists X \in [\![\alpha]\!])\ [X \leqslant Y] \tag{47}$$

$$\alpha \text{ é } \Omega\text{-completa} \quad \Leftrightarrow \quad [\![\alpha]\!] \subseteq [\Omega] \tag{48}$$

De forma natural se introduz a noção de "pseudo-completude" de KC-conuntos e sequências substituindo Ω por um gerador pseudo-aleatório $\left\{\Omega_{e(s)}\right\}$ e quantificando em relação a todas as "seeds" de tamanho limitado.

1.7 Incerteza

Na noção 81, página 79 vimos duas formas simples para avaliar a incerteza de um conjunto A. Vimos a **incerteza de Hartley** (ou simplesmente **incerteza**) de um conjunto $A \neq \emptyset$, representada por $\vartheta(A)$, definida por $\vartheta(A) = \log_2[\![A]\!]$, sendo $[\![\cdot]\!]$ a função recursiva que mede a cardinalidade de um conjunto. Esta incerteza só faz sentido em conjuntos finitos. Vimos também como a **incerteza uniforme** contorna este problema, apresentando-se como uma funcional $\vartheta[\cdot]$ que, a cada conjunto A, associa uma função $\vartheta[A]$ tal que $\vartheta[A](n) \simeq \vartheta(A \restriction n)$.

Outra forma de incerteza é a **incerteza limite**, representada por $\vartheta_l[A]$ e já aplicável a conjuntos infinitos, definida como o tamanho do menor elemento de A; isto é, $\vartheta_l[A] = \min_{x \in A} |x|$.

Em qualquer destas formas de incerteza vamos convencionar que a incerteza de um conjunto vazio é $-\infty$.

A incerteza de Hartley $\vartheta(A)$ depende apenas do número de elementos do conjunto A; não depende dos elementos do conjunto. Desta forma a incerteza de um conjunto singular é sempre nula independentemente do inteiro que determina o conjunto. Já a incerteza limite desse conjunto singular depende do inteiro que o determina; de facto coincide com o tamanho desse inteiro.

O **tamanho** é uma outra medida aplicável apenas a conjuntos finitos; recorde-se que $|A|$ é definido como o tamanho do índice canónico do conjunto; isto é, $|A| = \log_2(1 + \sum_{a \in A} 2^a)$. Ou $|A| = |x|$ tal que $A = D_x$.

Em geral as várias medias de um conjunto finito (cardinalidade, incerteza e tamanho) produzem resultados distintos como se ilustra no seguinte exemplo.

Exemplo 11:
Suponhamos que A é o conjunto formado por todos os inteiros no intervalo $[0..n-1]$. A cardinalidade é $[\![A]\!] = n$. O código que o representa é $\sum_{k=0}^{n-1} 2^k = 2^n - 1$ cujo tamanho é $|A| = \log_2(1 + (2^n - 1)) = n$. Portanto, neste caso, $[\![A]\!] = |A| = n$ e $\vartheta(A) = \lceil \log_2 n \rceil$.

Suponhamos, agora, que n é par e o conjunto B é formado pelos inteiros pares no mesmo intervalo $[0..n-1]$. Obviamente que $[\![B]\!] = [\![A]\!]/2$ e que $\vartheta(B) \sim \vartheta(A) - 1$. O código de B é $\sum_{k=0}^{n/2-1} 2^{2\,k} = (2^n - 1)/3$; o tamanho deste código é $n - 2$; logo, neste caso, $|B| = |A| - 2$ é distinto da cardinalidade $[\![B]\!]$.

Generalizando, pode-se verificar que se formarmos um conjunto dos múltiplos de p contidos em A (isto é $\{\, i\,p \mid i = 0..n/p - 1 \,\}$) temos uma incerteza que é aproximadamente $\vartheta(A) - \log_2 p$ e um tamanho que é aproximadamente $|A| - p$.

A intuição por detrás da noção de incerteza pode ser vista por diferentes perspectivas das chamadas **propriedades discriminantes**

123 Noção
Seja X um domínio e $\sim$ uma relação de equivalência nesse domínio. Seja $\mathcal{T} = \{\rho_e\}$ uma família de relações

1. Sob o ponto de vista da **complexidade computacional**

1.8 Unidireccionalidade, Fuga de Informação e Geradores Pseudo-Aleatórios

Em Criprografia são essenciais as funções computáveis $f : \varpi \to \varpi$ cuja inversa f^{-1} não seja computável; informalmente pode-se exprimir esta exigência da seguinte forma:

> *dado um qualquer* y*, não deve ser computável decidir se* $f^{-1}(y) \neq \emptyset$ *ou, caso esta condição se verifique, encontrar um* x *em* $f^{-1}(y)$ *.*

Caso existam, essas funções são designada por **undirecionais** (**"one-way"**).

A condição de unidirecionalidade pode ser refinada de vários modos; nomeadamente pode-se colocar uma condição de ausência de **fuga de informação** (**"information leakage"**) do "input" para o "output". Informalmente existe

Vamos reformular a noção de teste de Martin-Löf em termos de funções parciais livres de prefixos. Vamos considerar, como referência, uma função universal livre de prefixos φ. Designaremos por φ_e e $\mathcal{W}_e$, respectivamente, a função universal livre de prefixos determinada pelo programa e, e o respectivo domínio.

124 NOÇÃO
Um real recursivo $A \in 2^{\varpi}$ *é* ***aceite*** *por* φ_e *(ou* φ_e ***sucede*** *em* A*), e escreve-se* $\varphi_e|A$ *, quando algum prefixo de* A *pertence ao domínio de* φ_e*; isto é* $\downarrow A \cap \mathcal{W}_e \neq \emptyset$ *ou, equivalentemente,* $A \in \uparrow \mathcal{W}_e$ *.*

Quando $\varphi_e|A$ *é decidível para todo o r.r.* A*, então* φ_e *diz-se uma* **observação**.

Uma sequência de observações uniformemente recursiva $\tau = \left\{\varphi_{e(k)}\right\}_{k\in\varpi}$ *determina um* **teste de Martin-Löf** *quando, para todo* k*,* $\mathcal{W}_{e(k+1)} \subseteq \mathcal{W}_{e(k)}$ *e* $\lambda(\mathcal{W}_{e(k)}) > k$ *.*

$A \in 2^{\varpi}$ **satisfaz** *o teste* τ*, e escreve-se* $\tau\|A$ *, se e só* $\varphi_{e(k)}|A$ *, para todo* k*.*

A sequência $\alpha = \{x_n\}_{n\in\varpi}$ **satisfaz** *o teste* τ*, e escreve-se* $\tau\|\alpha$ *, quando para todo o KC-conjunto computável* X *que preveja* α *, verifica-se* $\tau\|w(X)$ *.*

125 NOÇÃO

Dada uma função parcial rcursiva f*, uma* **fuga de informação de "input"** *("inputs information leak") para* f*, é uma par de testes de Martin-Löf,* $\langle\tau, \tau^f\rangle$ *tais que, para todo o real recursivo* A*, verifica-se* $\tau\|A$ *se e só se* $\tau^f\|f^{-1}(A)$ *.*

Dada uma função parcial rcursiva f*, uma* **fuga de informação de "input"** *("inputs information leak") para* f*, é uma par de testes de Martin-Löf,* $\langle\tau, \tau^f\rangle$ *tais que, para toda a sequência* $\alpha = \{x_n\}_{n\in\varpi}$*, verifica-se* $\tau\|\alpha$ *se e só se* $\tau^f\|f^{-1}(\alpha)$ *.*

f *é* **fortemente unidirecional** *se não tem quaisquer fugas de informação.*

*f é τ-**fortemente unidirecional** quando dado τ é intratável encontrar uma fuga de informação com teste τ no output.*

2.Teoria dos Números

As técnicas criptográficas lidam essencialmente com domínios de informação finitos. Isto significa que a representatividade matemática dos items de informação vai ser realizada por domínios discretos de informação.

Domínios discretos e fintos são representáveis, em último caso, por conjuntos de *strings* de bits. Um outra representação possível assenta nos inteiros.

Ao longo deste capítulo procuraremos caracterizar alguns dos domínios matemáticos que têm estas características e que são amplamente usados nas técnicas criptográficas.

2.1 Divisibilidade

$\mathbb{Z}$ – conjunto dos inteiros com a estrutura algébrica de um **anel** com adição $+$ e multiplicação $\times$.

$\mathbb{Z}_n$ – conjunto dos inteiros $0, 1, \ldots, (n-1)$ com a estrutura de um **anel** com a adição $+$ e multiplicação efectuadas módulo n

EXEMPLO 12: $\mathbb{Z}_5$

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\times$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

$\mathbb{Z}_2 = \{0, 1\}$ é um corpo particularmente importante pois representa a estrutura de uma álgebra booleana; as tabelas de adição e multiplicação são:

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

Note-se que a multiplicação é equivalente à conjunção lógica `and` e a adição é equivalente à disjunção exclusiva `xor`.

□

$\mathbb{Z}_n^*$ – **grupo multiplicativo** formado por todos os elementos invertíveis de $\mathbb{Z}_n$; i.e., elementos $a \in \mathbb{Z}_n$ para os quais existe $b \in \mathbb{Z}_n$ tal que $a \times b = 1 \pmod{n}$.

126 FACTO
$a \in \mathbb{Z}_n$ *é invertível em* $\mathbb{Z}_n$ *se e só se* $\gcd(a, n) = 1$

Notas

- Sendo p primo todos os elementos de $\mathbb{Z}_p$, excepto 0, são invertíveis.
- $\mathbb{Z}_2^*$ é o conjunto singular $\{1\}$ e representa o grupo multiplicativo mais simples (reduz-se à unidade).
- $\mathbb{Z}_9^* \equiv \{\, 1\,,\, 2\,,\, 4\,,\, 5\,,\, 7\,,\, 8\,\}$ com $2^{-1} = 5$, $4^{-1} = 7$ e $8^{-1} = 8$.

A **ordem** de um grupo $\langle \mathcal{G}, \cdot \rangle$ é o número de elementos do grupo e representa-se por $|\mathcal{G}|$.

O grupo é **cíclico**, quando existe um elemento g (dito **gerador** do grupo) tal que, $\forall x \in \mathcal{G}$ existe um inteiro positivo $n \in \mathbb{N}$ tal que $x = g \cdot g \cdot g \ldots \cdot g$ (n vezes). [25]

[25] Quando o grupo é aditivo é costume representar $g \cdot g \cdot \ldots \cdot g$ (n vezes) por $n\,g$ e chama-se a esta operação, o **produto escalar discreto**. Quando o grupo é multiplicativo a mesma expressão representa-se por g^n e chama-se **exponenciação discreta**.

127 Facto

Se $\mathcal{G}$ é um grupo cíclico finito de gerador g, então

$$n = m \pmod{|\mathcal{G}|} \quad \Leftrightarrow \quad g^n = g^m$$

Exemplo 13:

$\mathbb{Z}_9^*$ é um **grupo cíclico** de **ordem** 6.
2 e 5 são **geradores do grupo**
4 e 7 geram **subgrupos de ordem** 3

Exemplos de $g^i \pmod 9$

i	0	1	2	3	4	5
$g = 2$	1	2	4	8	7	5
$g = 4$	1	4	7	1	4	7
$g = 5$	1	5	7	8	4	2
$g = 7$	1	7	4	1	7	4

2.2 Resultados Fundamentais da Divisibilidade

128 Teorema (fundamental da aritmética)
Cada $m > 1$ admite uma única **factorização**

$$m = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_k^{e_k}$$

em que $1 < p_1 < p_2 < \cdots < p_k$ são primos.

129 Teorema (pequeno de Fermat)
Se p é primo então, para todo $a \geq 0$, verifica-se $a^p = a \pmod p$.

130 Corolário *Se p é primo, $a \in \mathbb{Z}_p^*$ e $k \geq 0$, então $a^{p-k-1} = a^{-k}$ em $\mathbb{Z}_p^*$.*

Se a^{-1} existe em $\mathbb{Z}_p^*$ então, tomando congruências $\pmod p$, tem-se $a^{-k} = a \cdot a^{-k-1} = a^p \cdot a^{-k-1} = a^{p-k-1}$.

131 Corolário *Se p é primo e k é um qualquer divisor de $(p-2)$ então a função $\alpha_k : x \mapsto x^k \pmod p$ é um automorfismo no grupo multiplicativo $\mathbb{Z}_p^*$.*

Claramente α_k preserva a a estrutura do grupo multiplicativo $\mathbb{Z}^*$. Basta provar, portanto, que é um isomorfismo.

Tomando sempre congruências $\pmod p$, suponhamos que existiam $x, y \in \mathbb{Z}_p^*$ tais que $x^k = y^k$; sendo $(p-2)$ um múltiplo de k será também $x^{(p-2)} = y^{(p-2)}$; pelo facto anterior, $x^{-1} = x^{p-2}$ e $y^{-1} = y^{p-2}$; conclui-se que $x^{-1} = y^{-1}$ e, portanto, $x = y$.

132 DEFINIÇÃO

A ***função de Euler-phi****, representada por $\phi(\cdot)$, associa a cada $m > 1$ o número de inteiros $0 \leq a < m$ tais que $\gcd(a, m) = 1$.*

PROPRIEDADES DA FUNÇÃO DE EULER-PHI

- $\phi(m) = \prod_{i=1}^{k} p_i^{(e_i-1)} \times (p_i - 1)$
- $a^{\phi(m)} = 1 \pmod m$ (*Teorema de Euler*)
- Seja $m = p * q$ o produto de dois primos distintos e $\varphi = \phi(m)/2$

$$i = j \pmod \varphi \Longrightarrow x^i = x^j \pmod m$$

Em particular (*Teorema do RSA*)

$$a = b^{-1} \pmod{\varphi} \implies (x^a)^b = x \pmod{m}$$

- $m = \sum_{a|m} \phi(a)$ ($a|m$ representa a asserção "a divide m").

Note-se que o teorema RSA continua a verificar-se sempre que φ for um múltiplo comum de $(p-1)$ e $(q-1)$.

Nomeadamente quando $\varphi = \phi(m) = (p-1) * (q-1)$ e, ainda, quando $\varphi = \text{lcm}(p-1, q-1) = (p-1) * (q-1)/\gcd(p-1, q-1)$.

EXEMPLO 14:

- $\phi(12) = \phi(2^2 \times 3) = 2^1 \times (2-1) \times (3-1) = 4$
- $5^4 = 625 = 52 \times 12 + 1 = 1 \pmod{12}$
- $15 = 3 \times 5 \qquad \phi(15) = 8 \qquad 1025 = 1 \pmod{8}$. Logo $a^{1025} = a \pmod{15}$.
- $\phi(1) + \phi(2) + \phi(3) + \phi(4) + \phi(6) + \phi(12) = 1 + 1 + 2 + 2 + 2 + 4 = 12$

133 DEFINIÇÃO
O menor $t > 0$ *tal que* $a^t = 1 \pmod{m}$ *é a* ***ordem*** *do elemento* $a \in \mathbb{Z}_m^*$. *Se* $t \equiv \phi(m)$, a *diz-se* ***elemento primitivo*** *de* $\mathbb{Z}_m^*$

134 Facto

(a) *Se* $a^s = 1 \pmod{m}$ *então* s *é um múltiplo da ordem de* a.

(b) $\mathbb{Z}_m^*$ *tem um elemento primitivo se e só se* $m = 2$, 4 , p^n *ou* $2p^n$, *sendo* p *um primo ímpar.*

(c) *Se* $\mathbb{Z}_m^*$ *tem um elemento primitivo então é um grupo cíclico em que cada um dos seus elementos primitivos é um gerador do grupo.*

(d) *Se* $a \in \mathbb{Z}_m^*$ *é um elemento primitivo então qualquer outro* $b \in \mathbb{Z}_n^*$ *é um elemento primitivo se e só se tiver a forma* $a^k \pmod{m}$ *com* $k \in \mathbb{Z}_{\phi(m)}$. *Donde, se* $\mathbb{Z}_m^*$ *tiver algum elemento primitivo, terá* $\phi(\phi(m))$ *elementos primitivos distintos.*

$\mathbb{Z}_{21}^*$ não é cíclico. Porém $\mathbb{Z}_{22}^*$ e $\mathbb{Z}_{23}^*$ são ambos cíclicos. De facto $\mathbb{Z}_{21}^* = \{1, 2, 4, 5, 8, 10, 11, 13, 16, 17, 19, 20\}$ não tem nenhum elemento de ordem superior a 6 (note-se que $\phi(21) = 12$).

2.3 Raízes Quadradas

Um **resíduo quadrático** em $\mathbb{Z}_n$ é um número $a \neq 0$ para o qual existe $x \in \mathbb{Z}_n$ tal que

$$a = x^2 \pmod{m}$$

Nestas circunstâncias, o valor x é uma **raíz quadrada modular** de a.

135 Definição
*Seja $p > 2$ um primo. O **símbolo de Legendre** $\left(\frac{a}{p}\right)$ é definido como sendo 0 se $p|a$, é igual a 1 se a é um resíduo quadrático módulo p e é igual a (-1) em caso contrário.*

*Se $m > 1$ tem uma factorização $p_1 p_2 \ldots p_n$ por primos não necessariamente distintos então o **símbolo de Jacobi** $\left(\frac{a}{m}\right)$ define-se como $\Pi_{i=1}^{n} \left(\frac{a}{p_i}\right)$.*

O valor do símbolo de Jacobi $\left(\frac{a}{m}\right)$ é 0 se e só $\gcd(a, m) \neq 1$. Se $a \in \mathbb{Z}_n^*$ o símbolo $\left(\frac{a}{m}\right)$ é sempre ± 1, com igual probabilidade (excepto quando m é uma quadrado onde o valor é sempre 1).

Existe um algoritmo bastante eficiente para determinar símbolos de Legendre e de Jacobi (sem recorrer à factorização de m).

136 FACTO

Se p é primo então, para todo a

$$\left(\frac{a}{p}\right) = a^{(p-1)/2} \pmod{p}$$

Se a é um resíduo quadrático (i.e. $a^{(p-1)/2} = 1 \pmod{p}$) tem duas raízes quadradas modulares x calculadas da seguinte forma[26]*:*

- *Se $(p+1)$ é divisível por 4 então $x = \pm a^{(p+1)/4} \pmod{p}$.*
- *Se $(p-5)$ é divisível por 8 e se for $k \doteq (p-5)/8$ então*

$$x = \pm a\, y\, (2\, a\, y^2 - 1) \pmod{p}$$

  sendo $y \doteq (2\, a)^k \pmod{p}$.
- *Se $p = 2^t\, q + 1$, com q ímpar e $t > 2$ então*

$$x = a^{(q+1)/2} \cdot (u^q)^s \pmod{p}$$

  com u um qualquer não-resíduo quadrático módulo p e $s < 2^{t-1}$, um expoente encontrado por tentativas.

[26] Neste caso, $a^{(p+1)/2} = a^{(p+1)/2} * a^{(p-1)/2} = a^p = a \pmod{p}$.

Existe um algoritmo para encontrar s com complexidade $\mathcal{O}(t)$ baseado na sua expansão em bits e no seguinte facto:

137 FACTO

Se for

$$s = s_0 + 2 \cdot s_1 + 2^2 \cdot s_2 + \ldots + 2^{t-2} \cdot s_{t-2} \qquad s_i \in \mathbb{Z}_2$$

então verifica-se

$$\begin{aligned}
a^{(p-1)/4} \cdot (u^{(p-1)/2})^{s_0} &\equiv 1 \pmod{p} \\
a^{(p-1)/8} \cdot (u^{(p-1)/4})^{s_0} \cdot (u^{(p-1)/2})^{s_1} &\equiv 1 \pmod{p} \\
\ldots &\equiv \ldots \\
a^{(p-1)/2^t} \cdot (u^{(p-1)/2^{t-1}})^{s_0} \cdot \ldots \cdot (u^{(p-1)/2})^{s_{t-2}} &\equiv 1 \pmod{p}
\end{aligned}$$

A 1ª equação determina s_0, a segunda determina s_1, etc. . .

2.4 Algoritmos

Algoritmo de Euclides

Determina $\gcd(a, b)$ quando $a \geq b > 0$.

```
enquanto b > 0  { r = a mod b ; a = b ; b = r }
```

Existe uma versão extendida que, quando $\gcd(a, b) = 1$, determina o inverso de a módulo b.

Estes algoritmos extendem-se a qualquer anel comutativo, nomeadamente aos anéis de polinómios.

Teorema Chinês dos Restos

Se $N = n_1 \times n_2 \times \cdots \times n_k$ é um produto de números primos entre si, existe um único $x < N$ que é solução do sistema de equações

$$x = a_1 \pmod{n_1}, \quad x = a_2 \pmod{n_2}, \quad \cdots \quad x = a_k \pmod{n_k}$$

dados os "restos" $0 < a_i < n_i$, com $i = 1 \ldots k$.

$$x = \sum_{i=1}^{k} a_i \, x_i \, N_i \pmod{N} \qquad \text{com} \quad N_i = N/n_i, \quad x_i = N_i^{-1} \pmod{n_i}$$

Cálculo eficiente de exponenciais

Para um qualquer grupo multiplicativo $\langle G\ ,\ \cdot\rangle$ calcula $x = a^b$, com $a \in G$ e um inteiro $b \geq 0$, a partir da representação binária de b

Seja $b = b_1 + 2b_2 + 2^2 b_3 + \cdots + 2^{n-1} b_n$ com $b_i \in \mathbb{Z}_2$.

```
x = 1 ;
para i = n até 1 { x = x * x; se b[i] { x = x * a } }
```

O número de multiplicações está limitado ao dobro do número de bits necessários para representar b

Teste e Geração de Primos

Primos são essenciais em várias técnicas criptográficas. Distribuem-se de modo basicamente uniforme; para cada $n > 2$, o número de primos menores ou iguais a n tende para $\frac{n}{\ln n}$

Os algoritmos mais eficientes para **geração** de grandes primos são baseados no algoritmo

```
repetir { gerar um número aleatório P }
at\'e { é-primo?(P) = verdadeiro }
```

Donde assentam em duas operações básicas: **geração** de números aleatórios e **teste** da propriedade "*ser primo*".

Normalmente um teste determinístico é computacionalmente intratável. Por isso usam-se testes não determinísticos que se baseiam na existência, para cada $p \gg 3$ de conjuntos $W(p) \subset \mathbb{Z}_p$ com a propriedade

(i) $|W(p)| = 0$ se p é primo e $|W(p)| \geq p/2$ se p não é primo.
(ii) Para cada $a \in \mathbb{Z}_p$, o teste $a \in W(p)$ é computacionalmente tratável.

Algoritmo de teste

1. Escolhe-se um número $a \in \mathbb{Z}_p$ aleatoriamente; se ocorrer $a \in W(p)$ então, com probabilidade 1, p não é primo. O algoritmo termina com a resposta **não** e sem erro.
   Se $a \in \overline{W(p)}$ então p pode ou não ser primo; porque $|W(p)| \geq |\overline{W(p)}|$ a probabilidade de ser primo é pelo menos igual à probabilidade de não ser.
2. Repete-se (1) até se atingir um **limite de tentativas** t ou aí terminar.
3. Se o limite de tentativas t for alcançado a resposta é **sim** e tem uma probabilidade de erro inferior a 2^{-t}.

Critério de Fermat Se $p > 2$ é primo, $a^{p-1} = 1 \pmod p$ para todo $0 < a < p$.

$$W(p) \equiv \{0 < a < p \mid a^{p-1} \neq 1 \pmod p\}$$

Infelizmente $W(p)$ pode ser vazio mesmo quando p não é primo; os chamados **números de Carmichael** satisfazem essa propriedade.

Critério de Euler Se $p > 2$ é primo, $a^{(p-1)/2} = \left(\frac{a}{p}\right) \pmod p$ para todo $0 < a < p$.

$$W(p) \equiv \{0 < a < p \mid a^{(p-1)/2} \neq \left(\frac{a}{p}\right) \pmod p\}$$

Nota Se p não é primo, formalmente o símbolo de Legendre não existe; existe o **símbolo de Jacobi** que é o produto dos símbolos de Legendre de a em relação a cada um dos factores primos de p. O algoritmo para calcular $\left(\frac{a}{p}\right)$ é, no entanto semelhante, e muito eficiente (ver `Koblitz`).

O *algoritmo de Solovay-Strassen* implementa este critério. Tem vindo a ser substituído pelo *algoritmo de Miller-Rabin* baseado em

Critério de Miller-Rabin Se p é primo e $q = 2^{-s}(p-1)$ é o maior divisor ímpar de $(p-1)$ então, para todo $0 < a < p$, uma de duas situações ocorre

$$a^q = 1 \pmod p \quad \textbf{ou}$$

$$a^{2^i q} = -1 \pmod p \quad \text{para algum} \quad 0 \leq i < s$$

Os conjuntos $W(p)$ definem-se apropriadamente a partir deste critério.

Ver, no Handbook of Applied Criptography, detalhes, comparações e implementações destes algoritmos.

2.5 Corpos Finitos

Um **corpo finito** é um anel finito e comutativo em que todos os elementos, excepto o 0, têm inversa multiplicativa.

A **característica** de um corpo finito é o menor inteiro $v > 0$ tal que $\overbrace{1+1+\cdots+1}^{v \text{ vezes}} = 0$.

$\mathbb{Z}_p$, para p primo, é um **corpo finito** de característica p; note-se que, neste caso, $\mathbb{Z}_p \setminus \{0\} \equiv \mathbb{Z}_p^*$

138 FACTO

Seja $\mathbb{F}_m$ um corpo finito com m elementos.

(a) O conjunto $\mathbb{F}_m$ é identificável com o conjunto das raízes do polinómio $X^m - X \in \mathbb{F}_m[X]$, numa extensão do corpo $\mathbb{F}_m$

(b) Se $\mathbb{F}_m$ tem característica $p \neq 0$ então p é primo e existe uma dimensão n tal que $m = p^n$ e

$$\mathbb{F}_m \sim (\mathbb{Z}_p)^n$$

(c) Existe um ***polinómio primitivo***[27] $\mathbf{c} \in \mathbb{Z}_p[X]$ *de grau n tal que*

$$\mathbb{F}_m \cong \mathbb{Z}_p[X]/\mathbf{c}\mathbb{Z}_p[X]$$

[27] Um polinómio em $\mathbb{Z}_p[X]$, mónico e irreduzível, é **primitivo** em $\mathbb{F}_m$ se divide $X^m - X$ mas não divide nenhum $X^k - X$ para $k < m$.

Nota: $\mathbb{Z}_p[X]/\mathbf{c}\mathbb{Z}_p[X]$ *é o anel dos polinómios de coeficientes em* $\mathbb{Z}_p$ *em que a adição e multiplicação de polinómios é efectuada módulo* $\mathbf{c}(X)$.

(d) *O grupo multiplicativo* $\mathbb{F}_m^*$ *, formado pelos elementos invertíveis de* $\mathbb{F}_m$*, é cíclico e existem* $\phi(m-1)$ *geradores diferentes deste grupo.*

(e) *Se* $\mathbf{c} \in \mathbb{Z}_p[X]$ *é um polinómio primitivo seja* K *uma qualquer extensão de* $\mathbb{Z}_p$ *(i.e.,* K *contém* $\mathbb{Z}_p$ *como sub-corpo) que contém todas as raízes desse polinómio*[28]*; seja* β *uma raíz de* $\mathbf{c}$ *em* K *que não seja raíz de nenhum polinómio* $(X^k - 1)$ *com* $k < m - 1$ [29]*.*

Então:

(i) *O menor anel que contém* $\mathbb{Z}_p$ *e o elemento* β *forma um sub-corpo de* K *que representamos por* $\mathbb{Z}_p(\beta)$ *; sendo* $n = \text{grau}(\mathbf{c})$ *, o elemento genérico desse corpo tem a forma*

$$a_0 + a_1\,\beta + a_2\,\beta^2 + \cdots + a_{n-1}\,\beta^{n-1} \quad \text{com} \quad a_i \in \mathbb{Z}_p \tag{49}$$

(ii) *A menos de um isomorfismo* $\mathbb{F}_m$ *identifica-se com* $\mathbb{Z}_p(\beta)$ *.*

(f) *Os elementos* β , β^p , β^{p^2} , $\cdots$, $\beta^{p^{n-1}}$ *estão contidos* $\mathbb{Z}_p(\beta)$ *e formam o conjunto das* n *raízes do polinómio* $\mathbf{c}$ *.*

Comentários

[28] Diz-se que K é um corpo que **fractura** o polinómio $\mathbf{c}[X]$ sobre $\mathbb{F}_p$.

[29] Estas raízes designam-se por **raízes primitivas**.

(a) Este facto define a representação essencial para os elementos de um corpo finito. O polinómio $X\,(X^{m-1}-1)$ tem m raízes: o 0 e as $m-1$ soluções distintas da equação $X^{m-1}=1$ (designadas **raízes da unidade**).

O resultado é uma simples consequência do facto: se x e y verificam $x^m = x$ e $y^m = y$ então, em $\mathbb{F}_m$, necessariamente se verifica $(x+y)^m = (x+y)$ e $(x\,y)^m = x^m\,y^m$.

Esta representação é muito importante sob o ponto de vista da análise das propriedades algébricas de $\mathbb{F}_m$, mas não é muito útil em termos de manipulação. Por isso são necessárias outras representações.

(b) Este facto significa que cada elemento $x \in \mathbb{F}_m$ é representável por um vector de n componentes em que todas essas componentes são elementos de $\mathbb{Z}_p$.

Em caso particular muito importante ocorre quando $p=2$. Quando a característica é 2 o corpo finito diz-se binário e cada um dos seus 2^n elementos é representável por um vector de n componentes em $\mathbb{Z}_2$; i.e., uma palavra de n bits.

Esta 2ª representação já permite uma forma simples de somar elementos de $\mathbb{F}_m$: para somar dois elementos $x, y \in \mathbb{F}_m$ basta somar os respectivos vectores componente a componentes.

A multiplicação destes elementos já não pode ser feita nesta representação vectorial porque não está, genericamente, definida a multiplicação de vectores.

(c) A procura dos polinómios primitivos $\mathbf{c}[X]$ é muito importante porque permite expressar um corpo finito como um corpo de polinómios $\mathbb{F}_p[X]/\mathbf{c}$.

As raízes de um polinómio primitivo devem estar associadas ao elemento 0 de $\mathbb{F}_m$; por isso o polinómio primitivo tem de dividir $X^m - X$. Por outro lado não pode dividir nenhum outro polinómio da forma $X^k - X$, com $k < m$, porque neste caso o espaço quociente teria menos elementos distintos dos que os m exigidos pelo corpo $\mathbb{F}_m$.

Esta 3ª representação já permite fazer multiplicações. As n componentes do vector são agora interpretadas como coeficientes de um polinómio.

Este facto diz essencialmente que existe um polinómio de grau n, $\mathbf{c} \in \mathbb{Z}_p[X]$ tal que para multiplicar $x, y \in \mathbb{F}_m$ basta multiplicar os dois polinómios que os representam e reduzir o resultado módulo $\mathbf{c}$.

(d) Sendo $\mathbb{F}_m$ um corpo todos os seus elementos, excepto 0, são invertíveis. Portanto o grupo multiplicativo $\mathbb{F}_m^*$ tem $m-1$ elementos. Adicionalmente o grupo é cíclico: todo o elemento de $\mathbb{F}_m$, excepto o 0, tem a forma g^i com $i \in \mathbb{Z}_{m-1}$.

Os elementos do grupo cíclico são as raízes de $X^{m-1}-1$ na extensão de $\mathbb{F}_m$ que as contém.

Como elemento de corpo quociente de polinómios $\mathbb{Z}_p[X]/\mathbf{c}\mathbb{Z}_p[X]$ o monómio X é um gerador uma vez que, reduzido módulo $\mathbf{c}$ todas as potências X^k, com $k < m$, têm de ser distintas; de facto, se fosse $X^k = X^i \pmod{c}$, então $X^k - X^i$ seria divisível por $\mathbf{c}[\mathbf{X}]$ o que contraria a definição de polinómio primitivo.

(e) A representação $\mathbb{Z}_p(\beta)$ é, geralmente, a representação preferida de $\mathbb{F}_m$. Porque β é raiz de um polinómio de grau n com coeficientes em $\mathbb{Z}_p$, é sempre possível escrever $\beta^n = c_0 + c_1\,\beta + \cdots + c_{n-1}\,\beta^{n-1}$, com $c_i \in \mathbb{Z}_p$; em qualquer multiplicação de dois elementos da forma (49), sempre que o ocorra um termo da forma β^k com $k \geq n$, pode-se fazer substituições sucessivas de β^n de acordo com esta igualdade de forma a reduzir o resultado, sempre, a um expressão da forma (49).

Os geradores de $\mathbb{F}_m \sim \mathbb{Z}_p(\beta)$ podem ser determinados a partir de β. De facto o elemento β é um gerador de $\mathbb{Z}_p(\beta)^*$: qualquer $x \neq 0$ pode ser escrito como $x = \beta^i$ com $i \in 0..p^n - 1$.

Isto facilita o cálculo de multiplicações ($\beta^i \cdot \beta^j = \beta^{i+j}$) mas dificulta o cálculo das somas.

Seja $\tau : \mathbb{Z}_{m-1} \rightharpoonup \mathbb{Z}_{m-1}$ a função parcial que associa cada $i \in \mathbb{Z}_{m-1}$, tal que $\beta^i + 1 \neq 0$, ao índice $\tau(i)$ que verifica $\beta^{\tau(i)} = \beta^i + 1$. Esta função chama-se o *logaritmo de Zech* de base β; então

$$\beta^i + \beta^j \;=\; (\beta^{i-j} + 1)\,\beta^j \;=\; \beta^{\tau(i-j)+j}$$

Isto dá uma forma "imediata" de calcular somas de potências $(\beta^i + \beta^j)$ quando $\tau(i-j)$ é definido; se não for definido é porque o resultado da soma é zero.

Obviamente que isto presupõe o cálculo *à priori* do logaritmo de Zech. O que não é, geralmente, uma tarefa simples.

(f) É muito simples provar que, para quaisquer dois elementos $x, y \in \mathbb{Z}_p(\beta)$, se verifica sempre $(x+y)^p = x^p + y^p$ e

$(x \cdot y)^p = x^p \cdot y^p$; basta fazer a expansão apropriada e notar que, para todo $a \in \mathbb{Z}_p$, $a^p = a$. Deste modo, para todo o polinómio $\mathbf{p}[X] \in \mathbb{Z}_p[X]$, tem-se $(\mathbf{p}(x))^p = \mathbf{p}(x^p)$.
Por isso, se $\alpha \in \mathbb{Z}_p(\beta)$ é raiz de um qualquer polinómio $\mathbf{p}[X] \in \mathbb{Z}_p[X]$, teremos $\mathbf{p}(\alpha^p) = (\mathbf{p}(\alpha))^p = 0$. Logo α^p também será raiz do mesmo polinómio.

Por este processo, a partir da raiz β, gera-se $\beta^p, \beta^{p^2}, \cdots, \beta^{p^{n-1}}$ todas raizes de $\mathbf{c}$; uma vez que os diferentes expoentes p^i, com $i \in 0..n-1$ são todos distinctos em $\mathbb{Z}_{p^n-1}$, e β é um gerador do grupo cíclico $\mathbb{Z}_p(\beta)^*$, todas as raízes β^{p^i} serão distintas.
Por isso $\mathbb{Z}_p(\beta)$ contém, não só a raiz β, como também todas as restantes raízes do polinómio primitivo $\mathbf{c}[X]$. Deste modo $\mathbb{Z}_p(\beta)$ também fratura o polinómio sobre $\mathbb{Z}_p$. De facto, é o menor corpo (a menos de um isomorfismo) que verifica esta propriedade e, por isso, se chama o *corpo mínimo de fratura* de $\mathbf{c}[X]$ sobre $\mathbb{Z}_p$.

EXEMPLO 15: Considere-se o corpo $\mathbb{F}_9$. Como representá-lo?

Dado que $9 = 3^2$ a característica do corpo é 3 e a dimensão é 2. Isto significa que o polinómio característico é um monómio de grau 2 com coeficientes em $\mathbb{Z}_3$.

Os polinómios primitivos de $\mathbb{F}_9$ serão monómios de grau 2 em $\mathbb{Z}_3[X]$, irredutíveis, que dividem $X^9 - X$ mas não dividem nenhum outro $X^k - X$ com $k < 9$.

Usando o sistema Pari (ou qualquer sistema de computação análogo) encontram-se os seguintes factores de

$X^9 - X$ que são irredutíveis e têm grau 2.

$$(X^2 + 1) \qquad (X^2 + X - 1) \qquad (X^2 - X - 1)$$

Note-se que, em $\mathbb{Z}_3$, -1 é equivalente a 2. Neste corpo Em $\mathbb{Z}_3$ o polinómio $(X^2 + 1)$ divide $(X^5 - X)$; de facto $(X^5 - X) = X \cdot (X^4 - 1) = X \cdot (X^2 - 1) \cdot (X^2 + 1)$; isso exclui-o de ser um polinómio primitivo.

Restam os polinómios $(X^2 + X - 1)$ e $(X^2 - X - 1)$. Usando `Pari` de novo, verifica-se que nenhum deles divide qualquer polinómio $(X^k - X)$, com $k \in 2..8$. Portanto ambos os polinómios são primitivos e qualquer um pode ser usado como polinómio característico.

Tomemos, por exemplo, $X^2 - X - 1$ como característico[30]; a partir daqui existem duas representações possíveis para os elementos de $\mathbb{F}_9$

1. Tomemos uma extensão K qualquer de $\mathbb{Z}_3$ que contenha todas as raízes do polinómio característico $X^2 - X - 1$. Tomemos uma qualquer raíz β desse polinómio. Como consequências imediatas, β verifica $\beta^2 = \beta + 1$ e a outra raíz do polinómio é $1 - \beta$.

[30] Por curiosidade, o número irracional $(1 + \sqrt{5})/2$, que é a raiz real e positiva deste polinómio, é um dos números mais famosos da história da Matemática; é chamado *razão áurea* ou *número de ouro* ou *golden rule*, ou ainda várias outras designações análogas. É considerado a proporção ideal para a harmonia de dimensões de edifícios, dimensões de instrumentos musicais, progressão de escalas musicais, etc.

O sub-anel $\mathbb{Z}_3(\beta)$ de K, formado por todos os elementos da forma $a + b\,\beta$, é um corpo que verifica: $(a + b\beta) + (c + d\beta) = (a + c) + (b + d)\beta$ e $(a + b\beta)\,(c + d\beta) = (ac + bd) + (ad + bc + bd)\beta$.

2. A segunda representação é polinomial. Cada elemento de $\mathbb{F}_9$ é descrito por um polinómio $(x + y\,X) \in \mathbb{Z}_3[X]/\mathbf{c}, \mathbb{Z}_3[X]$ em que $\mathbf{c}$ denota o polinómio característico $X^2 - X - 1$.
   As operações de soma e multiplicação são efectuadas como em quaisquer polinómios tendo em atenção, apenas, que operações nos coeficientes são sempre efectuadas em $\mathbb{Z}_3$ e que os polinómios de grau superior a 2 são reduzidos módulo o polinómio característico.

Obviamente que as duas representações são isomórficas: cada uma delas se converte na outra de uma forma única. No entanto é preciso constatar que lida com entidades diferentes: polinómios no segundo caso e extensões algébricas no primeiro caso.

Considere-se a representação de um elemento genérico $x \in \mathbb{F}_9$ por um elemento de $\mathbb{Z}_3(\beta)$.

A menos de um isomorfismo o corpo $\mathbb{Z}_3(\beta)$ é único e os seus elementos são

$$\{0\ ,\ -1\ ,\ 1\ ,\ \beta\ ,\ -\beta\ ,\ 1+\beta\ ,\ 1-\beta\ ,\ -1+\beta\ ,\ -1-\beta\}$$

Todo $x \in \mathbb{F}_9$, excepto 0, é invertível; também $\mathbb{F}_9^*$ é cíclico de gerador β; isto significa todo $x \in \mathbb{F}_9^*$ é escrito na forma $x = \beta^k$ para algum $k \in \mathbb{Z}_8$.

Usando Pari pode-se calcular uma tabela das potências β^k e verificar

β^0	β^1	β^2	β^3	β^4	β^5	β^6	β^7
1	β	$1+\beta$	$1-\beta$	-1	$-\beta$	$-1-\beta$	$-1+\beta$

Para calcular o inverso de um elemento pode-se usar esta tabela; por exemplo (note-se que $6 = -2 \pmod 8$)

$$(1+\beta)^{-1} = (\beta^2)^{-1} = \beta^6 = -1-\beta$$

O logaritmo de Zech pode ser calculado usando a tabela anterior e tem-se

k	0	1	2	3	4	5	6	7
$\tau(k)$	4	2	7	6	$\perp$	3	5	1

Por exemplo, para calcular

- $\beta^5 + \beta^3 = \beta^{\tau(5-3)+3} = \beta^{7+3} = \beta^2$ porque os expoentes são vistos $\pmod 8$.
- $\beta^6 + \beta^2 = 0$ porque $\tau(6-2) = \perp$.

Exemplo 16: Considere-se agora a representação de $\mathbb{F}_{256}$.

Como $256 = 2^8$ temos um corpo de característica 2 isomórfico com $(\mathbb{Z}_2)^8$: os elementos de $\mathbb{F}_{256}$ devem, assim, ser representados por uma palavra de 8 *bits* (um "byte").

Usando `Pari` é possível determinar todos os polinómios irredutíveis de grau 8 que são factores de $X^{255}+1$ (note-se que, com característica 2, tem-se $X = -X$). Desses polinómios selecionam-se aqueles que não dividem nenhum $X^k + 1$, com $k < 255$.

Por este processo verifica-se que não existe nenhum polinómio primitivo com menos de 5 coeficientes não-nulos. Um desses polinómios é $(X^8 + X^4 + X^3 + X + 1)$ que pode ser usado como polinómio característico.

A estrutura de um corpo finito $\mathbb{F}_m$ é também determinada pelos seus automorfismos; isto é, as aplicações $\mathbb{F}_m \longrightarrow \mathbb{F}_m$ que preservam a estrutura do corpo (endomorfismos) e são injectivas.

Os automorfismos de $\mathbb{F}_m$ que fixam[31] os elementos de $\mathbb{F}_p$ formam o **grupo de Galois** $\mathrm{Gal}(\mathbb{F}_m/\mathbb{F}_p)$. A operação de grupo é a composição de morfismos com a identidade 1. Considera-se o grupo multiplicativo: $\sigma \cdot \delta$ é o morfismo $x \mapsto \sigma(\delta(x))$, $\sigma^0 = \mathbf{1}$ e σ^k é $\underbrace{\sigma \cdot \sigma \cdots \sigma}_{k \text{ vezes}}$.

139 FACTO

Seja $p \neq 0$ a característica de $\mathbb{F}_m$. Para todos $x, y \in \mathbb{F}_m$, $(x+y)^p = x^p + y^p$, $(x \cdot y)^p = x^p \cdot x^p$, $x^p = 0$ se e só se $x = 0$ e $x^p = x$ se e só se $x \in \mathbb{F}_p$.

A aplicação $\sigma : x \mapsto x^p$ designa-se por **morfismo de Frobenius** e este resultado afirma que se trata de um automorfismo em $\mathbb{F}_m$ que discrimina os elementos de $\mathbb{F}_p$ por serem os que são fixos por σ.

Comentários

1. O morfismo de Frobenius fixa os elementos de $\mathbb{F}_p$ contidos em $\mathbb{F}_m$; de facto, pelo pequeno teorema de Fermat, $x^p = x$ para todo $x \in \mathbb{F}_p$. Por outro lado, sabemos que $\mathbb{F}_p$ se identifica com o conjunto das raízes numa sua extensão (como é o caso de $\mathbb{F}_m$) do polinómio $X^p - X$. Portanto qualquer elemento de $\mathbb{F}_m$ que seja fixo por σ se identifica com um elemento de $\mathbb{F}_p$.

[31] O morfismo σ fixa x quando $\sigma(x) = x$.

2. Usando a noção de característica é muito simples mostrar que σ é realmente um endomorfismo; isto é, $\sigma(x+y) = (x+y)^p = x^p + y^p = \sigma(x) + \sigma(y)$ e $\sigma(x \cdot y) = (x \cdot y)^p = x^p \cdot y^p = \sigma(x) \cdot \sigma(y)$.

3. Como primeira consequência, tem-se que, para todo o polinómio $\mathbf{p}[X] \in \mathbb{F}_p[X]$ se verifica $(\mathbf{p}[X])^p = \mathbf{p}[X^p]$.

   De facto, se for $\mathbf{p}[X] = p_0 + p_1 X + p_2 X^2 + \cdots + p_d X^d$ então

$$(\mathbf{p}[X])^p = \quad a_0^p + a_1^p X^p + a_2^p (X^p)^2 \cdots + a_d^p (X^p)^d =$$

$$= \quad a_0 + a_1 X^p + a_2 (X^p)^2 \cdots + a_d (X^p)^d = \quad \mathbf{p}[X^p]$$

   já que, por serem elementos de $\mathbb{F}_p$, todos os a_i verificam $a_i^p = a_i$.

4. Como corolário da observação anterior, se α é uma raiz do polnómio $\mathbf{p}[X]$ numa qualquer extensão de $\mathbb{F}_p$, então α^p será também uma raiz.

$$\mathbf{p}[\alpha] = 0 \quad \Longrightarrow \quad \mathbf{p}[\alpha^p] = (\mathbf{p}[\alpha])^p = 0$$

5. Para verificar-mos que o morfismo de Frobenius σ é um isomorfismo, dado que é linear, basta verificar que não existe nenhum $x \neq 0$ que verifique $\sigma(x) = 0$.

   Tomando a representação polinomial genérica, definida em (49), para um elemento $x \in \mathbb{F}_m$; pelas observações anteriores

$$x^p \;=\; a_0 + a_1\, \beta^p + a_2\, (\beta^p)^2 + \cdots + a_{n-1}\, (\beta^p)^{n-1}$$

   Sendo β uma raiz do polinómio característico, β^p é também uma raiz do mesmo polinómio. Portanto o morfismo de Frobenius toma um elemento $x \in \mathbb{F}_m$ e limita-se a produzir uma mudança de base: produz a soma formal com os mesmos coeficientes mas efectuada com uma raiz diferente do polinómio característico. Se fosse $x^p = 0$ todas as componentes a_i teriam de ser nulas e, assim, seria também $x = 0$.

Um dos resultados mais importantes do estudo dos corpos finitos pode ser enunciado da forma seguinte

140 FACTO
Se $L \simeq \mathbb{F}_m$ e $K \simeq \mathbb{F}_q$ são dois corpos finitos com a mesma característica p (com $m = p^n$ e $q = p^r$) então L é uma extensão de K se e só se existe $s \geq 1$ tal que $n = s\,r$.

Nestas circunstâncias o grupo de Galois $\mathrm{Gal}(L/K)$ é cíclico, tem ordem s e é gerado pelo morfismo de Frobenius $\sigma : x \mapsto x^q$ de L em K.

Comentários

Recordemos que o grupo de Galois $\mathrm{Gal}(L/K)$ é formado por todos os automorfismos em L que fixam os elementos de K. A operação de grupo é a composição de morfismos e o elemento neutro é o morfismo identidade.

Como caso particular temos $K \equiv \mathbb{F}_p$ (corresponde a ser $r = 1$, $p = q$ e $s = n$). Aqui o resultado diz-nos que $\mathbb{F}_{p^n}$ é sempre uma extensão de $\mathbb{F}_p$; diz-nos também que o corpo dos automorfismos é formado pelas n potências $\{\, \sigma^k \mid k \in 0..n-1 \,\}$ do morfismo de Frobenius (que, aqui, é simplesmente $x \mapsto x^p$).

141 Definição

Sejam L e K corpos finitos como no facto 140. O **traço** *de L em K é a aplicação definida por*

$$\mathsf{tr}_{L/K}(x) \doteq \sum_{k=0}^{s-1} \sigma^k(x)$$

A **norma** *de L em K é a aplicação*

$$\mathsf{nr}_{L/K}(x) \doteq \prod_{k=0}^{s-1} \sigma^k(x)$$

em que $\sigma : x \mapsto x^q$ denota o morfismo de Forbenius de L em K.

142 Facto

Subentendendo-se os corpos L e K. Para todo $x, y \in L$ e todo $a \in K$, verifica-se:

(i) $\mathsf{tr}(x) \in K$ *e* $\mathsf{nr}(x) \in K$

(ii) $\mathsf{tr}(x+y) = \mathsf{tr}(x) + \mathsf{tr}(y)$ *e* $\mathsf{tr}(a\,x) = a\,\mathsf{tr}(x)$,

(iii) $\mathsf{nr}(x \cdot y) = \mathsf{nr}(x) \cdot \mathsf{nr}(y)$ *e* $\mathsf{nr}(a\,x) = a^s \mathsf{nr}(x)$

(iv) $\mathsf{tr}(\cdot)$ *é uma aplicação sobrejectiva de L em K e* $\mathsf{nr}(\cdot)$ *é uma aplicação sobrejectiva de L^* em K^*.*

Esboço de prova

Este resultado pretende afirmar que o traço $\mathrm{tr}(x)$ é sempre um elemento de $\mathbb{F}_p$, que todo $c \in \mathbb{F}_p$ é traço de algum $x \in \mathbb{F}_m$ e que a aplicação preserva somas e multiplicações escalares.

Seja $t = \mathrm{tr}(x)$; então $\sigma(t) = \sum_{k=1}^{n} \sigma^k(x)$; como $\sigma^n(x) = x$ então $\sigma(t) = x + \sum_{k=1}^{n-1} \sigma^k(x) = t$; concluímos que t é fixo pelo morfismo de Frobenius e, por isso, tem de ser um elemento de $\mathbb{F}_p$.

Transformações semelhantes (usando as propriedades de σ) permitem concluir facilmente que $\mathrm{tr}(\cdot)$ preserva somas e multiplicação escalar.

Para provar que é sobrejectiva, considere-se um qualquer $c \in \mathbb{F}_p$; seja $y \in \mathbb{F}_m$ tal que $\mathrm{tr}(y) \neq 0$; um tal y tem de existir por que existem, quanto muito, p^{n-1} raizes do polinómio $x + x^p + x^{p^2} + \cdots + x^{p^{n-1}}$ e existem p^n elementos em $\mathbb{F}_m$. Definindo $x \doteq (c/\mathrm{tr}(y))\, y$, temos um elemento com traço c.

2.6 Corpos de Galois

São os corpos finitos de característica 2.

143 FACTO

Se $\mathbb{F}_m$ tem característica 2 e dimensão n e $\mathcal{B} \subseteq \mathbb{F}_m$ é um qualquer subconjunto com n elementos de tal forma que nenhum seu subconjunto não vazio soma zero, então o seu "power set" $\wp(\mathcal{B})$ gera o corpo $\mathbb{F}_m$ da seguinte forma: cada $x \in \mathbb{F}_m$ é representado pelo único $\alpha \subseteq \mathcal{B}$ cuja soma de elementos coincide com x.

$$x = \Sigma_{a \in \alpha} a$$

Neste caso $\mathcal{B}$ diz-se uma **base** de $\mathbb{F}_{2^n}$. As bases mais frequentes são

Base Polinomial Tipo 0	$\mathcal{B} = \{1, \beta, \beta^2, \ldots, \beta^{n-1}\}$
Base Polinomial Tipo 1	$\mathcal{B} = \{\beta, \beta^2, \ldots, \beta^{n-1}, \beta^n\}$
Base Normal	$\mathcal{B} = \{\rho, \rho^2, \rho^4, \ldots, \rho^{2^{n-1}}\}$

com β uma raíz do polinómio característico em $\mathbb{F}_m$ e ρ um elemento com traço 1.

EXEMPLO 17: O corpo finito $\mathbf{GF}(2^4)$ determinado pelo polinómio característico $\mathbf{c}[X] = (X^4 + X + 1)$ está representado na tabela 3 numa base polinomial do tipo 0 e usando polinómios na variável X. Note-se que, genericamente, o gerador do grupo cíclico $\mathrm{GF}(2^m)^*$ é muito simples de encontrar: é o polinómio X.

X^i	$X^i \mod \mathbf{c}[X]$	$(\mathbb{Z}_2)^4$	X^i	$X^i \mod \mathbf{c}[X]$	$(\mathbb{Z}_2)^4$
$-$	0	0000			
X^0	1	0001	X^1	X	0010
X^2	X^2	0100	X^3	X^3	1000
X^4	$X+1$	0011	X^5	X^2+X	0110
X^6	X^3+X^2	1100	X^7	X^3+X+1	1011
X^8	X^2+1	0101	X^9	X^3+X	1010
X^{10}	X^2+X+1	0111	X^{11}	X^3+X^2+X	1110
X^{12}	X^3+X^2+X+1	1111	X^{13}	X^3+X^2+1	1101
X^{14}	X^3+1	1001	X^{15}	1	0001

Figura 3: $\mathrm{GF}(2^4)$

Usando essa tabela pode-se ver alguns exemplos de codificação de $\mathbb{Z}_{16}$ em GF(2^4)

$$7 + 11 = 0111 + 1011 = 1100 = 12$$

$$7 * 11 = 0111 * 1011 = X^{10} * X^7 = X^{17} = X^2 = 0100 = 4$$

$$(9)^2 = (1001)^2 = (X^{14})^2 = X^{28} = X^{13} = X^3 + X^2 + 1 = 1101 = 13$$

$$9^{-1} = (X^{14})^{-1} = X^{-14} = X^1 = 0010 = 2$$

Os corpos finitos binários GF(2) são particularmente importantes em Criptografia e, por isso, é necessário pensar em mecanismos computacionais eficientes para manipular estas estruturas.

Note-se que, sendo GF(2) é isomórfico com $\mathbb{Z}_2^n$ (vectores de n bits), o que representa cada uma destas componentes vai determinar o algoritmo usado para realizar cada uma das operações básicas.

Um primeiro ponto importante é a especialização das noções de *traço* e *norma* em GF(2^n). Se atender-mos à definição 141 temos, neste caso, característica $p = 2$ e extensão em cause é $[\text{GF}(2^n)/\text{GF}(2)]$, temos $s = n$ e o morfismo de Frobenius é $\sigma : x \longrightarrow x^2$. Portanto

$$\text{tr}(x) = \sum_{k=0}^{n-1} x^{2^k} \quad , \quad \text{nr}(x) = \prod_{k=0}^{n-1} x^{2^k}$$

Falando brevemente da noção de norma em $\mathrm{GF}(2^n)$ tem-se

$$\mathrm{nr}(x) = \prod_{k=0}^{n-1} x^{2^k} = x^{(\sum_{k=0}^{n-1} 2^k)} = x^{2^n-1}$$

Portanto $\mathrm{nr}(x) = 1$ se $x \neq 0$ e $\mathrm{nr}(x) = 0$ se $x = 0$. A norma é, em $\mathrm{GF}(2^n)$ o simétrico do **símbolo de Kronecker**: $\delta(x) = 1$ sse $x = 0$.

O processo para cálculo do traço é mais complexo e vai depender do tipo de base utilizada.

Bases Polinomiais

Usando a representação $\mathbb{Z}_2(\beta)$ (com β uma raíz do polinómio característico), a forma (49) no facto 138

$$x_0 + x_1 \cdot \beta + \cdots + x_{n-1} \cdot \beta^{n-1} \qquad x_i \in \mathrm{GF}(2) \tag{50}$$

indica que o conjunto

$$\mathcal{B}_{\beta,0} = \left\{1 \ , \ \beta \ , \ \beta^2 \ , \ \cdots \ , \ \beta^{n-1}\right\} \tag{51}$$

forma uma base polinomial do tipo 0.

Um vector de bits $x^{\tilde{}} \in \mathrm{GF}(2)^n$ identifica (de forma única) um elemento $x \in \mathrm{GF}(2^n)$ através da representação (50).

Genericamente o operador $(\cdot)^{\tilde{}}$ *associa um emento* $x \in \mathrm{GF}(2^n)$ *à sua representação* $x^{\tilde{}} \in \mathrm{GF}(2)^n$ *numa base* $\mathcal{B}$ *implícita no contexto.*

Um elemento importante é o que representa β^n. Note-se que, se for $\mathbf{c}[X]$ o polinómio característico, tem-se

$$c_0 + c_1 \cdot \beta + \cdots + c_{n-1} \cdot \beta^{n-1} = \beta^n$$

porque β é raíz do polinómio; isto significa que β^n é representado pelo vector $c = (c_0, c_1, \cdots, c_{n-1}) \in \mathrm{GF}(2)^n$ formado pelos coeficientes do polinómio característico para termos de ordem $< n$.

No exemplo 17 note-se como o polinómio X^4 é equivalente a $1 + X$; portanto a representação de β^4, neste caso, seria o vector $(1, 1, 0, 0)$.

Com representação polinomial de tipo 0 o cálculo da soma é muito simples: para somar dois elementos $x, y \in \mathrm{GF}(2^n)$ representados pelos vectores $x^{\tilde{}}, y^{\tilde{}} \in \mathrm{GF}(2)^n$ basta somar as componentes bit a bit: a representação de $x + y$ será $x^{\tilde{}} \oplus y^{\tilde{}}$.

O cálculo da multiplicação é mais complexo e exige a seguinte definição

144 DEFINIÇÃO

A ***matriz companheira*** *do polinómio* $\mathbf{c}[X]$ *é uma matriz* $\mathbf{A} \in \mathsf{GF}(2)^{n\times n}$ *dada por*

$$\begin{cases} \mathbf{A}_{ij} = 1 \quad sse\ i = j+1 & para\ j < n-1 \\ \mathbf{A}_{ij} = c_i & para\ j = n-1 \end{cases}$$

A matriz companheira de $\mathbf{c}[X]$ é a matriz que tem esse polinómio como polinómio característico e é "o mais simples possível": a última coluna coincide com o vector das componentes de $\mathbf{c}$ e as $n-1$ primeiras colunas têm o primeiro elemento sempre 0 e os restantes são determinados pela matriz identidade $\mathbf{I}_{n-1}$ de dimensão $n-1$.

No exemplo 17 a matriz companheira seria

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

A relação entre multiplicação e matriz companheira deriva do seguinte facto

145 FACTO

Para quaisquer $x, y \in \mathsf{GF}(2^n)$, *verifica-se* $(\beta \cdot x)^{\tilde{}} = \mathbf{A}\, x^{\tilde{}}$ *e, genericamente,*

$$(y \cdot x)^{\tilde{}} \;=\; \bigoplus_{y_i=1} \mathbf{A}^{\,i}\, x^{\tilde{}} \tag{52}$$

Para um qualquer vector $u \in \mathsf{GF}(2)^n$, o cálculo de $\mathbf{A}\, u$ é particularmente simples; se representar-mos por $\overrightarrow{u}$ o vector formado pelo desvio ("shift") de um bit de u para a direita, então facilmente se confirma que $\mathbf{A}\, u \;=\; \overrightarrow{u} \oplus u_{n-1} \cdot c$.

Um algoritmo eficiente para a multiplicação em (52), será

```
M := 0
para i = 0 ate n-1 fazer
  se bit0(y) entao M := M + x ;  shifleft(y)
  shiftright(x) ; se carry entao x := x + c
fim
```

Numa base polinomial o cálculo do traço segue directamente a definição e usa este algoritmo de multiplicação para construir os diferentes quadrados. Para calcular o traço de $\tilde{x}$ faz-se

```
T := x
```

```
para i = 1 ate n-1 fazer
  T := x + T*T
fim
```

Uma base polinomial do tipo 1 tem a forma

$$\mathcal{B}_{\beta,1} = \left\{\beta \ , \ \beta^2 \ , \ \cdots \ , \ \beta^{n-1} \ , \ \beta^n\right\} \tag{53}$$

Aqui é o elemento 1 que é escrito como uma soma de elementos da base; sabendo que o polinómio característico tem sempre $c_0 = 1$, temos $\mathbf{c}(\beta) = 0$ e portanto

$$1 = c_1 \cdot \beta + c_2 \cdot \beta^2 + \cdots + c_{n-1} \cdot \beta^{n-1} + c_n \cdot \beta^n$$

tendo em atenção que se tem sempre $c_n = 1$.

Bases Normais

Escolhendo um elemento $\rho \in \mathrm{GF}(2^n)$ tal que $\mathrm{tr}(\rho) = 1$ então

$$\mathcal{N}_\rho = \left\{\rho \ , \ \rho^2 \ , \ \cdots \ , \ \rho^{2^k} \ , \ \cdots \ , \ \rho^{2^{n-1}}\right\}$$

forma uma base normal. De facto a soma dos elementos de $\mathcal{N}_\rho$ é igual a 1 porque coincide com o traço de ρ (que, por hipótese, é igual a 1); nenhum outro subconjunto não-vazio de $\mathcal{N}_\rho$ pode somar 0 porque, por aplicação sucessiva da função quadrado aos seus elementos, seriam gerados todos os elementos de $\mathcal{N}_\rho$ que, somados, dariam 0 (contradizendo a conclusão anterior).

Um vector $x\tilde{} = (x_0, \cdots, x_{n-1}) \in \mathrm{GF}(2)^n$ representa um elemento $x \in \mathrm{GF}(2^n)$ através da soma

$$x = x_0 \cdot \rho + x_1 \cdot \rho^2 + x_2 \cdot \rho^4 + \cdots + x_{n-1} \cdot \rho^{2^{n-1}} \tag{54}$$

As igualdades essenciais para os diversos algoritmos são

$$1 = \rho + \rho^2 + \rho^4 + \cdots + \rho^{2^{n-1}} \quad , \quad \rho = \rho^{2^n}$$

A primeira resulta da hipótese $\mathrm{tr}(\rho) = 1$; a segunda deriva da construção do corpo finito.

Tal como nas bases polinomiais, a representação da soma é a soma bit-a-bit das representações: $(x+y)\tilde{} = x\tilde{} \oplus y\tilde{}$, com $x, y \in \mathrm{GF}(2^n)$.

Outras operações simples são a construção de quadrados e de traços; de facto, usando (54), tem-se

$$x^2 = (\sum_{k=0}^{n-1} x_k \cdot \rho^{2^k})^2 = \sum_{k=0}^{n-1} x_k \cdot \rho^{2^{k+1}} =$$

$$= x_{n-1} \cdot \rho + \sum_{k=1}^{n-1} x_{k-1} \cdot \rho^{2^k}$$

isto porque $x_{n-1} \cdot \rho^{2^n} = x_{n-1} \cdot \rho$. Consequentemente a representação de x^2 obtém-se fazendo um desvio com rotação para a direita dos bits de de $x^{\sim}$; esta operação será representada por $\overset{\circlearrowleft}{x^{\sim}}$.

$$(x^2)^{\sim} = (\overset{\circlearrowleft}{x^{\sim}})$$

Para o cálculo do traço note-se que, para todo k, $\mathsf{tr}(\rho^{2^k}) = \mathsf{tr}(\rho) = 1$. Portanto

$$\mathsf{tr}(x) = \sum_{k=0}^{n-1} x_i = \mathsf{Tr}(x^{\sim})$$

Nota: Para qualquer $u \in \mathsf{GF}(2)^n$ define-se $\mathsf{Tr}(u) = \sum_{i=0}^{n-1} u_i$

A multiplicação

$$x \cdot y \;=\; \sum_{x_i=1} \sum_{y_j=1} \rho^{2^i} \cdot \rho^{2^j} \tag{55}$$

é bastante mais complicada e, genericamente, mais complexa nas bases normais do que nas bases polinomiais. A dificuldade reside no facto de (55) ser formado por termos da forma $\rho^{2^i} \cdot \rho^{2^j} = \rho^{2^i+2^j}$ que é necessário converter para termos do tipo ρ^{2^k}.

Porém, para alguns valores particulares de n, a conversão é simples e a multiplicação nas bases normais é tão ou mais eficiente que em bases polinomiais. São as chamadas **bases normais óptimas**.

146 FACTO

Se $p = n + 1$ é primo e 2 é gerador do grupo cíclico $\mathbb{Z}_p^*$ então existe ρ que verifica $\mathrm{tr}(\rho) = 1$ e em que o conjunto

$$\left\{ 1 \;,\; \rho \;,\; \rho^2 \;,\; \rho^4 \;,\; \cdots \;,\; \rho^{2^{n-1}} \right\}$$

determina um sub-grupo cíclico de $\mathrm{GF}^*(2^n)$ de ordem p.

Prova: Se p for primo então $2^{p-1} = 1 \pmod p$ e, portanto, $2^n - 1 = 2^{p-1} - 1$ é divisível por p. Como $2^n - 1$ é a ordem e $\mathrm{GF}^*(2^n)$, concluímos que $\mathrm{GF}^*(2^n)$ contém um sub-grupo cíclico $G \subseteq \mathrm{GF}^*(2^n)$ de ordem p.

Seja ρ um gerador de G; deste modo $G = \{\, \rho^s \mid s \in \mathbb{Z}_p \,\}$. Como, por hipótese, 2 é um gerador de $\mathbb{Z}_p^*$ (que tem ordem $p-1=n$) as potências $2^k \pmod p$ (com $k \in \mathbb{Z}_n$) geram os expoentes $s = 1, 2, \cdots, p-1$ (mas não o expoente $s=0$ que determina o elemento $\rho^0 = 1$); portanto temos

$$G = \{1\} \cup \{\, \rho^{2^k} \mid k \in \mathbb{Z}_n \,\} = \left\{1 \ , \ \rho \ , \ \rho^2 \ , \ \cdots \ , \ \rho^{2^{n-1}}\right\}$$

Resta provar que $\mathrm{tr}(\rho) = 1$; tem-se

$$\mathrm{tr}(\rho) = \sum_{k=0}^{n-1} \rho^{2^k} = \sum_{s=1}^{p-1} \rho^s = (\rho^p + \rho) \cdot (1+\rho)^{-1} = 1$$

porque, dada a ordem de G, $\rho^p = 1$.

147 FACTO

Nas condições do facto 146 tem-se:

(i) Se $2^i + 2^j = 0 \pmod p$ *então*

$$\rho^{2^i} \cdot p^{2^j} = \rho + \rho^2 + \rho^4 + \cdots + \rho^{2^{n-1}}$$

(ii) Se $2^i + 2^j \neq 0 \pmod p$ *então*

$$\rho^{2^i} \cdot p^{2^j} = \rho^{2^{\lambda_{ij}}} \quad \textit{com} \quad \lambda_{ij} \doteq i + \tau(j-i) \pmod n$$

em que $\tau(\cdot)$ *denota o logaritmo de Zech de base* 2 *em* $\mathbb{Z}_p^*$.

Prova

O resultado anterior diz-nos que os elementos ρ^{2^i}, p^{2^j} pertencem a um subgrupo multiplicativo de ordem p. Portanto $\rho^{2^i} \cdot p^{2^j} = \rho^{2^i + 2^j}$ é um elemento do mesmo subgrupo; o que significa que é 1 ou então é da forma ρ^{2^k} para algum $k \in \mathbb{Z}_n$.

Se $2^i + 2^j = 0 \pmod p$ então $\rho^{2^i} \cdot p^{2^j} = 1 = \mathrm{tr}(\rho) = \rho + \rho^2 + \cdots + \rho^{2^{n-1}}$.

Se $2^i + 2^j \neq 0 \pmod p$ então pode-se escrever na forma $2^i\,(1 + 2^{j-i}) \pmod p$. Se $2^k \neq -1 \pmod p$, o logaritmo de Zech $\tau(k) \in \mathbb{Z}_n$ está definido e é um elemento que verifica $2^{\tau(k)} = 2^k + 1 \pmod p$; então concluímos (com os expoentes vistos sempre como elementos de $\mathbb{Z}_n$)

$$2^i + 2^j = 2^i\,(1 + 2^{j-i}) = 2^i\,2^{\tau(j-i)} = 2^{i + \tau(j-i)} \pmod p$$

Tabelando o logaritmo de Zech, o que não é muito difícil para os valores usuais de n, este resultado permite construir uma forma computacionalmente eficiente de implementar (55): permite construir o vector de bits, que representa $\tilde{x} \cdot \tilde{y}$, usando apenas operações básicas *xor* e *shift* sobre os vectores de bits x e y.

EXEMPLO 18: Tomemos de novo $\mathrm{GF}(2^4)$; note-se que $p = n + 1$ é primo ($p = 5$) e que 2 é gerador de $\mathbb{Z}_5^*$; de facto, em $\mathbb{Z}_5^*$, $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 3$.

Portanto $\mathrm{GF}(2^4)$ tem uma base normal óptima e as multiplicações podem ser facilmente efectuadas. Tabelando o logaritmo de Zech pela definição ($2^{\tau(k)} = 2^k + 1$) tem-se

$$\tau(0) = 1 \ , \ \tau(1) = 3 \ , \ \tau(2) = \perp \ , \ \tau(3) = 2$$

Suponhamos que se se pretende efectuar a multiplicação dos elementos $x, y \in \mathrm{GF}(2^n)$ que têm as representações $x^{\tilde{}} = (1, 0, 1, 0)$ e $y^{\tilde{}} = (0, 1, 1, 0)$. Isto significa que $x = \rho + \rho^4$ e $y = \rho^2 + \rho^4$.

Denotamos por Λ_{ij} a representação vectorial do elemento $\rho^{2^i} \cdot \rho^{2^j}$. Usando (55) vemos que só interessa calcular Λ_{ij} quando se verifica $x_i = 1$ e $y_j = 1$.

$$(x \cdot y)^{\tilde{}} \ = \ \Lambda_{01} \oplus \Lambda_{02} \oplus \Lambda_{21} \oplus \Lambda_{22}$$

Sempre que for $j - i = 2 \pmod 4$ obtemos um valor para o qual o logaritmo de Zech é indefinido e, neste caso, o resultado anterior diz-nos que $\Lambda_{ij} = (1, 1, 1, 1)$. Nas restantes situações Λ_{ij} é um vector que tem uma única componente igual a 1 determinada pelo índice $\lambda_{ij} \doteq i + \tau(j - i) \pmod 4$.

Para facilitar o cálculo construímos a seguinte tabela

i	j	$j-i$	λ_{ij}
0	1	1	3
0	2	2	$\perp$
2	1	3	0
2	2	0	3

$$\Lambda_{01} = \Lambda_{22} = (0,0,0,1)$$

$$\Lambda_{02} = (1,1,1,1) \ , \ \Lambda_{21} = (1,0,0,0)$$

portanto $(x \cdot y)^{\sim} = (0,1,1,1)$

ou seja $x \cdot y = \rho^2 + \rho^4 + \rho^8$

2.7 Geração de sequências aleatórias e pseudo-aleatórias

A segurança de muitas técnicas criptográficas depende da capacidade de se gerar quantidades imprevisíveis. Sem perda de generalidade pode-se considerar que essas quantidades são bits e são geradas por duas classes de processos:

Sequências aleatórias são produzidas por dispositivos que dependem de fontes aleatórias naturais.

Em *hardware*: o ruído térmico em resistências ou semicondutores, as flutuações na frequência de um oscilador, etc...

Em *software*: o relógio do sistema, o tempo entre eventos de entrada (toques de tecla, movimentos do rato,...), parâmetros de carga do sistema operativo (tamanho do *heap* ou do *stack* de processos do sistema, tamanho de *buffers* ou filas de espera, ...).

Nenhuma destas fontes, isoladamente, pode ser considerada à prova de ataque. Recomenda-se sempre uma **mistura de fontes** que é feita concatenando valores de várias fontes, passando este valor por uma função de *hash* (como o SHA-1) e seleccionando apenas alguns dos bits do resultado.

Neste caso é considerado intratável (apesar de não ser formalmente demonstrado) atacar uma das fontes de modo a criar uma **tendência** ("*bias*") em relação à emissão de um determinado valor (0 ou 1) ou **correlacionar** a emissão de um bit com emissões anteriores.

Sequências Pseudo-aleatórias de elementos de um domínio finito X

$$x_1, x_2, \ldots, x_i, \ldots$$

É produzida a partir de outra sequência $s_0, s_1, \ldots, s_k, \ldots$ por:

$$s_k = G(s_{k-1}), \qquad x_k = h(s_k), \quad \text{com} \quad k > 0$$

(sendo h uma função *one-way*) de tal forma que é satisfeita a condição: *Conhecidos os primeiros k valores da sequência $x_1, \ldots, x_k$ é computacionalmente intratável prever o próximo elemento x_{k+1} com probabilidade superior a $\varepsilon + |X|^{-1}$; $\varepsilon > 0$ é arbitrariamente pequeno.*

Seleccionando alguns dos bits dos vários x_k constrói-se a sequência de bits pretendida.

O valor s_0, que determina toda a sequência dos s_k (e portanto dos x_k), chama-se *semente* (ou "*seed*").

Sequências que não verificam a condição anterior não são consideradas **criptográficamente seguras** mas podem ser usadas noutro tipo de aplicações (por exemplo, na geração de valores de teste nos algoritmos de teste de números primos).

EXEMPLO 19: (**gerador linear**)

$$s_k = a \times s_{k-1} + b \qquad a, b, s_k \in \mathbb{Z}_p, \quad a \neq 0$$
$$x_k = s_k \pmod 2 \qquad x_k \in \mathbb{Z}_2$$

Prova-se que este gerador não é criptograficamente seguro pois consegue-se determinar s_0 a partir do conhecimento de p e de poucos bits x_k.

Gerador RSA

Parâmetros: dois primos p e q, para os quais a factorização de $m = p \cdot q$ é intratável, e $a \in \mathbb{Z}^*_{\phi(m)}$; os valores a, m podem ser tornados públicos, o valor $\phi(m) = (p-1) \cdot (q-1)$ é secreto e p e q são destruídos.

Semente: um valor aleatório $s_0 \in \mathbb{Z}^*_m$ (gerado um por dos mecanismos de *hardware* ou *software* atrás referidos).

Gerador:

$$s_k = s_{k-1}^a \pmod m, \qquad x_k = s_k \pmod 2$$

Este gerador é criptograficamente seguro mas pouco eficiente porque necessita de uma exponenciação para cada bit gerado. Existe uma variante, chamado *gerador de Micalli-Schnorr*, que dá, por iteração, tantos bits quantos os necessários para representar m (tipicamente 1024).

Gerador BBS (Blum-Blum-Shub)

Parâmetros dois primos p e q para os quais a factorização de $m = p \cdot q$ e intratável e que verificam $p = q = 3 \pmod 4$.

Semente: um valor $s_0 \doteq \omega^2 \pmod m$ em que $\omega \in \mathbb{Z}_m^*$ é aleatório.

Gerador

$$s_k = s_{k-1}^2 \pmod m, \qquad x_k = s_k \pmod 2$$

O gerador BBS é criptograficamente seguro mesmo quando se aproveita mais do que um bit por iteração. Porém não existe actualmente um processo de determinar quantos bits a mais é possível retirar de cada iteração mantendo a condição de segurança criptográfica.

Norma ANSI X9.17

Parâmetros: usa a função de cifragem do triplo DES, com uma chave composta κ, e um parâmetro $I \doteq \{d\}_\kappa$, sendo d uma representação da data+hora com 64 bits.

Semente: s_0 é um valor aleatório com 64 bits.

Gerador:

$$s_1 = I \oplus s_0 \qquad x_i = \{s_i\}_\kappa, \qquad s_{i+1} = I \oplus \{I \oplus x_i\}_\kappa \quad i > 0$$

Gerador FIPS 186

O *Federal Information Processing Standard* (FIPS) 186 acompanha a norma do DSA (*Digital Signature Algorithm*) e define dois geradores de números pseudo-aleatórios: uma versão para a geração de pares de chaves públicos-privada e uma versão para geração de chaves de sessão.

Na primeira versão o utilizador pode modificar a semente produzida pelo implementador com uma semente por ele escolhida.

Essencialmente o gerador tem uma construção do tipo da anterior envolvendo, como "misturador" de bits em cada iteração a função de *hash* SHA-1.

Para detalhes consultar o `Handbook of Applied Cryptography` de Menezes *et al.*.

2.8 Factorização de Inteiros

Dado um inteiro $m > 1$, determinar primos $1 < p_1 < \cdots < p_k$ e expoentes $e_1, \ldots, e_k$ tais que $m = p_1^{e_1} \times \ldots \times p_k^{e_k}$.

A factorização determina uma classe de funções *one-way*: computacionalmente a multiplicação é trivial mas a sua "inversa" factorização é, para valores apropriados de m, intratável. Existem porém algumas **factorizações triviais**.

Factorização por divisão Faz-se variar p ao longo dos "pequenos primos" $(2, 3, 5, \ldots)$ e testa-se $m = 0 \pmod{p}$. Se tal for possível faz-se $m \leftarrow m/p$ reduzindo a complexidade do problema da factorização.

Formalmente esta técnica encontra qualquer factorização de m só que implica $\lceil \sqrt{m} \rceil$ cálculos do módulo o que é intratável para grandes m.

Factorização de Fermat Se $m = p \times q$, com $q < p$ primos ímpares "próximos", encontrar p e q é simples.

Define-se $\mu = (p+q)/2$ e $\upsilon = (p-q)/2$; donde $m = p \times q = (\mu^2 - \upsilon^2)$ e

$$\mu^2 = m + \upsilon^2 \qquad \text{com} \quad \upsilon^2 \ll m \tag{†}$$

A partir do valor inicial $\mu = \lceil (\sqrt{m}) \rceil$ e incrementando sucessivamente $\mu \leftarrow \mu + 1$, testam-se, para cada μ, valores $\upsilon = 0, 1, \ldots, \lceil (\sqrt{\mu^2 - m}) \rceil$ até que (†) se verifique.

Conhecidos μ e υ determina-se $p = \mu + \upsilon$ e $q = \mu - \upsilon$.

EXEMPLO 20: Para $m = 1127843$ temos o valor inicial $\mu = \lceil(\sqrt{m})\rceil = 1062$; como $m - \mu^2 = 1$ é logo um quadrado perfeito, conclui-se $\upsilon = 1$ e $\mu = 1062$. Assim $p = 1063$ e $q = 1061$.

Factorização de Pollard

Gerando uma sequência pseudo-aleatória finita $\rho_0, \rho_1, \ldots \rho_n$ em $\mathbb{Z}_m$, calculam-se os valores $d_i = \gcd(\rho_i, m)$ até se obter algum $d_s > 1$ (como resultado do processo) ou se atingir o limite da sequência dos ρ_i (com indicação de falha).

Método rho Os ρ_i são determinadospor um gerador quadrático

$$x_i = x_{i-1}^2 + 1 \pmod m \qquad y_i = (y_{i-1}^2 + 1)^2 + 1 \pmod m \qquad (\dagger)$$

$$\rho_i = x_i - y_i \pmod m$$

Justificação Se $x_0 = y_0$ então $y_i = x_{2i}$, para todo $i \geq 0$: a solução é atingida quando for $x_{2s} = x_s \pmod p$ para algum p (desconhecido) divisor de m.

Todo o eventual p, divisor de m, determina uma sequência $x'_i = x_i \pmod p$ que verifica a mesma igualdade (†) e que, eventualmente, será periódica; quando se atingir um s que seja múltiplo do período teremos $x'_{2s} = x'_s$ e portanto atinge-se a solução. O número de operações necessárias para encontrar s é da ordem de $\sqrt{\sqrt{m}}$.

Método (p-1) Dado um limite B, os ρ_i são gerados por

$$X_1 = 2, \quad X_i = (X_{i-1})^i \pmod{m} \qquad i = 2, \ldots, B$$

$$\rho_i = X_i - 1$$

Justificação Se p for um divisor primo de m e for $q \leq B$ para todo o divisor primo de $(p-1)$, então $(p-1)$ tem de dividir $B!$. Pelo teorema de Fermat, será $2^{B!} = 1 \pmod{p}$.

No final do ciclo temos $X_B = 2^{B!} \pmod{m}$; portanto $X_B = 2^{B!} = 1 \pmod{p}$. Logo $(X_B - 1)$ é múltiplo de p e será $\gcd(X_B - 1, m) > 1$.

A condição pode-se verificar antes de i atingir B; basta que X_i acumule um expoente múltiplo de $(p-1)$.

O problema está na determinação do valor B que seja limite superior de todos os factores de $(p-1)$

As técnicas de factorização poderiam, por si só, dar origem a um curso tanto ou mais extenso quanto o que aqui apresentamos. Recentemente tem aparecido algoritmos que diminuiriam substancialmente a complexidade computacional deste problema. O criptógrafo preocupado com a segurança dos seus sistemas deve estar atento a estes métodos.

ECM (Elliptic Curve Method) É uma variante do método $(p-1)$ de Pollard; note-se que $(p-1)$ é a ordem do grupo cíclico $\mathbb{Z}_p^*$ e os elementos X_i percorrem esse grupo. O ECM substitui o grupo $\mathbb{Z}_p^*$ por uma curva elíptica aleatória sobre $\mathbb{Z}_p$ cuja ordem B é computacionalmente limitada.

Detalhes deste e dos outros algoritmos desta secção podem-se obter em A Course in Computacional Algebraic Number Theory de H.Cohen.

Factorização Quadrática

148 FACTO

Se se verificar

$$x^2 = y^2 \pmod{m} \qquad \& \qquad x \neq \pm y \pmod{m} \tag{†}$$

então $\gcd(x-y, m)$ *é um divisor não trivial de* m. *Se* m *for ímpar e tiver* k *factores primos distintos então, para cada* $y \in \mathbb{Z}_m^*$, *existem* 2^{k-1} *soluções* $x \in \mathbb{Z}_m$ *de (†).*

Seja $\mathcal{B} = \{p_1, \ldots, p_k\}$ uma **base de primos**; um **número-$\mathcal{B}$** é um inteiro cujos factores primos são todos elementos de $\mathcal{B}$. Se $a = p_1^{e1} \times \ldots \times p_k^{ek}$, com $e_i \geq 0$, é um número-$\mathcal{B}$ representamos por $\|a\|$ o vector em $(\mathbb{Z}_2)^k$ formado pelas paridades dos vários expoentes: $\langle e_1 \pmod 2, \ldots, e_k \pmod 2 \rangle$

Por tentativas geram-se pares $\langle x_i \ , \ a_i \rangle$ tais que $x_i^2 = a_i \pmod m$, e os a_i são números-$\mathcal{B}$ de factorização conhecida.

Se forem encontrados $a_1, \ldots a_\mu$ tais que $\|a_1\| + \ldots + \|a_p\| = 0$ então o produto $a_1 \times \ldots \times a_\mu$ é um quadrado perfeito cuja raiz quadrada y é facilmente calculada já que as factorizações dos a_i são conhecidas; se for

$x_1 \times \ldots \times x_\mu \neq \pm y \pmod{m}$, como temos

$$(x_1 \times \ldots \times x_\mu)^2 = a_1 \times \ldots \times a_\mu = y^2 \pmod{m}$$

pode-se calcular $\gcd(x_1 \times \ldots \times x_\mu - y, m)$ e obter o factor pretendido.

Os melhores métodos generalistas conhecidos, nomeadamente o **Quadratic Sieve Method**, são baseados neste processo e definem critérios eficientes para gerar as diversas escolhas aqui indicadas.

2.9 Logaritmo Discreto

A segurança de muitas técnicas criptográficas depende crucialmente da característica *one-way* da "exponenciação" em **grupos cíclicos**: a função directa é computacionalmente tratável mas a função inversa deve ser computacionalmente intratável.

A propriedade criptográficamente significante de um grupo cíclico finito $\langle G\ ,\ \cdot\rangle$ é a existência do **isomorfismo de grupos** $\mathbb{Z}_{|G|} \simeq G$ estabelecida pela exponenciação $\exp_g : i \mapsto g^i$ e pela sua função inversa $\log_g : G \to \mathbb{Z}_{|G|}$ (o **logaritmo discreto** em G).

O *Problema do Logaritmo Discreto* (PLD) em G é a determinação, dado o gerador $g \in G$ e um elemento $x \in G$, do único $i \in \mathbb{Z}_{|G|}$ tal que $x = g^i$. Em particular para $G \equiv \mathbb{Z}_p^*$ temos

149 Definição
Dado um primo p ímpar, um algoritmo resolve $\textsc{LogDisc}(p)$ *quando, dados um gerador $\alpha \in \mathbb{Z}_p^*$ e um elemento $\beta \in \mathbb{Z}_p^*$, determina o único $a \in \mathbb{Z}_{p-1}$ tal que $\beta = \alpha^a \pmod p$.*

Procura exaustiva em memória

Constrói-se uma tabela de pares $\langle i\ ,\ \alpha^i \pmod p\rangle$, com $i = 0, \ldots, (p-2)$ e ordena-se pela segunda componente. Dado β procura-se este valor na segunda coluna da tabela; o resultado a é dado pela primeira componente do par encontrado.

Complexidade A ordenação é feita para facilitar a procura; o algoritmo exige memória proporcional a p e um número de comparações proporcional a $(p-1)$.

Procura exaustiva no tempo

É gerada a sequência de elementos de $\mathbb{Z}_p^*$

$$x_0 = 1, \qquad x_i = \alpha \times x_{i-1} \pmod{p} \quad i = 1, 2, \ldots, (p-2)$$

que termina quando se verificar $x_i = \beta$. O índice i correspondente é o resultado pretendido.

Complexidade Exige memória constante mas tempo de procura é proporcional a $(p-1)$ multiplicações.

Algoritmo de Shank ("baby-step, giant-step")

É um compromisso entre os dois algoritmos anteriores. Escolhe-se $n = \lceil \sqrt{p} \rceil$, calcula-se $\alpha_n \doteq \alpha^{-n} \pmod{p}$ e constrói-se a tabela de pares $\langle j \ , \ \alpha^j \pmod{p} \rangle$, com $j = 0, \ldots, n-1$.

Como cada $a \in \mathbb{Z}_{p-1}$ pode ser escrito na forma $a = i \times n + j$ com $i, j \in \mathbb{Z}_n$, temos

$$\beta = \alpha^a \pmod{p} \qquad \text{sse} \qquad \beta \times (\alpha_n)^i = \alpha^j \pmod{p}$$

Para $i = 0, 1, \ldots, n-1$ calcula-se $\beta_i \equiv \beta \times (\alpha_n)^i \pmod{p}$ e procura-se, na segunda coluna da tabela, algum elemento igual a β_i ; se existir, com o respectivo índice j e com o índice i de β_i pode-se calcular a.

Complexidade A tabela exige memória proporcional a $\sqrt{p}$; o número total de procuras é proporcional a p e cada uma exige um número de multiplicações proporcional a $\sqrt{p}$.

Em relação aos dois algoritmos anteriores, este é um compromisso memória-tempo.

Algoritmo de Pohlig-Hellman

Pretende-se determinar $a \in \mathbb{Z}_{p-1}$ tal que $\beta = \alpha^a \pmod{p}$. Suponhamos[32] que é conhecida a factorização de $(p-1)$.

Fazendo q percorrer todos os factores primos de $(p-1)$, se for possível determinar $a_q \in \mathbb{Z}_q$ tal que

$$a = a_q \pmod{q} \qquad (\dagger)$$

então é possível calcular a usando o teorema chinês dos restos.

1º caso: q *é primo*

[32] Muitas técnicas criptográficas usam primos da forma $p = q\,2^t + 1$, com q primo.

Seja $r = (p-1)/q$ e $\gamma = \alpha^r \pmod p$; de (†) concluímos $q|(a - a_q)$ e portanto $(p-1)|(a-a_q)\,r$ já que $(p-1) = q\,r$.

Donde $a\,r = r\,a_q \pmod{p-1}$ e , pelo isomorfismo $\mathbb{Z}_{p-1} \simeq \mathbb{Z}_p^*$, temos

$$(\alpha^a)^r = (\alpha^r)^{a_q} \pmod p \qquad \text{ou seja} \qquad \beta^r = (\gamma)^{a_q} \pmod p \qquad (‡)$$

Pode-se ver (‡) como um problema de cálculo do logaritmo discreto a_q de um novo elemento $\beta^r \pmod p$ num grupo multiplicativo de gerador γ e ordem q. Este problema é resolvido por um dos algoritmos anteriores.

2º caso: $q \equiv \mu^e$ *com* μ *primo*.

Representa-se a_q na base μ: temos $a_q \equiv x_1 + \mu x_2 + \cdots + \mu^{e-1} x_e$.
De (†) temos $\mu|(a - x_1)$; tal como anteriormente, com $r = (p-1)/\mu$, tem-se $a\,r = r\,x_1 \pmod{p-1}$ o que permite, determinar x_1 resolvendo um PLD num sub-grupo de ordem μ e gerador $\gamma = \alpha^r \pmod p$.

Tem-se $\mu^2|(a - x_1 - x_2\,\mu)$ donde $(a - x_1)\,(r/\mu) = r\,x_2 \pmod{p-1}$. Daí determina-se x_2 resolvendo um PLD no mesmo sub-grupo.

E assim sucessivamente determinam-se todos os dígitos $x_1, x_2, \ldots x_e$.

Complexidade: *proporcional ao maior divisor primo de* $(p-1)$

Método ρ-Pollard

Quando se pretende resolver o problema do logaritmos discreto num sub-grupo de ordem q de $\mathbb{Z}_p^*$ pode-se usar um método análogo ao método da factorização ρ de Pollard.

Problema Dado q primo que divide $(p-1)$, dados $\alpha, \beta \in \mathbb{Z}_p^*$ de ordem q, determinar $\mu \in \mathbb{Z}_q$ tal que $\beta = \alpha^{\mu} \pmod{p}$.

Algoritmo

1. Divide-se o subgrupo gerado por α em três conjuntos disjuntos S_0, S_1 e S_2 de tamanho aproximadamente igual e fácilmente computáveis.[33]
2. Define-se a sequência

$$x_0 = 1 \qquad x_{i+1} = \begin{cases} \beta\, x_i \pmod{p} & \text{se } x_i \in S_0 \\ x_i^2 \pmod{p} & \text{se } x_i \in S_1 \\ \alpha\, x_i \pmod{p} & \text{se } x_i \in S_2 \end{cases}$$

[33] Por exemplo, $S_i \equiv \{\, x \mid x = i \pmod 3 \,\} \quad i = 0, 1, 2$.

3. Tem-se $x_i = \alpha^{a_i}\,\beta^{b_i} = \alpha^{a_i+\mu\,b_i} \pmod p$ em que

$$(a_0, b_0) = (0, 0) \qquad (a_{i+1}, b_{i+1}) = \begin{cases} (a_i, b_i + 1) \pmod q & \text{se } x_i \in S_0 \\ (2a_i, 2b_i) \pmod q & \text{se } x_i \in S_1 \\ (a_i + 1, b_i) \pmod q & \text{se } x_i \in S_2 \end{cases}$$

4. Usando a iteração de Pollard-Floyd, determina-se x_k tal que

$$x_k = x_{2k} \pmod p$$

Nesse caso $\alpha^{a_k+\mu\,b_k} = \alpha^{a_{2k}+\mu\,b_{2k}} \pmod p$ e, portanto,

$$\mu = (b_k - b_{2k})^{-1}\,(a_{2k} - a_k) \pmod q$$

A sequência de pares definida em (3.) permite calcular (a_k, b_k) e (a_{2k}, b_{2k}); donde é possível determinar μ.

Complexidade computacional: *proporcional a* $\sqrt{q}$.

3.Funções Booleanas

O projecto e segurança de muitas técnicas criptográficas estão ligadas ao estudo das funções da forma $f : \mathbb{B}^n \longrightarrow \mathbb{B}^n$ ou $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ que tomam como argumento uma sequência finita de *bits* de comprimento fixo e devolvem uma sequência do mesmo tamanho ou então um simples bit.

Das várias representações possíveis para tais funções escolhemos algumas que são particularmente importantes:

1. Funções booleanas de n variáveis booleanas

$$f : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$$

2. Funções boolenas sobre o corpo de Galois de ordem n

$$f : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2)$$

3.1 Funções de argumento $\mathrm{GF}(2)^n$

Nas funções booleanas $f : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$ os argumentos x são vectores de n bits; as operações básicas sobre os argumentos são:

1. Selecção $\mathrm{sel} : \mathbb{Z}_n \times \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$,
$$\mathrm{sel}(i, x) = x_i$$
i.e. dado $i \in \mathbb{Z}_n$, seleciona a componente de ordem i do vector x.
2. Operações binárias $\oplus$, $* : \mathrm{GF}(2)^n \times \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^n$
são as duas funções binárias que aplicam *xor* e *and* bit a bit.

$$(x \oplus y)_i \;=\; x_i + y_i \qquad (x * y)_i \;=\; x_i \cdot y_i$$

A noção de *índice* pode ser gereralizada; podemos tomar como índice um qualquer conjunto de inteiros $i \in \mathbb{Z}_n$; por exemplo, se tivermos $n = 8$, um índice será, por exemplo, $\{0, 3, 7\}$. O valor booleano seleccionado por este índice será

$$x_{0,3,7} \;\doteq\; x_0 \cdot x_3 \cdot x_7$$

Genéricamente um **índice** para funções boolenas de n argumentos booleanos é um sub-conjunto $u \subseteq \mathbb{Z}_n$; o conjunto desses índices representa-se por $\mathbb{U}_n$ e identifica-se com $\wp(\mathbb{Z}_n)$.

A função selecção $\mathsf{sel} : \mathbb{U}_n \times \mathsf{GF}(2)^n \longrightarrow \mathsf{GF}(2)$ generaliza-se facilmente

$$\mathsf{sel}(u, x) = \Pi_{i \in u}\, x_i \qquad , \qquad \mathsf{sel}(\emptyset, x) = 1 \tag{56}$$

150 DEFINIÇÃO
Para $u \in \mathbb{U}_n$ *designa-se por* x_u *o monómio* $\mathsf{sel}(u, x)$. *Designa-se por* $[x]_u$ *o polinómio* $\mathsf{sel}(u, x) \cdot \Pi_{i \notin u}(1 + x_i)$.

Por exemplo, se for $n = 4$ e $u = \{0, 2\}$, temos

$$x_u \;=\; x_0 \cdot x_2 \qquad \text{e} \qquad [x]_u = x_0 \cdot (1 + x_1) \cdot x_2 \cdot (1 + x_3)$$

Toda a função booleana de domínio $\mathsf{GF}(2)^n$ pode ser escrita como um polinómio a n variáveis $x_0, x_1, \ldots, x_{n-1}$ dado por

$$f(x) \;=\; \sum_{u \in \mathbb{U}_n} a(u)\, x_u \tag{57}$$

para uma função $a : \mathbb{U}_n \longrightarrow \mathsf{GF}(2)$.

Esta é a chamada **forma normal algébrica** de f e é completamente determinada pela função a dita *função de índices* ou *espectro de índices* ou, simplesmente, *espectro* se não existirem ambiguidades [34].

Uma forma equivalente de representar a mesma função consiste em considerar o conjunto $\mathcal{A} \subseteq \mathbb{U}_n$ de índices u para os quais $a(u) = 1$; essencialmente $\mathcal{A}$ é o conjunto que tem a função de coeficientes como função característica.

Nesse caso (57) escreve-se

$$f(x) = \sum_{u \in \mathcal{A}} x_u \tag{58}$$

Daqui se deduz que o número total de funções booleanas de argumento $\mathrm{GF}(2)^n$ é 2^{2^n}.

O **grau** de f é definido como $(\max_{u \in \mathcal{A}} |u|)$: i.e. a maior cardinalidade de um índice em $\mathcal{A}$.

Se o grau é 0 ou 1 a função diz-se **afim**.

Claramente que o número total de funções afins é 2^{n+1} metade das quais são **lineares**: i.e. funções f tais que $a(\emptyset) = 0$ ou, equivalentemente, $\emptyset \notin \mathcal{A}$.

[34] Veremos adiante que é importante definir um outra forma de espectro de funções booleanas: o espectro de *Walsh-Hadamard*.

EXEMPLO 21: Considere-se as funções booleanas $f : \mathbb{B}^4 \longrightarrow \mathbb{B}$ com 4 argumentos booleanos. Um exemplo de uma função na forma normal algébrica será

$$f(x) = 1 + x_0 + x_2 + x_1\,x_2 + x_1\,x_3 + x_0\,x_1\,x_3$$

O conjunto dos índices que correspondem a termos não nulos é

$$\mathcal{A} \;=\; \{\emptyset \;,\; \{0\} \;,\; \{2\} \;,\; \{1,2\} \;,\; \{1,3\} \;,\; \{0,1,3\}\}$$

O maior deles tem 3 elementos; por isso f tem grau 3.

Esta função não é afim. Um exemplo de função afim será

$$g(x) = 1 + x_0 + x_2$$

que não é uma função linear devido à presença da constante 1. Um exemplo de função linear será

$$h(x) \;=\; x_0 + x_2$$

Em termos de espectros, o de g é $\{\emptyset \;,\; \{0\} \;,\; \{2\}\}$ enquanto que o de h é $\{\{0\} \;,\; \{2\}\}$.

O **símbolo de Kronecker** de ordem n é a função $\delta : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$ que verifica $\delta(x) = 1$ se e só se $x = (0, 0, \cdots, 0)$.

151 FACTO

Para todo $x \in \mathsf{GF}(2)^n$ *verifica-se*

$$\delta(x) = (1 + x_0) \cdot (1 + x_1) \cdot (1 + x_2) \cdots (1 + x_{n-1}) = [x]_\emptyset$$

Para quaisquer $X, Y \subseteq \mathbb{U}_n$ define-se a sua **união disjunta** por

$$X \uplus Y \doteq (X \setminus Y) \cup (Y \setminus X)$$

e a sua **convolução** por

$$X \circledast Y \doteq \biguplus_{x \in X} \{ x \cup y \mid y \in Y \}$$

152 FACTO

Sejam $\mathcal{A}, \mathcal{B} \subseteq \mathbb{U}_n$ *os espectros de funções* f, g *respectivamente; i.e.*

$$f(x) = \sum_{u \in \mathcal{A}} x_u \qquad g(x) = \sum_{v \in \mathcal{B}} x_v$$

(i) Se f *é uma função constante então: se for* $f(x) = 1$, *o seu espectro é* $\{\emptyset\}$ *e, se for* $f(x) = 0$, *o seu espectro é* $\{\ \}$.

(ii) *Se* $\mathcal{A} = \mathbb{U}_n$ *então* f *coincide com o símbolo de Kronecker* δ. *Se* $\mathcal{A} = \mathbb{U}_n \setminus \emptyset$ *então* $f = 1 + \delta$ *(i.e.* $f(x) = 1$ *se e só se* $x \neq 0$*).*

(iii) $(f + g)$ *tem espectro* $\mathcal{A} \uplus \mathcal{B}$ *e* $(f \cdot g)$ *tem espectro* $\mathcal{A} \boxplus \mathcal{B}$.

$$(f + g)(x) = \sum_{u \in \mathcal{A} \uplus \mathcal{B}} x_u \qquad (f \cdot g)(x) = \sum_{u \in \mathcal{A} \boxplus \mathcal{B}} x_u$$

Corolário: $1 + f$ *(a negação de* f*) tem espectro* $\mathcal{A} \uplus \emptyset$.

(iv) *Se* $g(x) = f(z * x)$, *para algum* $z \in \mathrm{GF}(2)^n$, *então*

$$\mathcal{B} = \{\, u \in \mathcal{A} \mid z_u = 1 \,\}$$

Comentários: O uso de tratamentos espectrais têm longa tradiação em Engenharia. Essencialmente procura-se uma representação alternativa para funções de tal forma que o estudo dessas funções possa ser feito (de maneira mais simples) na representação espectral.

Tradicionalmente procura-se analisar as relações entre propriedades no domínio das funções e propriedades nos domínios dos espectros e formas simples de determinar umas e outras.

Este resultado e os que se seguem vêm dentro dessa tradição. São apresentados várias formas particulares de construir funções e é analisada a forma equivalente no domínio dos espectros.

(i) As funções constantes são polinómioas de grau zero; o espectro de $f(x) = 1$ deriva directamente da definição de $x_\emptyset$. De forma semelhante temos o polinómio vazio que origina $f(x) = 0$.

(ii) Se tivermos $x = (0, 0, \ldots, 0)$ então temos $x_u = 1$ se e só se $u = \emptyset$. Sendo $f(x) = \sum_{x \in \mathbb{U}_n} x_u$ teremos claramente $f(x) = 1$.
Se for $x \neq (0, 0, \ldots, 0)$ então seja $\bar{x} \doteq \{ i \mid x_i = 1 \}$; claramente $x_u = 1$ se e só se $u \subseteq \bar{x}$. O número de índices u tais que $x_u = 1$ é, assim, um número par (é dado por $2^{|\bar{x}|}$); consequentemente $f(x) = \sum_{x \in \mathbb{U}_n} x_u = 0$ já que é a a soma de um número par de elementos não nulos.

(iii) Os espectros de $(f + g)$ e $(f \cdot g)$ resultam directamente da expansão destas duas expressões substituindo $f(x)$ e $g(x)$ pelas respectivas somas.

(iv) Sendo $f(x) = \sum_{u \in \mathcal{A}} x_u$ então

$$g(x) = f(z * x) = \sum_{u \in \mathcal{A}} (z * x)_u = \sum_{u \in \mathcal{A}} z_u \cdot x_u = \sum_{u \in \mathcal{A} \wedge z_u = 1} x_u$$

O conjunto $f^{-1}(1) = \{ x \mid f(x) = 1 \}$ chama-se **suporte** de f e é representado por $\mathrm{supp}(f)$.
Chama-se **peso** de f (representado por $\mathrm{wt}(f)$) à cardinalidade desse conjunto – $\mathrm{wt}(f) = |f^{-1}(1)|$.
A função é **balanceada** quando, para metade dos seus argumentos, o valor for 1; isto é, $\mathrm{wt}(f) = 2^{n-1}$.

153 FACTO
Para um qualquer $z \in \mathrm{GF}(2)^n$, seja $\delta^{(z)}(x) \doteq \delta(x \oplus z)$ (i.e., $\delta^{(z)}(x) = 1$ se e só se $x = z$). Então:

(i) $\delta^{(z)} \cdot \delta^{(w)} = \delta(z \oplus w) \cdot \delta^{(z)}$

(ii) Qualquer função booleana f pode ser representada por

$$f = \sum_{z \in \mathsf{supp}(f)} \delta^{(z)} \tag{59}$$

(iii) O espectro de $\delta^{(z)}$ é o conjunto $\uparrow \bar{z} \doteq \{ u \mid \bar{z} \subseteq u \}$ sendo $\bar{z} \doteq \{ i \mid z_i = 1 \}$ (i.e., $\bar{z}$ é o maior índice u tal que $z_u = 1$).

(iv) Para todo $z \in \mathsf{GF}(2)^n$ tem-se $[x]_{\bar{z}} = \delta^{(z)}(x)$.

Notas O primeiro resultado traduz apenas propriedades de $\delta^{(z)}$ que resultam directamente da definição.

O segundo resultado é consequência imediata do primeiro e prova-se considerando a expansão de $f(x)$ nos casos em que $x \in \mathsf{supp}(f)$ e $x \notin \mathsf{supp}(f)$. Tem muita importância computacional quando o suporte da função é tem baixa cardinalidade (f tem pouco peso) e pode ser calculado facilmente. Nestas circunstâncias (59) é uma forma conveniente de representar a função.

O resultado (iii) é importante nomeadamente porque sugere um ataque a uma função boolena $\mathcal{Q} : \mathbb{B}^n \longrightarrow \mathbb{B}$ como a procura de um $k \in \mathbb{B}^n$ tal que $\mathcal{Q}(k) = 1$.

Suponhamos que tínhamos a certeza que existia só um k nestas condições mas que esse valor era, obviamente, desconhecido. Isso é equivalente a dizer que $\mathcal{Q}$ tem a forma $\delta^{(k)}$, para um k desconhecido. Suponhamos também (e isto é uma suposição muito forte) que existia um modo qualquer de determinar o espectro $\mathsf{Sp}(\mathcal{Q})$ de $\mathcal{Q}$.

O resultado anterior diz-nos que o espectro $\mathsf{Sp}(\mathcal{Q})$ é $\uparrow\bar{k}$. Portanto é um conjunto que tem um limite inferior $\bar{k}$ que pode ser calculado como

$$\bar{k} \;=\; \bigcap_{u\in \mathsf{Sp}(\mathcal{Q})} u$$

Supondo finalmente que esse limite era computável; então fica determinado $\bar{k}$ e, consequentemente, fica determinado k.

Para o provar considere-se a função f que tem $\uparrow\bar{z}$ por espectro. Notando que $x_u = 1$ se e só se $u \subseteq \bar{x}$, tem-se

$$f(x) \;=\; \sum_{u\supseteq \bar{z}} x_u \;=\; \sum_{u\supseteq \bar{z}\wedge u\subseteq \bar{x}} 1$$

Se $x = z$ existe apenas um índice u que satisfaz a condição $u \supseteq \bar{z} \wedge u \subseteq \bar{x}$ (nomeadamente $u = \bar{x} = \bar{z}$). Neste caso o somatório tem uma só parcela e o resultado é 1. Se $x \neq z$ então o conjunto de todos os índices u que satisfazem a condição ou é vazio (quando $\bar{x} \subset \bar{z}$) ou tem um número par de elementos; em qualquer dos casos o somatório é zero. Donde $f(x) = 1$ se e só se $x = z$; portanto, $f \equiv \delta^{(z)}$.

154 TEOREMA

O espectro $\mathsf{Sp}(f)$ *de uma função booleana* f *verifica*

$$\mathsf{Sp}(f) \;=\; \biguplus_{z\in \mathsf{supp}(f)} \uparrow\bar{z} \tag{60}$$

Seja $f^{(z)}$ *a função definida por* $f^{(z)}(x) = f(z \oplus x)$ *. Então*

$$\mathsf{Sp}(f^{(z)}) \;=\; \biguplus_{y \in \mathsf{supp}(f)} \uparrow(\bar{z} \uplus \bar{y}) \tag{61}$$

A prova do primeiro resultado é consequência imediata de (59) e do facto 152.

Para provar o segundo basta notar que

$$f^{(z)}(x) = \; f(z \oplus x) = \sum_{y \in \mathsf{supp}(f)} \delta^{(y)}(z \oplus x) = \sum_{y \in \mathsf{supp}(f)} \delta^{(y \oplus z)}(x)$$

Se atender-mos que $\overline{(y \oplus z)} = \{\, i \mid y_i + z_i = 1 \,\} = \bar{y} \uplus \bar{z}$ e que o espectro de $\delta^{(y \oplus z)}$ é $\uparrow \overline{y \oplus z} = \uparrow(\bar{y} \uplus \bar{z})$ obtém-se a expressão pretendida para o espectro.

A representação do espectro em termos do suporte da função f não é muito conveniente já que é difícil, quase sempre, calcular esse suporte. Por isso é necessária uma abordagem alternativa ao cálculo de $\mathsf{Sp}(f)$ e, para isso, é necessário introduzir uma representação de espectros inspirada nos *Binary Decision Diagrams* ou BDD's e uma interpretação desses diagrama análoga à usada no método de *Davis-Purtnam*.

Começa-mos por um exemplo.

EXEMPLO 22: Consider-se a função booleana em $\mathrm{GF}(2)^4$

$$f(x_0, x_1, x_2, x_3) = 1 + x_0 \cdot x_1 + x_0 \cdot x_2 + x_1 \cdot x_2 \cdot x_3$$

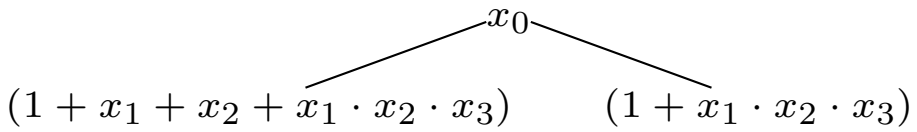
Pode-se sempre escrever

$$f(x_0, x_1, x_2, x_3) \;=\; x_0 \cdot f(1, x_1, x_2, x_3) + (1 + x_0) \cdot f(0, x_1, x_2, x_3)$$

ou seja, com alguma manipulação

$$= x_0 \cdot (1 + x_1 + x_2 + x_1 \cdot x_2 \cdot x_3) + (1 + x_0) \cdot (1 + x_1 \cdot x_2 \cdot x_3)$$

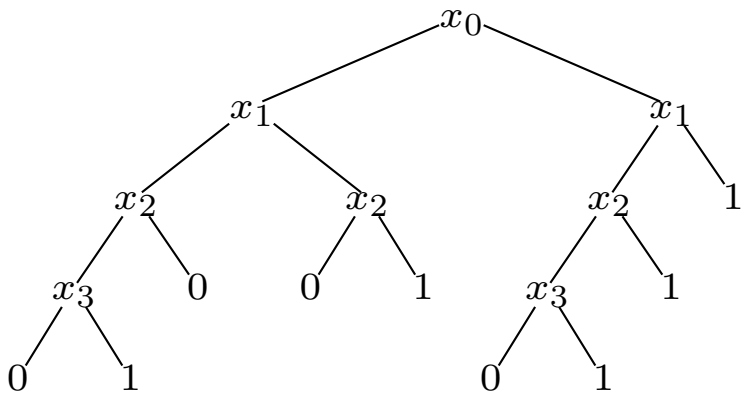
O que sugere uma representação arbórea



Repetindo o processo para as restantes variáveis, vemos que

$$\begin{aligned}(1 + x_1 + x_2 + x_1 \cdot x_2 \cdot x_3) &= x_1 \cdot x_2 \cdot (1 + x_3) + (1 + x_1) \cdot (1 + x_2) \\ (1 + x_1 \cdot x_2 \cdot x_3) &= x_1 \cdot (x_2 \cdot (1 + x_3) + (1 + x_2) \cdot 1) + (1 + x_1) \cdot 1\end{aligned}$$

O que sugere a seguinte árvore



Note-se que os percursos iniciados na raiz determinam valores de x e os respectivos valores $f(x)$ consoante a folha onde os percursos terminam. Os percursos são determinados pelas regras muito simples: $x_i = 1$ significa "*virar à esquerda* no nodo x_i", enquanto $x_i = 0$ significa "*virar à direita*.no nodo x_i".

Nesta árvore, por exemplo, os seguintes percursos terminam no valor 0

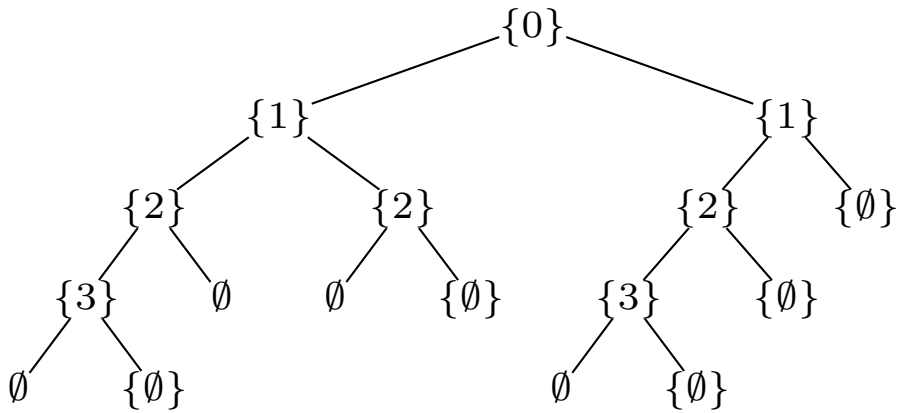
$$\{x_0 = 1\ ,\ x_1 = 0\ ,\ x_2 = 1\ ,\ x_3 = ?\} \quad , \quad \{x_0 = 1\ ,\ x_1 = 1\ ,\ x_2 = 0\ ,\ x_3 = ?\}$$
$$\{x_0 = 1\ ,\ x_1 = 1\ ,\ x_2 = 1\ ,\ x_3 = 1\} \quad , \quad \{x_0 = 0\ ,\ x_1 = 1\ ,\ x_2 = 1\ ,\ x_3 = 1\}$$

Todos os restantes percursos terminam no valor 1. Isto diz-nos que

$$f(1,0,1,0) = f(1,0,1,1) = f(1,1,0,0) =$$
$$= f(1,1,0,1) = f(1,1,1,1) = f(0,1,1,1) = 0$$

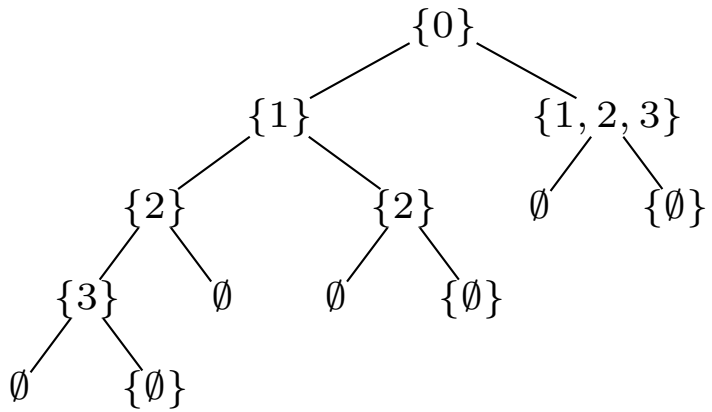
Uma árvore equivalente à anterior pode ser construída colocando nos nodos e nas folhas os espectros das expressões

que ocorrem na árvore anterior.



Admitindo que os nodos podem ter quaisquer índices (e não apenas índices singulares) a árvore pode-se simplificar

para



Esta árvore permite, agora, reconstruir o espectro da função inicial. De facto é fácil de verificar:

(i) Se a árvore for uma folha o espectro é o que a folha indica: $\emptyset$ ou $\{\emptyset\}$.

(ii) Se a árvore for um triplo $\alpha = \langle \sigma \ , \ \alpha^+ \ , \ \alpha^- \rangle$, o seu espectro $\mathcal{A}$ é

$$\mathcal{A} \ = \ \mathcal{A}^- \uplus \{\sigma\} \barwedge (\mathcal{A}^+ \uplus \mathcal{A}^-)$$

sendo $\mathcal{A}^+, \mathcal{A}^-$ os espectros representados pelas sub-árvores α^+ e α^-.

Para sistematizar esta construção seja $\mathbb{U}_{k,n} \doteq \wp(\mathbb{Z}_n \setminus \mathbb{Z}_k)$; isto é, o conjunto de todos os índices formado por elementos $k \leq i < n$.

155 DEFINIÇÃO
[35] *Dado um polinómio reduzido (sem monómios repetidos)* $f = \sum_{u \in \mathcal{A}} x_u$ *com* $\mathcal{A} \subseteq \mathbb{U}_{k,n}$ *o* **fraccionamento** *de* f *em* $\mathbb{U}_{k,n}$ *é:*

1. *Se* f *é uma função constante, o fraccionamento coincide com a própria função.*
2. *Se* f *não é constante (tem grau maior do que* 0*), sejam:*
   (i) $f^-(x_{k+1}, \cdots, x_{n-1})$ *o polinómio que se obtém eliminando de* f *todos os monómios que contenham* x_k*; seja* α^- *o seu fraccionamento em* $\mathbb{U}_{k+1,n}$ *determinado por este processo.*
   (ii) $f^+(x_{k+1}, \cdots, x_{n-1})$ *o polinómio que se obtém de* f *eliminando* x_k *de todos os seus monómios e reduzindo o resultado final através da eliminação de pares de monómios iguais; seja* α^+ *o seu fraccionamento em* $\mathbb{U}_{k+1,n}$ *.*

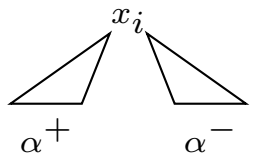
   Se $f^- = f^+$*, o fraccionamento* α *de* f *coincide com* $\alpha^+ = \alpha^-$*; em caso contrário, é o triplo* $\alpha = \langle x_k \ , \ \alpha^+ \ , \ \alpha^- \rangle$ *.*

   A **ordem** *do fraccionamento é* 0 *se* f *for constante ou, em caso contrário, é* $(n - i)$ *sendo* i *o índice da variável que constitui o primeiro elemento deste triplo.*

Esta definição sugere imediatamente uma representação arbórea para o fraccionamento de uma função cujo espectro está contido em $\mathbb{U}_{k,n}$. As funções de grau zero serão as folhas da árvore. As funções não constantes têm fraccionamentos que são triplos

[35] Baseada na noção de fraccionamento ("*split*")de Davis-Putnam.

$\langle x_i \ , \ \alpha^+ \ , \ \alpha^- \rangle$ (com $i \geq k$) formados por uma variável e dois outros fraccionamentos.

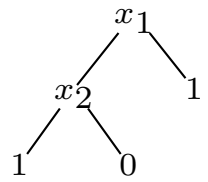


Se a função não for constante existe pelo menos um monómio com uma ou mais variáveis. Não é necessário que contenha a variável x_k; por exemplo, para $k = 0$ e $n = 3$, considere-se a função $f(x_0, x_1, x_2) = 1 + x_1 + x_1 \cdot x_2$.

$f(x_0, x_1, x_2)$ tem 3 monómios nenhum dos quais contém x_0. O cálculo de f^+ e de f^-, com o fraccionamento feito em $\mathbb{U}_{0,3}$, não modifica f uma vez que não contém x_0; teremos, assim, $f = f^+ = f^-$.

Porém, quando se efectua o fraccionamento em $\mathbb{U}_{1,3}$, tem-se $f^- = 1$ e $f^+ = 1 + 1 + x_2$ que, após redução, se simplifica em x_2.

O fraccionamento final de f está representado ao lado e, como vemos, tem ordem 2.



156 TEOREMA

Nas condições da definição anterior.

1. *Os polinómios f^+ e f^- estão reduzidos.*

2. *A condição* $f^+ = f^-$ *verifica-se se e só se* f *não contém* x_k *em nenhum dos seus monómios. Neste caso tem-se* $f^+ = f^- = f$.
3. *Sendo* $\mathcal{A}^+, \mathcal{A}^- \in \mathbb{U}_{k+1,n}$ *os espectros de* f^+ *e* f^- *então*

$$\mathcal{A}^- = \{\, u \mid u \in \mathcal{A} \ \&\ k \notin u \,\}$$

$$\mathcal{A}^+ = \{\, u \setminus \{k\} \mid u \in \mathcal{A} \,\}$$

$$\mathcal{A} = \mathcal{A}^- \uplus (\mathcal{A}^+ \uplus \mathcal{A}^-) \uplus \{\{k\}\}$$

4. *Verifica-se*

$$f^-(x_{k+1}, \cdots, x_{n-1}) = f(0, x_{k+1}, \cdots, x_{n-1})$$

$$f^+(x_{k+1}, \cdots, x_{n-1}) = f(1, x_{k+1}, \cdots, x_{n-1})$$

$$f = x_k \cdot f^+ + (1 + x_k) \cdot f^-$$

Sejam f, g duas funções booleanas e α, β os respectivos fraccionamentos em algum $\mathbb{U}_{k,n}$. Representemos por $(\alpha + \beta)$, $(\alpha \cdot \beta)$, $\alpha^{(z)}$ os fraccionamentos, respectivamente, das funções $(f + g)$, $(f \cdot g)$, $f^{(z)}$.

157 FACTO
Verifica-se:

1. $(\alpha + 0) = (\alpha \cdot 1) = \alpha$, $(\alpha \cdot 0) = 0$, $0^{(z)} = 0$, $1^{(z)} = 1$, $\alpha \cdot \alpha = \alpha$ *e* $\alpha + \alpha = 0$.
2. *Redução:* $\langle x \ , \ \alpha \ , \ \alpha \rangle$ *simplifica em* α .
3. *Se for* $\alpha = \langle x_k \ , \ \alpha^+ \ , \ \alpha^- \rangle$ *e* β *é um fracionamento de ordem inferior à de* α*, então*

$$(\alpha + \beta) = \langle x_k \ , \ \alpha^+ + \beta \ , \ \alpha^- + \beta \rangle$$
$$(\alpha \cdot \beta) = \langle x_k \ , \ \alpha^+ \cdot \beta \ , \ \alpha^- \cdot \beta \rangle$$

4. *Se* α *e* β *têm a mesma ordem e se for* $\alpha = \langle x_k \ , \ \alpha^+ \ , \ \alpha^- \rangle$ *e* $\beta = \langle x_k \ , \ \beta^+ \ , \ \beta^- \rangle$ *, então*

$$(\alpha + \beta) = \langle x_k \ , \ \alpha^+ + \beta^+ \ , \ \alpha^- + \beta^- \rangle$$
$$(\alpha \cdot \beta) = \langle x_k \ , \ \alpha^+ \cdot \beta^+ \ , \ \alpha^- \cdot \beta^- \rangle$$

5. *Se for* $\alpha = \langle x_k \ , \ \alpha^+ \ , \ \alpha^- \rangle$ *, então*

$$\alpha^{(z)} = \begin{cases} \langle x_k \ , \ (\alpha^+)^{(z)} \ , \ (\alpha^-)^{(z)} \rangle & \text{se} \quad z_k = 0 \\ \langle x_k \ , \ (\alpha^-)^{(z)} \ , \ (\alpha^+)^{(z)} \rangle & \text{se} \quad z_k = 1 \end{cases}$$

□

Frequentemente as funções boolenas aparecem agrupadas em vectores. Uma função booleana vectorial $\mathbf{S} : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^n$ pode ser descrita por um vector de n funções booleanas escalares

$$\mathbf{S}(x_1, x_2, \cdots, x_n) \quad = \quad \begin{bmatrix} h_1(x_1, x_2, \cdots, x_n) \\ h_2(x_1, x_2, \cdots, x_n) \\ \cdots \\ h_n(x_1, x_2, \cdots, x_n) \end{bmatrix}$$

Na terminologia criptográfica, uma tal função é designada por uma $n \times n$ **S-Box**.

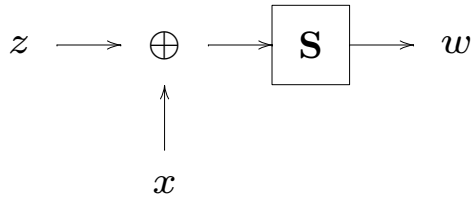
Facilmente se generaliza o conceito para S-Boxes não quadradas: uma $n \times m$ **S-Box** é uma função $\mathbf{S} : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^m$ definida por um vector de m componentes em que, cada uma, é uma função $h_i : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$, com $i \in 0..m-1$.

EXEMPLO 23: As três funções boolenas

$$\begin{bmatrix} h_0(x) = & 1 + x_0 \cdot x_1 \\ h_1(x) = & x_0 \cdot x_2 \cdot (1 + x_1) \\ h_2(x) = & 1 + x_0 + x_1 + x_2 \end{bmatrix}$$

definem uma SBox quadrada 3×3.

O exemplo mais corrente desta representação pode ser descrito pela figura seguinte



(62)

Pode-se ver x como uma chave, z como o texto de uma mensagem a cifrar e w como o criptograma resultante.

Problemas directos:

(I) determinar se existe algum valor de x que seja solução da equação

$$\mathbf{S}(z \oplus x) = w \tag{63}$$

(II) Caso exista gerar aleatoriamente **uma** solução

(III) Caso existam, enumerar **todas** as soluções.

O problema de tipo **I** procura saber, apenas, se existe uma chave, o problema de tipo **II** procura descobrir uma chave e o problema de tipo **III** procura enumerar todas as chaves.

EXEMPLO 24: Recuperemos a **SBox** 3×3 do exemplo 23 e suponhamos o seguinte par *entrada-saída*

$$z = (1, 0, 1) \qquad w = (0, 1, 0)$$

A equação que resulta de (65) (com $w_0 = 0, w_1 = 1, w_2 = 0$) será

$$h_0(z \oplus x) \cdot (1 + h_1(z \oplus x)) \cdot h_2(z \oplus x) = 1$$

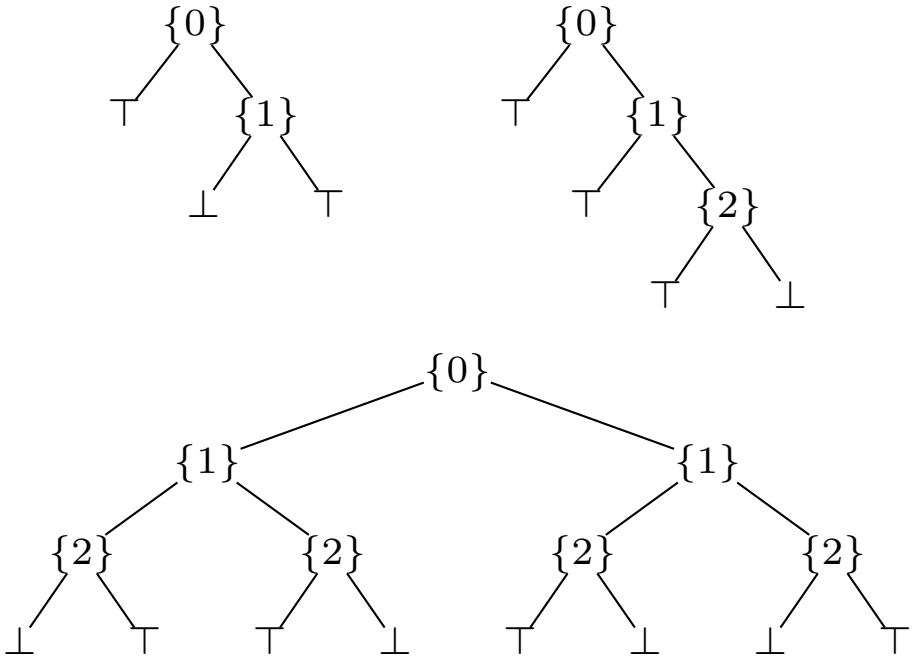
expandindo

$$(1 + (1 + x_0) \cdot x_1) \cdot (1 + (1 + x_0) \cdot (1 + x_1) \cdot (1 + x_2)) \cdot$$
$$\cdot (1 + (1 + x_0) + x_1 + (1 + x_2)) = 1$$

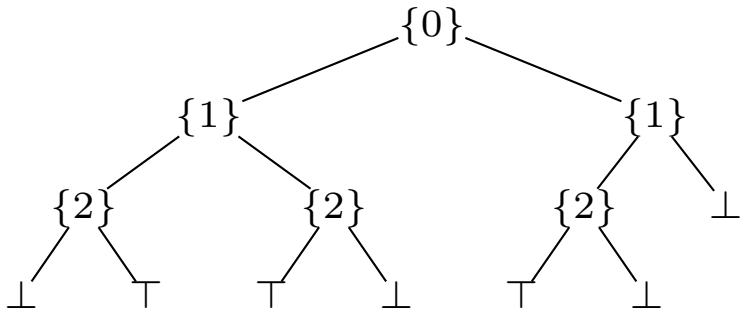
simplificando

$$(1 + x_1 + x_0 \cdot x_1) \cdot (1 + (1 + x_0) \cdot (1 + x_1) \cdot (1 + x_2)) \cdot (1 + x_0 + x_1 + x_2) = 1$$

Os espectros dos três factores nesta equação serão (abrev. $\top \equiv \{\emptyset\}$, $\bot \equiv \emptyset$),



Usando as regras no facto 157 o espectro resultante será



(64)

Esta árvore permite resolver os problemas (I), (II) e (III), neste caso, usando o resultado seguinte

158 FACTO

Seja $f : \mathrm{GF}(2) \to \mathrm{GF}(2)$ *uma função booleana e* α *o seu fraccionamento em* $\mathbb{U}_n$ *reduzido (não contém componentes da forma* $\langle u\ ,\ \beta\ ,\ \beta\rangle$ *) e construído segundo as regras do facto 157. Então:*

1. $\mathrm{supp}(f) = \emptyset$ *se e só se* $\alpha \equiv \bot$.
2. $x \in \mathrm{supp}(f)$ *se e só determina um caminho válido em* α *de acordo com as seguintes regras:*
   (a) *Numa folha* $\top$, x *determina o caminho válido vazio* ε *; não existe caminho válido numa folha* $\bot$.
   (b) *O caminho válido determinado por* x *em* $\langle u\ ,\ \alpha^+\ ,\ \alpha^-\rangle$ *(caso exista) é a sequência* $u^s\,\omega$, *em que* $s = +$, *se for* $x_u = 1$, *e* $s = -$, *se for* $x_u = 0$, *e* ω *é o caminho válido determinado por* x *em* α^s .

Notas: O que este resultado nos diz é que, basicamente, o fraccionamento α é uma representação do conjunto supp(f) que é computacionalmente conveniente: as regras fornecem um algoritmo para determinar se o conjunto é ou não vazio e, caso não seja vazio, o valor lógico da relação $x \in \text{supp}(f)$.

Conhecido o fraccionamento α se se procurar soluções para os problemas básicos, teremos:

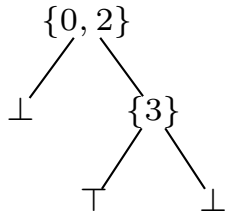
1. *Problema de tipo **I**:* saber se kwd$(f) = \emptyset$ reduz-se a saber se $\alpha = \bot$.
2. *Problema de tipo **II**:* se $\alpha \neq \bot$ encontrar um elemento arbitrário $x \in \text{kwd}(f)$ resolve-se gerando aleatoriamente um caminho válido em α. Isto significa criar um caminho que se inicie na raiz de α e siga, com igual probabilidade, por qualquer dos seus sub-fraccionamentos distintos de $\bot$.
3. *Problema de tipo **III**:* gerar todo o conjunto kwd(f) é equivalente a gerar todos os caminhos válidos em α. Computacionalmente, devido ao resultado anterior, não existe distinção entre α e o suporte de f; por isso, gerar kwd(f) é, essencialmente, construir o fraccionamento α.

É importante notar que diferentes valores de x podem determinar o mesmo caminho ω em α; isto acontece sempre que a árvore não está "cheia", i.e., quando o caminho não contém todos os possíveis índices $i \in 0..n-1$.

Tome-se, como exemplo, o seguinte fraccionamento em $\mathbb{U}_4$ (índices $0..3$) cujo o único caminho válido é

$$\{0,2\}^- \ \{3\}^+$$

Isto significa que qualquer x que pertença ao suporte da função tem de verificar $x_0 = x_2 = 0$ e $x_3 = 1$.



Note-se que o índice 1 não ocorre no caminho; isto significa que a componente x_1 não está fixa a nenhum dos valores 0 ou 1; representemos este facto pela relação $x_1 = ?$. O "pseudo- valor" $?$ é conhecido por *"do not care"* e, com esta notação, pode-se escrever o elemento x do suporte como $x = (0, ?, 0, 1)$.

Existe um outro "pseudo-valor" importante, representado pelo símbolo !, que indica que uma variável booleana está, simultaneamente, obrigada a ter um valor 0 e obrigada a ter um valor 1. Exemplo, $x = (!, 1, !, 1)$ indicaria que todas as componentes são fixas em 1 e, adicionamento, a primeira e terceira estão fixas em 0.

Numa máquina determinística típica isto é impossível e isso conduziria, naturalmente, a um resultado não-válido para uma computação que produzisse este pseudo-valor como resultado parcial. No entanto é possível ter modelos da computação onde um resultado ! seja perfeitamente válido; por exemplo, em modelos de computação quântica ou, genericamente, computação não-determinística.

Escrita de outro modo, a equação (63) é $\delta(w \oplus \mathbf{S}(z \oplus x)) = 1$, ou, equivalentemente,

$$\delta^{(w)}(\mathbf{S}^{(z)}(x)) = 1 \qquad \text{ou} \qquad [h^{(z)}]_{\bar{w}} = 1 \tag{65}$$

Podemos definir uma função boolena

$$\mathcal{Q}_{z,w}(x) \doteq \delta^{(w)}(\mathbf{S}^{(z)}(x)) \tag{66}$$

e a solução de um problema directo (procurar a chave x) como um ataque a $\mathcal{Q}_{z,w}$.

Uma computação tratável φ que produza uma solução para um problema directo do tipo **II** designa-se por **ataque directo** ao triplo $\langle \mathbf{S}, z, w \rangle$. A entropia de $\mathcal{Q}_{z,w}$ extendida aos ataques directos

$$\mathcal{H}(\mathcal{Q}_{z,w}) = \min_{\varphi} \mathcal{H}(\varphi) - \log_2 \{\mathcal{Q}_{z,w}\}_{\varphi}$$

é a **entropia directa** de $\langle \mathbf{S}, z, w \rangle$.

Nota Recordemos que $\{\mathcal{Q}_{z,w}\}_{\varphi}$ representa a probabilidade da computação φ produzir um resultado x tal que $\mathcal{Q}_{z,w}(x) = \delta(w \oplus \mathbf{S}(z \oplus x)) = 1$.

Tendo agora em atenção o facto (158) vemos que a complexidade computacional para encontrar essa solução x nas sua duas componentes (construir o fraccionamento α e um caminho qualquer nesse fraccionamento) tem uma complexidade computacional que é determinada, essencialmente, pela primeira componente.

A construção do fraccionamento, na pior das circunstâncias, pode ser exponencial com o número de *bits* e, por isso, dificilmente será considerada "tratável". Se, para um determinado triplo $\langle \mathbf{S}, z, w \rangle$ o cálculo do fraccionamento for tratável, então a probabilidade $\{\mathcal{Q}\}_{\varphi}$ é igual a 1 e a entropia reduz-se ao cálculo da menor entropia da computação φ que produz esse fraccionamento.

Deve-se ter em atenção que um ataque directo ao triplo $\langle \mathbf{S}, z, w \rangle$, com um par (z, w) particular, fornece muito pouca informação sobre $\mathbf{S}$; nomeadamente sobre a chave x se ela estiver a ser usada com outros pares *entrada-saída*. Por isso faz sentido definir uma forma mais realista de ataque.

Ataque 1 Seja μ um sub-conjunto de cardinalidade N do conjunto

$$\Omega \doteq \{ (z, w) \mid \mathbf{S}(z \oplus k) = w \} \tag{67}$$

Um **ataque directo** a $\mathbf{S}$ de dimensão N é uma computação tratável que, conhecidos $\mathbf{S}$ e μ, determina k.

Notas O ataque ao triplo $\langle \mathbf{S}, z, w \rangle$ fornece aleatoriamente uma solução x possível para k; se estiver em causa apenas um par (z, w) qualquer solução x serve. É uma ataque directo a $\mathbf{S}$ de dimensão 1.

Porém se este par for apenas um dos elementos de μ a solução x encontrada é apenas um dos valores possíveis para k. Sabe-se que k, solução do ataque directo, é um dos valores possíveis do suporte de $\mathcal{Q}_{z,w}$; mas não sabemos qual é. O valor x relaciona-se com k apenas pelo facto de serem ambos elementos desse suporte.

Pode-se construir uma função boolena, análoga a (66), que representa o facto de, para <u>todos</u> os pares $(z, w) \in \mu$, se verificar $\mathcal{Q}_{z,w}(x) = 1$

$$\begin{aligned} \mathcal{Q}_\mu(x) &\doteq \Pi_{(z,w)\in\mu}\, \mathcal{Q}_{z,w}(x) = \\ &= \Pi_{(z,w)\in\mu}\, \delta(w \oplus \mathbf{S}(z \oplus x)) \end{aligned} \tag{68}$$

Uma computação φ que encontre um x tal que $\mathcal{Q}_\mu(x) = 1$ não "acerta" necessariamente em k. A probabilidade de se ter $x = k$ é determinada pelo tamanho do suporte da função $\mathcal{Q}_\mu$; isto é, pelo *peso* dessa função.

Portanto a probabilidade de uma computação determinística ϕ, que tome o resultado de φ e o considere um resultado para k, é dado pela razão entre o número de possíveis k se tivéssemos toda a informação possível (isto é o peso da função $\mathcal{Q}_\Omega$) e o número de hipótese de resultados de φ dados pelo peso de $\mathcal{Q}_\mu$.

$$-\log_2 \left\{ \mathcal{Q}_\Omega : \mathcal{Q}_\mu \right\}_\phi \;=\; -\log_2 \mathrm{wt}(\mathcal{Q}_\Omega) + \log_2 \mathrm{wt}(\mathcal{Q}_\mu)$$

A entropia de ϕ é 0 porque é determinística; por isso, usando pode-se escrever a relação que dá a entropia do ataque directo à SBox $\mathbf{S}$

$$\mathcal{H}(\mathcal{Q}_\Omega) \;=\; \min_\mu \left(\mathcal{H}(\mathcal{Q}_\mu) + \log_2 \mathrm{wt}(\mathcal{Q}_\mu) - \log_2 \mathrm{wt}(\mathcal{Q}_\Omega) \right) \tag{69}$$

No caso particular em que só existe um k possível e a solução x para $\mathcal{Q}_\mu)(x) = 1$ pode ser encontrada deterministicamente por uma computação tratável (i.e. $\mathcal{H}(\mathcal{Q}_\mu) = 0$) ainda resta a componente $\log_2 \mathrm{wt}(\mathcal{Q}_\mu)$.

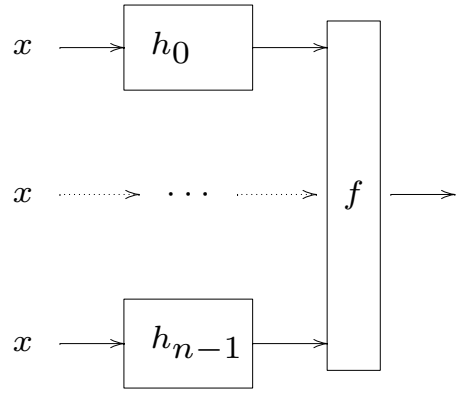
3.2 Composição de funções booleanas

Considere-se uma uma S-Box $k \times n$, $H : \mathrm{GF}(2)^k \rightarrow \mathrm{GF}(2)^n$. Pode-se escrever $H = (h_0, \ldots, h_{n-1})$ em que os vários h_i são funções boolenas de k argumentos.

$$h_i : \mathrm{GF}(2)^k \rightarrow \mathrm{GF}(2)$$

Seja $f : \mathrm{GF}(2)^n \rightarrow \mathrm{GF}(2)$ uma outra função boolena. É possível construir uma computação em que os resultados das n funções h_i são usados como argumentos de f; i.e, uma computação da forma

$$f(h_0(x), \ldots, h_{n-1}(x))$$



Fica definida uma nova função boolena (com k argumentos) a que chamamos **composição** de f e H e que representamos por

$$f \circ H \quad \text{ou} \quad H; f$$

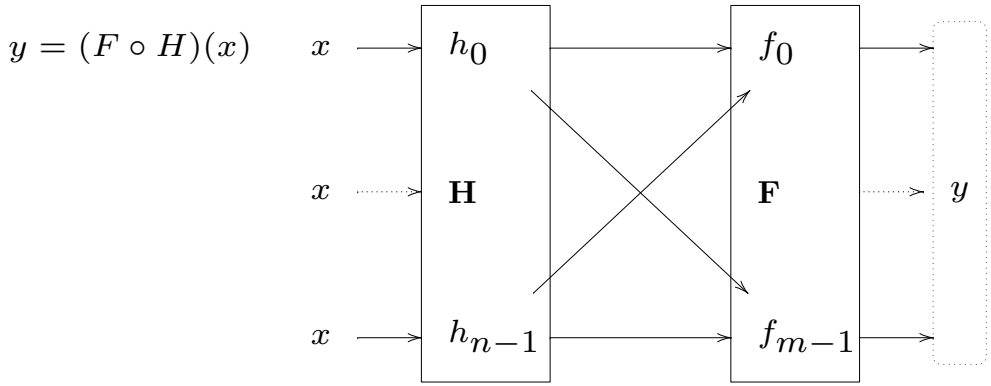
□

Considerando agora, não só uma função f, mas m funções deste tipo (formando uma S-Box de dimensão $n \times m$)

$$F = (f_0 \ , \ f_1 \ , \ \cdots \ , \ f_{m-1}) \qquad \text{com} \quad f_i : \mathrm{GF}(2)^n \rightarrow \mathrm{GF}(2)$$

então, através da composição de cada um dos f_i com H, constrói-se o vector de funções boolenas que define a **composição** das S-Boxes F e H.

$$F \circ H \doteq (f_0 \circ H \ , \ f_1 \circ H \ , \ \cdots \ , \ f_{m-1} \circ H)$$



Para exprimir o espectro destas composições é necessário aplicação a noção se selecção apresentada na definição 150 (página 184) a S-Boxes.

159 DEFINIÇÃO
Seja H uma S-Box $n \times m$; para $u \in \mathbb{U}_m$, as funções booleanas H_u e $[H]_u$ definem-se por

$$H_u = \Pi_{i \in u} \, h_i \qquad [H]_u = H_u \cdot \Pi_{i \notin u} \, (1 + h_i)$$

Deste modo, se f tem espectro $\mathcal{A}$, será $f(y) = \sum_{u \in \mathcal{A}} y_u$. Se for $y = H(x)$ teremos

$$(f \circ H)(x) \;=\; f(y) \;=\; \sum_{u \in \mathcal{A}} H_u(x)$$

Representemos por $\mathcal{B}_i$ o espectro da função boolena h_i. Então o espectro de $H_u = \Pi_{i \in u}\, h_i$ será $\left(\bigoplus_{i \in u} \mathcal{B}_i\right)$; usando as regras atrás definidas para produtos de espectros, este cálculo é polinomial em n.

O espectro de $(f \circ H)$ pode, agora, ser calculado como

$$\biguplus_{u \in \mathcal{A}} \left(\bigoplus_{i \in u} \mathcal{B}_i \right) \tag{70}$$

Funções Lineares

Infelizmente, no caso geral, a fórmula (70) tem pouco interesse computacional porque obriga a percorrer todo u no espectro de f que, em princípio, cresce exponencialmente com n.

No entanto, para algumas formas particulares de f, o espectro $\mathcal{A}$ assume formas que tornam simples o cálculo (70). Nomeadamente quando f é uma **função linear**.

Qualquer função linear $f : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$ verifica

$$f(x \oplus y) = f(x) + f(y) \quad \text{para todo} \quad x, y \in \mathrm{GF}(2)^n$$

e o seu espectro é formado por conjuntos com um único índice.

Neste caso (70) resume-se a $\biguplus_{\{i\} \in \mathcal{A}} \mathcal{B}_i$ que pode ser calculado com complexidade polinomial.

Uma função linear importante é a **função traço** definida por

$$\mathrm{Tr}(x_0, x_1, \cdots, x_{n-1}) \doteq x_0 + x_1 + \cdots + x_{n-1} \tag{71}$$

cujo espectro é o conjunto $\{ \{i\} \mid i \in 0..n-1 \}$ formado por todos os índices singulares.

Esta função determina a **paridade** do vector $x = (x_0, \cdots, x_{n-1})$; i.e. a paridade do número de componmentes iguais a 1 no vector x.

Esta noção de paridade pode ser generalizada. Tomemos uma função linear f qualquer e o vector constante c^f que tem componentes a 1 exactamente para os índices onde f tem monómios; isto é,

$$(c^f)_u = 1 \qquad \text{se e só se} \quad u \in \mathrm{Sp}(f) \tag{72}$$

Designamos c^f por **máscara** ou **paridade** de f . Nessas circunstâncias

160 FACTO

Se f *é linear e tem máscara* c^f *, então* $f(x) = \mathsf{Tr}(x * c^f)$ *para todo* $x \in \mathsf{GF}(2)^n$.

Exemplo: se for $f(x) = x_0 + x_2 + x_7$, com $x \in \mathsf{GF}(2)^8$, então a máscara é

$$c^f = (1, 0, 1, 0, 0, 0, 0, 1)$$

Note-se que $x * c^f$ é o vector $(x_0, 0, x_2, 0, 0, 0, 0, x_7)$; o seu traço constrói precisanente o valor de $f(x)$.

3.3 Diferenças e Linearidade

Considere-se de novo uma S-Box $\mathbf{S}$ e o problema definido em (62) e (63) de determinar a chave x tal que $\mathbf{S}(z \oplus x) = w$.

Vamos supor que $\mathbf{S}$ era tal que existia uma solução, pelo menos, para o seguinte problema:

Diferenças críticas: encontrar $a, b \in \mathrm{GF}(2)^n$ tais que,

$$\mathbf{S}(a \oplus y) \oplus \mathbf{S}(y) = b \qquad \text{para todo} \quad y \in \mathrm{GF}(2)^n \tag{73}$$

Os elementos (a, b) chamam-se **diferenças críticas**.

Se for possível encontrar um ou mais pares de difenças críticas (a, b) então qualquer cifra assente na complexidade computacional de inverter **S** perde entropia.

Por exemplo, o ataque directo a $\mathbf{S}$ tem a seguinte variante:

Ataque 2 Criptoanálise Diferencial do Ataque Directo

1. Recolher pares (z_j, w_j) com $w_j = \mathbf{S}(z_j \oplus k)$; a chave k é a mesma em todos os pares e é a incógnita deste ataque.
2. Recolher pares de diferenças críticas (a_i, b_i) .
3. Encontrar x' tal que, para algum i e todos os j , se verifique $\mathbf{S}(z_j + x) = w_j \oplus b_i$.
4. Nestas circunstâncias tem-se $x \doteq a_i \oplus x'$ é uma solução eventual para k .

A justificação reside no facto de, sendo (a_i, b_i) uma par de diferenças críticas, verifica-se $\mathbf{S}(a_i \oplus (z_j \oplus x')) \oplus \mathbf{S}(z_j \oplus x') = b_i$.

Por construção tem-se, para todo j , $\mathbf{S}(z_j \oplus x') = w_j \oplus b_i$; escolhendo $x = x' \oplus a_i$ verifica-se, para todo j, $\mathbf{S}(z_j \oplus x) \oplus (w_j \oplus b_i) = b_i$; donde $\mathbf{S}(z_j \oplus x) = w_j$ para todo j ; isto significa que x é, eventualmente, k .

Note-se que:

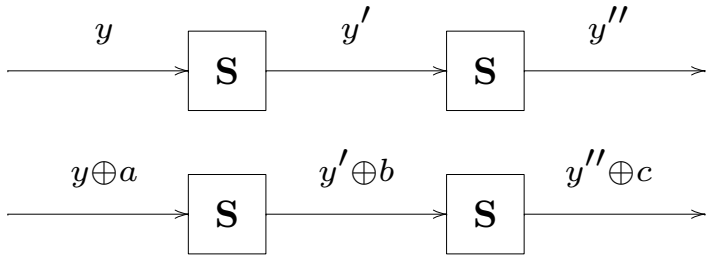
1. O passo (3) é, essencialmente, uma solução do problema inicial mas para valores diferente de *outupts*; é uma versão do problema inicial para pares $(z_j, w_j \oplus b_i)$.
   Aparentemente não se ganha nada com esta formalização dado que, como parte do problema inicial, temos de resolver um problema do mesmo tipo.
2. O ganho reside apenas no facto de ser possível adaptar o problema a uma escolha criteriosa de valores b_i caso seja possível calcular o respectivo a_i .

É possível resolver um ataque directo para diferentes conjuntos de pares $\mu_i = \left\{ (z_j, w_j \oplus b_i) \mid j \in 1.. \right\}$ e seleccionar a solução com menor entropia.

No entanto os ataques mais importantes resultam da aplicação das diferenças críticas à composição de S-Boxes.

Suponhamos um caso simples com a composição de $\mathbf{S}$ consigo próprio e vamos supor que (a, b) e (b, c) são dois pares de diferenças críticas.

Consider-se a seguinte situação em que a S-Box resultante é aplicada a duas *entradas* diferentes: y e $y \oplus a$.



(74)

Pelo propriedade das diferenças críticas, conhecidos os vários valores y, y' e y'' no primeiro caso, é muito simples determinar os valores respectivos quando a *entrada* muda de y para $y \oplus a$:

(i) O resultado intermédio muda de y' para $y' \oplus b$ porque o primeiro par de diferenças críticas me diz que $\mathbf{S}(y \oplus a) \oplus \mathbf{S}(y) = b$; logo, $\mathbf{S}(y \oplus a) = \mathbf{S}(y) \oplus b = y' \oplus b$.

(ii) O resultado final muda de y'' para $y'' \oplus c$ porque o par de diferenças críticas (b, c) diz-nos que $\mathbf{S}(y' \oplus b) \oplus \mathbf{S}(y') = c$.
A entrada do 2º $\mathbf{S}$ mudou de y' para $y' \oplus b$; a diferença crítica c repflecte-se, agora, na respectiva saída.

Por este facto se diz que

o par de diferenças críticas (a, b) **propaga** *a diferença* a, *na entrada, para a diferença* b *na saída,*

A propagação de diferenças diminui consideravelmente a entropia de uma concatenação de várias S-Boxes. O aumento de entropia que devia resultar dessa concatenção acaba por se não se manifestar dado que muita da complexidade se perde com a existência de diferenças críticas.

A propósito, o exemplo anterior demonstra o seguinte facto

161 FACTO
Se (a, b) *é um par de diferenças críticas para* $\mathbf{S}$ *e* (b, c) *é um par de diferenças críticas para* $\mathbf{R}$ *então* (a, c) *é um par de diferenças críticas para* $\mathbf{R} \circ \mathbf{S}$.

□

É possível exprimir diferenças críticas através de uma função booleana. Representemos por $\Delta^{\mathbf{S}}_{a,b}$ a função que, para todo o argumento y, verifica

$$\Delta^{\mathbf{S}}_{a,b}(y) \;=\; \delta(\mathbf{S}(a \oplus y) \oplus \mathbf{S}(y) \oplus b) \tag{75}$$

e representemos por

$$\rho_{\mathbf{S}}(a,b) \;\doteq\; 2^{-n} \cdot \mathsf{wt}(\Delta^{\mathbf{S}}_{a,b})$$

a razão entre número de argumentos y para os quais $\Delta^{\mathbf{S}}_{a,b}(y) = 1$ e o número total de argumentos possíveis.

Note-se que (a,b) é um par de diferenças críticas se e só se $\Delta^{\mathbf{S}}_{a,b}$ é a função constante 1. Ou seja, quando $\rho_{\mathbf{S}}(a,b) = 1$.

Porém pode acontecer que (a,b) não seja um par de diferenças críticas mas, ainda assim, exista a possibilidade de, para a maioria dos possíveis argumentos y, se verificar $\Delta^{\mathbf{S}}_{a,b}(y) = 1$; isto significa que será $\rho_{\mathbf{S}}(a,b) < 1$ mas, ainda assim, próximo do limite 1 .

Recordemos a noção de entropia; a notação $\left\{\Delta^{\mathbf{S}}_{a,b}\right\}_{\varphi}$ representa a probabilidade de a computação φ gerar um resultado y que verifique $\Delta^{\mathbf{S}}_{a,b}(y) = 1$.

Se (a,b) fosse um para de diferenças críticas esta probabilidade seria sempre 1 qualquer que seja a computação φ que produza resultados em $\mathsf{GF}(2)^n$.

É possível exprimir a entropia de $\Delta^{\mathbf{S}}_{a,b}$ relativo a φ.

$$\mathcal{H}(\Delta^{\mathbf{S}}_{a,b})_{\varphi} \doteq \mathcal{H}(\varphi) - \log_2 \left\{\Delta^{\mathbf{S}}_{a,b}\right\}_{\varphi}$$

Se φ for determinística será $\mathcal{H}(\varphi) = 0$. Se se comportar como um gerador aleatório, a probabilidade de "acertar" num elemento so suporte de $\Delta^{\mathbf{S}}_{a,b}$ é $\rho_{\mathbf{S}}(a,b)$. Nestas circunstâncias particulares (para um tal φ) será

$$\mathcal{H}(\Delta^{\mathbf{S}}_{a,b})_{\varphi} = -\log_2 \rho_{\mathbf{S}}(a,b) \qquad \text{ou} \tag{76}$$

$$\rho_{\mathbf{S}}(a,b) = 2^{-\mathcal{H}(\Delta^{\mathbf{S}}_{a,b})_{\varphi}}$$

Estas relações definem uma aproximação para a perda de entropia introduzida por um candidato a par de diferenças críticas quando não existe informação sobre a função de diferenças $\Delta^{\mathbf{S}}_{a,b}$ para além do seu suporte.

Facilmente se generaliza o facto 161 para "candidatos" a pares de diferenças críticas

162 FACTO

$$\rho_{\mathbf{R}\circ\mathbf{S}}(a,c) \geq \rho_{\mathbf{S}}(a,b) \cdot \rho_{\mathbf{R}}(b,c)$$

□

A **distância de Hamming** entre duas funções $f, g : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ (representa-se por $d(f,g)$) é o peso de $f+g$; isto é, a $d(f,g)$ conta o número de argumentos nos quais as funções divergem. A **não-linearidade** de f é a menor das distâncias $d(f,g)$ quando g percorre todas as funções booleanas afins em $\mathrm{GF}(2)^n$.

A **correlação** entre f e g, é a função racional

$$\mathcal{C}(f,g) \doteq 1 - 2^{-(n-1)} \cdot d(f,g) \tag{77}$$

Nota:
A correlação tem um resultado compreendido entre -1 e 1; caso as funções sejam iguais temos $d(f,g)=0$ e $\mathcal{C}(f,g)=1$; se as funções forem totalmente distintas (i.e. $f=1+g$) então $d(f,g)=2^n$ e $\mathcal{C}(f,g)=-1$; se o número de argumentos x para os quais $f(x) \neq g(x)$ for metade do total, então $d(f,g)=2^{n-1}$ e $\mathcal{C}(f,g)=0$.

Convenção:
Vimos que cada função linear $\omega : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ é únicamente determinada por um elemento $c^\omega \in \mathrm{GF}(2)^n$ designado por máscara ou paridade ω. Se não surgirirem ambiguidades designaremos a função e a respectiva paridade pelo mesmo símbolo. Assim escrever $\omega \in \mathrm{GF}(2)^n$ e $\omega(x)$ são notações compatíveis: a primeira representa o vector paridade e a segunda faz referência à função linear que esse vector determina.

Alguns resultados intermédios

163 FACTO

Sejam ω, υ funções lineares e f uma outra função boolena qualquer. Então

(i) $\mathcal{C}(\omega, \upsilon) = \delta(\omega \oplus \upsilon)$

(ii) $\mathcal{C}(1 + f, \omega) = -\mathcal{C}(f, \omega)$

(iii) *Se* $g(x) = f(x) + \upsilon(x)$ *então* $\mathcal{C}(g, \omega) = \mathcal{C}(f, \omega + \upsilon)$

164 DEFINIÇÃO

Dada uma função boolena $f : \mathsf{GF}(2)^n \longrightarrow \mathsf{GF}(2)$ *e uma função linear* ω *, seja*

$$F(\omega) \doteq \mathcal{C}(f, \omega) \tag{78}$$

A função $F : \omega \mapsto F(w)$ *, fazendo* ω *percorrer todas as* 2^n *funções lineares realizáveis em* $\mathsf{GF}(2)^n$*, chama-se o* **espetro de Walsh** *de* f *.*

A transformação $\mathcal{W} : f \mapsto F$ *, que associa* f *ao seu espectro de Walsh, chama-se* **transformada de Walsh-Hadamard**.

165 FACTO

Para todo $f, g \in \mathsf{GF}(2)^n \longrightarrow \mathsf{GF}(2)$ *verifica-se*

$$\mathcal{C}(f, g) = 2^{-n} \cdot \sum_x (-1)^{f(x)+g(x)} \tag{79}$$

e, se $F \doteq \mathcal{W}(f)$, *verifica-se para todo* $x \in \mathsf{GF}(2)^n$

$$(-1)^{f(x)} = \sum_{\omega} F(\omega)\,(-1)^{\omega(x)} \tag{80}$$

em que a primeira soma se estende a todos $x \in \mathsf{GF}(2)^n$ *e a segunda soma a todas as funções lineares* $\omega \in \mathsf{GF}(2)^n$.

Se $G \doteq \mathcal{W}(g)$ *verifica-se* $\mathcal{W}(f+g) = F \otimes G$ *em que a* **convolução** *de espectros de Walsh é definida por*

$$(F \otimes G)(\omega) = \sum_{\upsilon} F(\omega \oplus \upsilon) \cdot G(\upsilon) \tag{81}$$

166 Definição
Seja $\mathbf{S}$ *uma SBox definida por um vector de funções boolenas* $(h_0, h_1, \cdots, h_{n-1})$. *Para quaisquer duas funções lineares* $\omega, \upsilon \in \mathsf{GF}(2)^n$ *seja*

$$\mathcal{C}^{\mathbf{S}}(\upsilon, \omega) \doteq \mathcal{C}(\upsilon \circ \mathbf{S}, \omega) \tag{82}$$

Vista como uma matriz $2^n \times 2^n$, *a função* $\mathcal{C}^{\mathbf{S}}$ *chama-se* **matriz de correlação** *de* $\mathbf{S}$.

167 Facto
Nas condições da definição anterior,

(i) $\mathcal{W}(\upsilon \circ \mathbf{S}) = \bigotimes_{\upsilon_i=1} \mathcal{W}(h_i)$.

(ii) *Se* f *é uma função boolena de espectro* F *e* $g = f \circ \mathbf{S}$ *então o seu espectro* G *pode ser calculado como*

$$G \;=\; F \times \mathcal{C}^{\mathbf{S}}$$

em que G *e* F *são vistos como vectores-linha de* 2^n *componentes e a operação* $\times$ *é vista como a multiplicação de matrizes.*

(iii) *Se* $\mathbf{R}$ *é uma outra S-Box então a matriz correlação da composição* $\mathbf{R} \circ \mathbf{S}$ *é o produto (no sentido da multiplicação de matrizes) das duas matrizes de correlação.*

$$\mathcal{C}^{\mathbf{R}\circ\mathbf{S}} \;=\; \mathcal{C}^{\mathbf{R}} \times \mathcal{C}^{\mathbf{S}}$$

3.4 Funções de argumento GF(2^n)

As funções $f : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2^n)$ (SBoxes $n \times n$) têm argumentos que são vistos como elementos do corpo de Galois de ordem n. Como o domínio da função é finito ela pode ser sempre descrita[36] por um polinómio de grau inferior a $2^n - 1$

$$f(x) \;=\; \sum_{i=0}^{2^n-1} a_i\, x^i \;, \qquad a_i \in \mathrm{GF}(2^n) \tag{83}$$

Tendo em atenção que, para todo $x \neq 0$, se verifica

$$x^{2^n-1} = 1 \;,\; x^{2^n-2} = x^{-1} \;,\; \cdots \;,\; x^{2^n-1-i} = x^{-i}$$

então é possível escrever a representação anterior do seguinte modo:

$$f(x) \;=\; \begin{cases} a_0 & , \quad \text{se} \quad x = 0 \\ b_0 + \sum_{i=1}^{N} a_i\, x^i + b_i\, x^{-i} & , \quad \text{se} \quad x \neq 0 \end{cases} \tag{84}$$

[36] Qualquer função $f : \mathbb{F}_q \longrightarrow \mathbb{F}_q$, que passa por N pontos, pode ser aproximada por um polinómio de Lagrange de grau $N - 1$; o número total de argumentos de f é q e, por isso, existem, quanto muito, q pontos que a determinam; logo qualquer aproximação (incluindo a própria função) tem grau igual ou inferior a $q - 1$.

com

$$N = 2^{n-1} - 1 \ , \quad b_0 = a_0 + a_{(2^n-1)} \ , \quad b_i = a_{(2^n-1-i)} \quad i > 0$$

EXEMPLO 25: Na cifra AES as operações sobre *bytes* são descritas por funções $f : \mathrm{GF}(2^8) \rightarrow \mathrm{GF}(2^8)$. O corpo de Galois é representado numa base polinomial do tipo 0 usando o polinómio característico $\mathbf{c} = y^8+y^4+y^3+y+1$.

A definição do **AES** especifica uma função `ByteSub` que é a composição de duas funções $\mathrm{ByteSub} = \mathrm{Inv} \circ \mathrm{Afim}$. A primeira das quais é a função auto-inversa

$$\mathrm{Inv}(x) \ = \ \begin{cases} 0 & \text{se} \quad x = 0 \\ x^{-1} & \text{se} \quad x \neq 0 \end{cases} \tag{85}$$

A segunda função é uma transformação afim definida não sobre $\mathrm{GF}(2^8)$ mas sim sobre sobre $\mathrm{GF}(2)^8$; tem a forma

$$\mathrm{Afim}(x) \ = \ c \oplus (\omega_0(x), \cdots, \omega_{n-1}(x))$$

$$\text{com} \quad c, \omega_i \in \mathrm{GF}(2)^8$$

(os valores concretos das constantes c, ω_i constam na especificação oficial do AES).

É importante ter uma representação de ambas as funções da mesma forma. Por isso é indispensável ver como converter uma representação linear ou afim genérica em $\mathrm{GF}(2)^n$ numa função de domínio $\mathrm{GF}(2^n)$. É o que faremos em seguida.

Alguns casos especiais de funções de domínio $\mathrm{GF}(2^n)$ merecem referência especial:

1. Os **automorfismos** $f : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2^n)$ (funções injectivas que preservam a estrutura do corpo) fixam necessariamente os elementos de $\mathrm{GF}(2)$.

   Vendo $\mathrm{GF}(2^n)$ como uma extensão de $\mathrm{GF}(2)$, o facto 140 determina que f tem a forma $\sigma^k : x \mapsto x^{2^k}$, para algum $k \in 0..n-1$. Só as potências do morfismo de Frobenius $\sigma : x \mapsto x^2$ são automorfismos neste corpo. É possível escrever os σ^k de uma forma vectorial de modo a permitir o uso de uma álgebra matricial para representar transformações lineares.
   Para tal define-se a função $(\cdot)_\sigma : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2^n)^n$ que transforma um elemento x do corpo de Galois no vector, sobre esse corpo, formado por todos $\sigma^k(x)$ (as imagens de x pelos vários automorfismos).

$$x_\sigma \doteq (x \ , \ \sigma(x) \ , \ \sigma^2(x) \ , \ \cdots \ , \ \sigma^{n-1}(x)) \tag{86}$$

2. As **funções boolenas** $f : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2)$ podem ser representadas pelas formas genéricas em (83) ou (84) tendo em atenção que o contradomínio $\mathrm{GF}(2)$ está imerso em $\mathrm{GF}(2^n)$.

EXEMPLO 26: a função *traço* (ver definição 141 – página 149), é definida pelo polinómio $\mathrm{tr}(x) = x + x^2 + x^4 + \cdots + x^{2^{n-1}}$.

3. As **funções lineares** $f : \mathrm{GF}(2^n) \to \mathcal{K}$, sendo $\mathcal{K}$ uma qualquer extensão de $\mathrm{GF}(2)$, são funções que preservam somas e multiplicações por escalares:

$$f(x+y) = f(x) + f(y) \quad , \quad f(a\,x) = a\,f(x)$$

para todos $x, y \in \mathrm{GF}(2^n)$ e $a \in \mathrm{GF}(2)$.

A função linear f pode sempre ser escrita como um polinómio

$$f(x) \;=\; \sum_{k=0}^{n-1} a_k\,\sigma^k(x) \qquad \text{com} \quad a_k \in \mathcal{K} \tag{87}$$

Representanto por $\mathbf{a}$ o vector $(a_0, a_1, \cdots, a_{n-1})$, o polinómio (87) pode ser escrito como o produto escalar[37] de dois vectores

$$f(x) \;=\; \mathbf{a} \cdot x_\sigma \tag{88}$$

[37] Se $u, v \in X^n$ o produto escalar $u \cdot v$ é $u^t\,v = v^t\,u$, em que $(\cdot)^t$ representa a transposição de matrizes.

EXEMPLO 27: Como caso particular temos construções da forma $\mathrm{tr}(x \cdot y)$; pela definição verifica-se imediatamente que, para todos $x, y \in \mathrm{GF}(2^n)$

$$\mathrm{tr}(x\,y) \;=\; x_\sigma \cdot y_\sigma$$

Voltando à representação em (88), seja $y = f(x) = \mathbf{a} \cdot x_\sigma$. Então y_σ pode ser calculado simplesmente como

$$y_\sigma \;=\; \mathbf{A}\, x_\sigma$$

sendo $\mathbf{A}$ a matriz cujo elemento genérico é

$$\mathbf{A}_{ij} = (a_k)^{2^i} \qquad \text{com} \quad k = j - i \pmod{n-1} \tag{89}$$

4. As **funções bilineares** são funções de dois argumentos

$$f : \mathrm{GF}(2^n) \times \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2^n)$$

que são lineares em cada um dos seus argumentos.
Isto é, para todo $x, y, z \in \mathrm{GF}(2^n)$ e $a \in \mathrm{GF}(2)$

$$f(x, y+z) = f(x,y) + f(x,z) \quad , \quad f(x+z, y) = f(x,y) + f(z,y)$$

$$f(x, a \cdot y) = a \cdot f(x,y) = f(a \cdot x, y)$$

A forma mais geral para estas funções é dada por

$$f(x) \;=\; x_\sigma \cdot (\mathbf{A}\, y_\sigma) \tag{90}$$

sendo $\mathbf{A} \in \mathrm{GF}(2^n)^{n\times n}$ uma matriz com elementos em $\mathrm{GF}(2^n)$.

EXEMPLO 28:

Exemplos de formas bilineares em $\mathrm{GF}(2^n)$, com $n > 3$

$$f(x,y) = x \cdot y \quad , \quad f(x,y) = x^2 \cdot y^4 + x^4 \cdot y^2 + c \cdot x^8 \cdot y^8$$

5. As **funções quadráticas** $g : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2^n)$ têm a forma $g(x) = f(x,x)$ sendo $f(\cdot,\cdot)$ uma função bilinear. Isto é,

$$g(x) \;=\; x_\sigma \cdot \mathbf{A}\, x_\sigma \tag{91}$$

para uma matriz apropriada $\mathbf{A} \in \mathrm{GF}(2^n)^{n\times n}$.

EXEMPLO 29:

A função $g(x) = x^3$ é uma função quadrática já que pode ser escrita na forma $f(x,x)$ sendo $f(x,y) = x\, y^2$ que é, claramente, uma forma bilinear.

Pelo mesmo motivo qualquer monómio x^k é uma função quadrática se se puder escrever $k = 2^i + 2^j \pmod{2^n - 1}$ para i, j apropriados.

6. Uma base $\mathcal{B}$ de $\mathrm{GF}(2^n)$ determina sempre n **projecções**; i.e., funções booleanas $p_\mu : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2)$, uma para cada elemento $\mu \in \mathcal{B}$, de tal forma que qualquer $x \in \mathrm{GF}(2^n)$ pode ser reconstruído a partir dos elementos μ e dos valores $p_\mu(x)$

$$x \;=\; \sum_{\mu \in \mathcal{B}} p_\mu(x)\, \mu \tag{92}$$

Representemos por $(\cdot)^{\tilde{}} : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2)^n$ a função que associa x ao vector das suas projecções. Vectorialmente, (92) é

$$x \;=\; \mathcal{B} \cdot x^{\tilde{}} \tag{93}$$

Notas Algumas propriedades das projecções que derivam directamente de (92)

(i) Como a representação de x em $\mathcal{B}$ é única, as projecções são também únicas.

(ii) A projecção em $\mu \in \mathcal{B}$ de um outro elemento da base $\upsilon \in \mathcal{B}$ tem de ser zero porque, por definição de base, não é possível representar υ como uma soma de outros elementos da mesma base. Por isso

$$p_\mu(\upsilon) \;=\; \delta(\mu + \upsilon) \qquad \forall\, \mu, \upsilon \in \mathcal{B}$$

(iii) São sempre funções lineares que preservam a multiplicação escalar; i.e, verificam

$$p_\mu(0) = 0 \quad , \quad p_\mu(x+y) = p_\mu(x) + p_\mu(y) \quad , \quad p_\mu(a\,x) = a\,p_\mu(x)$$

para todos $x, y \in \mathrm{GF}(2^n)$ e $a \in \mathrm{GF}(2)$.

(iv) Para todo $x \in \mathrm{GF}(2^n)$ existe y tal que $p_\mu(x) \neq p_\mu(y)$; isto é, as projecções são funções sobrejectivas.

Se compararmos as duas últimas propriedades com as da função traço (página 149) vemos que elas coincidem; por isso é natural que existam semelhanças entre as duas noções. De facto é relativamente simples provar o seguinte resultado,

168 FACTO
Para qualquer elemento $\mu \in \mathcal{B}$ de uma base de $\mathrm{GF}(2^n)$ existe um único elemento $\mu' \in \mathrm{GF}(2^n)$, designado por **elemento dual** *de μ, tal que, para todo $x \in \mathrm{GF}(2^n)$,*

$$p_\mu(x) \;=\; \mathrm{tr}(\mu' x) \;=\; (\mu')_\sigma \cdot x_\sigma$$

O conjunto $\mathcal{B}' \doteq \{\, \mu' \mid \mu \in \mathcal{B} \,\}$ de todos os elementos duais forma uma nova base de $\mathrm{GF}(2^n)$ que será designada por **base dual** *de $\mathcal{B}$.*

Sugestão de prova: Construa-se a matriz $n \times n$ de elementos $\mathbf{T}_{\mu\upsilon} \doteq \mathrm{tr}(\mu\,\upsilon)$; as colunas da sua inversa $\mathbf{T}^{-1}$ determinam os elementos da base dual.

□

Ordenando os elementos de $\mathcal{B}$ num vector, e preservando a mesma ordem na base dual $\mathcal{B}'$, a prova do facto

anterior mostra que estes vactores estão relacionados por

$$\mathcal{B}' \;=\; \mathbf{T}^{-1}\,\mathcal{B} \tag{94}$$

em que $\mathbf{T}$ designa a matriz dos traços cruzados: $\mathbf{T}_{\mu\upsilon} = \mathrm{tr}(\mu\,\upsilon)$.

EXEMPLO 30: Consideremos de novo $\mathrm{GF}(2^4)$ com uma base polinomial do tipo 0

$$\mathcal{B} = \left\{1\ ,\ \beta\ ,\ \beta^2\ ,\ \beta^3\right\}$$

Para calcular a matriz dos traços $\mathbf{T}_{ij} \doteq \mathrm{tr}(\beta^i \cdot \beta^j) = \mathrm{tr}(\beta^{i+j})$ basta calcular a lista dos traços das potências de β para expoentes entre 0 e 6.

Usando o polinómio característico $\mathbf{c}(X) = X^4 + X + 1$ facilmente se constrói a tabela

i	0	1	2	3	4	5	6
$\mathrm{tr}(\beta^i)$	0	0	0	1	0	0	1

A matriz $\mathbf{T}$ e a sua inversa $\mathbf{T}^{-1}$ são

$$\mathbf{T} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Portanto a base dual é formada pelos elementos (pela ordem dos respectivos duais em $\mathcal{B}$)

$$\mathcal{B}' = \left\{ 1+\beta^3 \ , \ \beta^2 \ , \ \beta \ , \ 1 \right\}$$

e as projecções respectivas serão

$$p_1(x) = \mathsf{tr}(x) + \mathsf{tr}(x\,\beta^3) \qquad , \qquad p_\beta(x) = \mathsf{tr}(x\,\beta^2)$$
$$p_{\beta 2}(x) = \mathsf{tr}(x\,\beta) \qquad , \qquad p_{\beta 3}(x) = \mathsf{tr}(x)$$

o que permite concluir a identidade

$$x = \mathsf{tr}(x) + \mathsf{tr}(x\,\beta^3) + \mathsf{tr}(x\,\beta^2)\,\beta + \mathsf{tr}(x\,\beta)\,\beta^2 + \mathsf{tr}(x)\,\beta^3$$

□

A noção de elemento dual pode ser extendida a qualquer $x \in \mathrm{GF}(2^n)$. O **elemento dual** de x na base $\mathcal{B}$, representado por x', é elemento de $\mathrm{GF}(2^n)$ que tem exactamente as mesmas componentes de x mas na base dual $\mathcal{B}'$; i.e. para todo $\mu \in \mathcal{B}$

$$p_\mu(x) = \mathrm{tr}(\mu' x) = \mathrm{tr}(\mu x') = p_{\mu'}(x') \tag{95}$$

Tomando como implícita a base $\mathcal{B}$, o operador $(\cdot)^\sim$ associa cada elemento de $\mathrm{GF}(2^n)$ ao vector das suas projecções nessa base. Então verifica-se

$$x'^\sim = \mathbf{T}^{-1} x^\sim \quad , \quad x' = \mathcal{B} \cdot x'^\sim = \mathcal{B}' \cdot x^\sim \tag{96}$$

□

O operador $(\cdot)_\sigma$ pode ser extendido para vectores de elementos $\mathrm{GF}(2^n)$. Dado $A \in \mathrm{GF}(2^n)^n$ a matriz $A_\sigma \in \mathrm{GF}(2^n)^{n\times n}$ tem, por linha de ordem i, o vector $(A_i)_\sigma$; isto é $(A_\sigma)_{ik} = \sigma^k(A_i)$.

Nestas circunstâncias

169 FACTO

Se $\mathcal{B}, \mathcal{B}'$ são um par de bases duais então:

1. $(\mathcal{B}')_\sigma = \mathbf{T}^{-1}\,\mathcal{B}_\sigma$

2. $\tilde{x} = (\mathcal{B}')_\sigma \, x_\sigma \quad e \quad x_\sigma = (\mathcal{B}_\sigma)^t \, \tilde{x}$
3. $(\mathcal{B}_\sigma)^t \, \mathcal{B}_\sigma = \mathbf{T} \quad e \quad (\mathcal{B}_\sigma)^t \, (\mathcal{B}')_\sigma = \mathbf{I}.$
4. $\mathcal{B}_\sigma \, (x')_\sigma = (\mathcal{B}')_\sigma \, x_\sigma.$

Prova O primeiro resultado é consequência da relação $\mathcal{B}' = \mathbf{T}^{-1} \, \mathcal{B}$, da linearidade dos morfismos σ^k e do facto de fixarem todos os elementos da matrix $\mathbf{T}^{-1}$.

O segundo resultado é consequência do anterior e do facto 168. Os últimos resultados são as definições da matriz dos traços cruzados $\mathbf{T}$, da base dual e elemento dual.

□

Todas estas noções são essenciais quando se pretende estudar funções booleanas que requerem "mudança de representação"; isto ocorre quando uma função é formada pela composição de uma sequência de funções que manipulam as palavras de *bits* alternadamente na representação $\mathrm{GF}(2^n)$ e na representação $\mathrm{GF}(2)^n$.

É o caso da SBox do AES (introduzida na exemplo 25) que é determinada pela composição da função $x \mapsto x^{-1}$ em $\mathrm{GF}(2^8)$ seguida de uma transformação afim em $\mathrm{GF}(2)^8$.

O problema essencial é o seguinte:

Dada uma base de representação $\mathcal{B}$ *em* $\mathsf{GF}(2^n)$, *dada uma função* $f^{\tilde{}}$ *de domínio* $\mathsf{GF}(2)^n$, *construir a função* f *de domínio* $\mathsf{GF}(2^n)$ *que verifica* $f^{\tilde{}}(x^{\tilde{}}) = f(x)^{\tilde{}}$ *para todo* x.
Ou, inversamente, dada a função f, *construir* $f^{\tilde{}}$.

Note-se que: $f^{\tilde{}}(x^{\tilde{}})$ denota a função de palavras de bits $f^{\tilde{}}$ aplicada ao vector de bits que representa x (visto como elemento do corpo de Galois); $f(x)^{\tilde{}}$ é o vector de bits que representa o resultado $f(x)$; a relação entre as duas funções diz-nos que estes dois vectores são iguais.

Vamos procurar resolver este problema para algumas formas particulares de funções $f^{\tilde{}} : \mathsf{GF}(2)^n \to \mathsf{GF}(2)^n$ ou $f^{\tilde{}} : \mathsf{GF}(2)^n \to \mathsf{GF}(2)$.

$f^{\tilde{}}(x^{\tilde{}}) = x^{\tilde{}} \oplus c^{\tilde{}}$ com $c \in \mathsf{GF}(2^n)$.

Esta é a função *branqueamento* ("whitening") que ocorre quase sempre nos andares das cifras. A função de domínio $\mathsf{GF}(2^n)$ equivalente é

$$f(x) \;=\; x + c$$

$f^{\tilde{}}(x^{\tilde{}}) = c^{\tilde{}}(x^{\tilde{}}) = \mathsf{Tr}(c^{\tilde{}} * x^{\tilde{}})$ com $c \in \mathsf{GF}(2^n)$

Forma genérica da função booleana linear; a função de domínio $\mathsf{GF}(2^n)$ equivalente é

$$f(x) \;=\; \mathsf{tr}(c'\,x) \;=\; (c')_\sigma \cdot x_\sigma$$

Prova: Simples utilização das identidades no facto 169

$$f^{\tilde{}}(x^{\tilde{}}) = c^{\tilde{}} \cdot x^{\tilde{}} =$$

$$(\mathcal{B}_\sigma\,(c')_\sigma) \cdot (\mathcal{B}')_\sigma\, x_\sigma = (((\mathcal{B}')_\sigma)^t \mathcal{B}_\sigma\,(c')_\sigma) \cdot x_\sigma = (c')_\sigma \cdot x_\sigma$$

$f^{\tilde{}}(x^{\tilde{}}) = \mathbf{H}\, x^{\tilde{}}$ com $\mathbf{H} \in \mathrm{GF}(2)^{n\times n}$.

Esta é a forma genérica da SBox linear onde cada *bit* à saída é uma combinação linear dos *bits* de entrada.
A função de domínio $\mathrm{GF}(2^n)$ equivalente é o polinómio

$$f(x) = \quad \mathbf{h} \cdot x_\sigma \quad = \quad \sum_{k=0}^{n-1} \mathbf{h}_k\, x^{2^k} \tag{97}$$

em que $\mathbf{h} \in \mathrm{GF}(2^n)^n$ é o vector

$$\mathbf{h} = \quad (\mathcal{B}')_\sigma^t\, \mathbf{H}^t\, \mathcal{B}$$

Prova:

$$f(x) = \mathcal{B} \cdot f^{\sim}(x^{\sim}) =$$

$$\mathcal{B} \cdot (\mathbf{H}\,(\mathcal{B}')_{\sigma}\,x_{\sigma}) = ((\mathcal{B}')^{t}_{\sigma} \cdot \mathbf{H}^{t}\,\mathcal{B}) \cdot x_{\sigma}$$

É importante ter-se em conta que (97), apesar da forma aparentemente complexa, é uma função linear. porque tem a forma prevista em (87).

Pode-se interpretar $\mathbf{H}^{t}\,\mathcal{B}$ como o vector determinado pelas colunas de $\mathbf{H}$ vendo cada coluna como os coeficientes de um elemento de GF(2^{n}) na base $\mathcal{B}$.

Este vector (e, consequentemente, a matriz $\mathbf{H}$) pode ser recuperado de $\mathbf{h}$ por

$$\mathbf{H}^{t}\,\mathcal{B} \;=\; \mathcal{B}_{\sigma}\,\mathbf{h}$$

Isto permite fazer a conversão em sentido inverso: dada a função linear de domínio GF(2^{n}), obter a matriz que determina a função equivalente de domínio GF$(2)^{n}$.

A relação entre $\mathbf{H}$ e $\mathbf{h}$ pode ainde ser vista de outr forma; note-se que se podia escrever

$$\mathbf{h} \cdot x_{\sigma} = \; (\mathbf{H}^{t}\,\mathcal{B}) \cdot (\mathcal{B}')_{\sigma}\,x_{\sigma} = (\mathbf{H}^{t}\,\mathcal{B}) \cdot x^{\sim}$$

Finalmente pode-se ver

$$\mathbf{h} \;=\; (\mathbf{H}\,\mathcal{B}')^{t}_{\sigma}\,\mathcal{B}$$

$\mathbf{H}\,\mathcal{B}'$ representa ver as linhas da matriz como vectores representados na base $\mathcal{B}'$; isto é, se ω_{i} representar o elemento de GF(2^{n}) representado pela linha de ordem i da matriz **H**, então $\mathbf{H}\,\mathcal{B}' = (\omega'_{0}, \cdots, \omega'_{n-1})$ é o vector dos elementos duais.

EXEMPLO 31: Regressando à cifra AES e ao exemplo 25, a transformação afim $g\tilde{}(y\tilde{}) = \mathbf{b} \oplus \mathbf{H}\, y\tilde{}$ é definida, na especificação da cifra, por

$$\mathbf{b} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \qquad \mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

ou, representando cada *byte* em base hexadecimal e a marix como um vector de colunas,

$$\mathbf{b} = 63 \qquad \mathbf{H} = (\texttt{8F}\ ,\ \texttt{C7}\ ,\ \texttt{E3}\ ,\ \texttt{F1}\ ,\ \texttt{F8}\ ,\ \texttt{7C}\ ,\ \texttt{3E}\ ,\ \texttt{1F})$$

A especificação do AES define o polinómio caractarístico

$$\mathbf{c}[X] = 1 + X + X^3 + X^4 + X^8$$

Para uma base polinomial do tipo 0 gerada por este polinómio, e usando a mesma metodologia de representação,

temos a matriz dos traços cruzados, a sua inversa e $\mathcal{B}_\sigma$

$$\mathbf{T} = (05\ ,\ 0A\ ,\ 15\ ,\ 2B\ ,\ 57\ ,\ AE\ ,\ 5C\ ,\ B9)$$

$$\mathbf{T}^{-1} = (94\ ,\ 0D\ ,\ 1A\ ,\ A0\ ,\ 65\ ,\ CA\ ,\ 25\ ,\ 2A)$$

$$\mathcal{B}_\sigma = \begin{bmatrix} 01 & 01 & 01 & 01 & 01 & 01 & 01 & 01 \\ 02 & 04 & 10 & 1B & 5E & E4 & 4D & FA \\ 04 & 10 & 1B & 5E & E4 & 4D & FA & 02 \\ 08 & 40 & AB & B3 & E8 & 1D & 4A & EF \\ 10 & 1B & 5E & E4 & 4D & FA & 02 & 04 \\ 20 & 6C & 97 & 94 & 91 & 80 & 9A & C5 \\ 40 & AB & B3 & E8 & 1D & 4A & EF & 08 \\ 80 & 9A & C5 & 20 & 6C & 97 & 94 & 91 \end{bmatrix}$$

Os elementos de $\mathcal{B}_\sigma$ são polinómios; nesta representação são apresentados os seus coeficientes como vectores de bits, começando no grau mais elevado. Por exemplo $E4 = 11100100$ denota o polinómio $X^7 + X^6 + X^5 + X^2$.

Com estas três matrizes é possível computar todos os elementos necessários; por exemplo,

$$(\mathcal{B}')_\sigma = \mathbf{T}^{-1}\,\mathcal{B}_\sigma$$

que é a componente essencial para calcular $\mathbf{h}$ a partir de $\mathbf{H}$.

Feitas as contas conclui-se

$$\mathbf{h} = (\texttt{05}\ ,\ \texttt{09}\ ,\ \texttt{F9}\ ,\ \texttt{25}\ ,\ \texttt{F4}\ ,\ \texttt{01}\ ,\ \texttt{B5}\ ,\ \texttt{8F})$$

Conclui-se portanto que a transformação afim do AES é

$$\begin{aligned} g(y) = \quad & \texttt{63} + \texttt{05} \cdot y + \texttt{09} \cdot y^2 + \texttt{F9} \cdot y^4 + \texttt{25} \cdot y^8 + \\ & + \texttt{F4} \cdot y^{16} + \texttt{01} \cdot y^{32} + \texttt{B5} \cdot y^{64} + \texttt{8F} \cdot y^{128} \end{aligned}$$

A SBox do AES resulta da composição dessa função ao resultado da transformação

$$x \mapsto x^{254}$$

que mapeia 0 em si próprio e qualquer $x \neq 0$ em x^{-1}.

A transformação final abtém-se então substituindo, em $g(y)$, a variável y por x^{254} e reduzindo os expoentes módulo 255 (uma vez que $x^{255} = 1$ se $x \neq 0$).

Por exemplo

$$y^{128} \quad \mapsto \quad (x^{254})^{128} \quad \mapsto \quad x^{(254*128 \mod 255)} \quad \mapsto \quad x^{127}$$

Genericamente y^k reduz-se a x^{255-k}. O resultado final é

$$f(x) = \mathtt{63} + \mathtt{05} \cdot x^{254} + \mathtt{09} \cdot x^{253} + \mathtt{F9} \cdot x^{251} + \mathtt{25} \cdot x^{247} + \\ + \mathtt{F4} \cdot x^{239} + \mathtt{01} \cdot x^{223} + \mathtt{B5} \cdot x^{191} + \mathtt{8F} \cdot x^{127} \tag{98}$$

□

Finalmente vamos examinar a representação das **funções bilineares**

$$f(x, y) = x_\sigma \cdot \mathbf{A}\, y_\sigma \tag{99}$$

ou equivalentemente, com $\boldsymbol{\alpha} \doteq \mathcal{B}_\sigma\, \mathbf{A}\, \mathcal{B}_\sigma^t$

$$f(x, y) = x^{\tilde{}} \cdot \boldsymbol{\alpha}\, y^{\tilde{}} \tag{100}$$

Usando as identidades $x_\sigma = \mathcal{B}_\sigma^t\, x^{\tilde{}}$ e $\boldsymbol{\alpha} = \mathcal{B}_\sigma\, \mathbf{A}\, \mathcal{B}_\sigma^t$ tem-se

$$f(x, y) = (\mathcal{B}_\sigma^t\, x^{\tilde{}}) \cdot \mathbf{A}\, \mathcal{B}_\sigma^t\, y^{\tilde{}} = x^{\tilde{}} \cdot \mathcal{B}_\sigma\, \mathbf{A}\, \mathcal{B}_\sigma^t\, y^{\tilde{}} = x^{\tilde{}} \cdot \boldsymbol{\alpha}\, y^{\tilde{}}$$

Seja $\boldsymbol{\alpha}_k$ a matriz $\mathrm{GF}(2)^{n\times n}$ que selecciona a componente k de cada um dos elementos de $\boldsymbol{\alpha}$; seja z_k a componente correspondente de $f(x,y)$; então

$$z_k = x^{\tilde{}} \cdot \boldsymbol{\alpha}_k \, y^{\tilde{}}$$

Desta forma é possível calcular as componentes individuais de $f(x,y)$; expandindo obtém-se uma função booleana com monómios da forma $x_i \, y_j$.

$$z_k = \sum_{ij} \boldsymbol{\alpha}_k^{ij} \, x_i \, y_j$$

EXEMPLO 32: Um exemplo de tal função, em $\mathrm{GF}(2)^4$ seria definida pelo sistema de equações

$$\begin{bmatrix} z_0 = & x_0\,y_0 + x_1\,y_0 + x_1\,y_1 \\ z_1 = & x_1\,y_0 + x_1\,y_2 + x_2\,y_0 \\ z_2 = & x_3\,y_3 \\ z_3 = & x_2\,y_0 + x_0\,y_2 \end{bmatrix}$$

As matrizes $\boldsymbol{\alpha}_k$ são

$$\boldsymbol{\alpha}_0 = \begin{bmatrix} 1000 \\ 1100 \\ 0000 \\ 0000 \end{bmatrix} \qquad \boldsymbol{\alpha}_1 = \begin{bmatrix} 0000 \\ 1010 \\ 1000 \\ 0000 \end{bmatrix} \qquad \boldsymbol{\alpha}_2 = \begin{bmatrix} 0000 \\ 0000 \\ 0000 \\ 0001 \end{bmatrix} \qquad \boldsymbol{\alpha}_3 = \begin{bmatrix} 0010 \\ 0000 \\ 1000 \\ 0000 \end{bmatrix}$$

A matriz $\boldsymbol{\alpha}$ congrega estas componentes individuais em vectores de *bits*

$$\boldsymbol{\alpha} = \begin{bmatrix} 1000 & 0000 & 0001 & 0000 \\ 1100 & 1000 & 0100 & 0000 \\ 0101 & 0000 & 0000 & 0000 \\ 0000 & 0000 & 0000 & 0010 \end{bmatrix}$$

Conhecida a matriz $\boldsymbol{\alpha}$, é possível recuperar a matriz $\mathbf{A}$,

$$\boldsymbol{\alpha} = \mathcal{B}_\sigma \, \mathbf{A} \, \mathcal{B}_\sigma^t \quad \Longrightarrow \quad \mathbf{A} = (\mathcal{B}')_\sigma^t \, \boldsymbol{\alpha} \, (\mathcal{B}')_\sigma \quad \text{com} \quad (\mathcal{B}')_\sigma = \mathbf{T}^{-1} \, \mathcal{B}_\sigma$$

Tomemos o polinómio característico $\mathbf{c}[X] = X^4 + X + 1$ e a base polinomial de tipo 0 apresentada no exemplo 30 (página 235).

A base dual calculada nesse exemplo foi

$$\mathcal{B}' = \left\{ 1 + \beta^3 \ , \ \beta^2 \ , \ \beta \ , \ 1 \right\}$$

As matrizes $(\mathcal{B}')_\sigma$ e $\mathbf{A}$ que daí resultam são

$$(\mathcal{B}')_\sigma = \begin{bmatrix} 9 & 4 & 2 & 1 \\ D & 3 & 4 & 1 \\ E & 5 & 3 & 1 \\ B & 2 & 5 & 1 \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} A & 5 & B & A \\ 3 & D & D & B \\ 2 & D & 7 & 1 \\ 5 & 5 & 7 & D \end{bmatrix}$$

As **funções quadráticas** têm uma representação que deriva directamente da representação das funções bilineares: se for $z = g(x) = x_\sigma \cdot \mathbf{A} x_\sigma$ então teremos

$$z \;=\; \tilde{x} \cdot \boldsymbol{\alpha}\, \tilde{x}$$

com $\boldsymbol{\alpha} = \mathcal{B}_\sigma\, \mathbf{A}\, \mathcal{B}_\sigma^t$. Deste modo, a componente z_k do resultado é

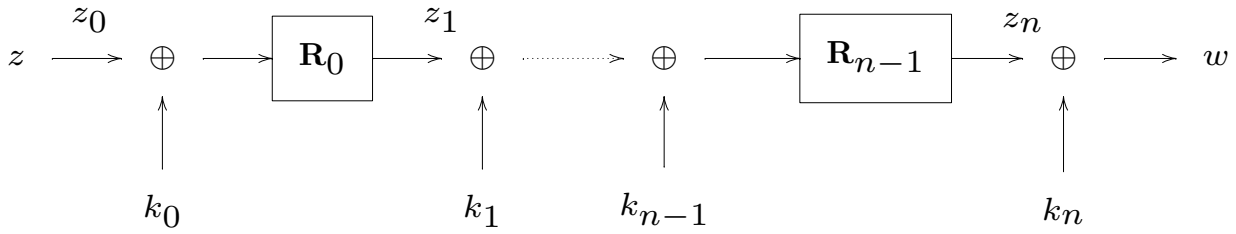
$$z_k \;=\; \sum_{ij} \boldsymbol{\alpha}_k^{ij}\, x_i\, x_j \tag{101}$$

em que $\boldsymbol{\alpha}_k^{ij}$ denota a componente de ordem k do elemento de índices i, j da matriz $\boldsymbol{\alpha}$.

O valor boolenao $\boldsymbol{\alpha}_k^{ij}$ determinam se o monómio $x_i\, x_j$ ocorre ou não na função boolena que calcula z_k. Assim, em termos de representações em $\mathrm{GF}(2)^n$, as formas quadráticas produzem (como seria de esperar) funções boolenas de grau 2; todos os monómios são da forma $x_i\, x_j$.

3.5 Criptoanálise Algébrica

As principais cifras simétricas[38] modernas têm a seguinte estrutura



(102)

A função **cifrar** ($w = f(k, z)$) processa-se de acordo com

1. As componentes essenciais são n SBoxes $\mathbf{R}_i$ invertíveis (os "**rounds**") e um **programador de chaves** (não representado na figura) que expande a **chave de controlo** k em $n+1$ **chaves de "round"** $k_0, k_1, .., k_n$.
2. Existe um **estado** z_i , inicializado com a entrada z , e que é recorrentemente alterado por duas transformações: uma soma $\oplus$ com a chave k_i e a aplicação de $\mathbf{R}_i$. Após o "round" final, a saída w obtém-se somando uma última chave k_n ao estado.
3. A relação de recorrência, fazendo variar $i \in 0..n-1$, é

$$z_0 = z \quad , \quad z_{i+1} = \mathbf{R}_i(z_i \oplus k_i) \qquad w = z_n \oplus k_n \tag{103}$$

[38] Cifras que usam a mesma chave para cifrar e decifrar.

A operação **decifrar** ($z = f^{-1}(k, w)$) tem a mesma estrutura com as alterações:

1. A novas chaves de "round" k'_i são as iniciais mas são apresentadas pela ordem inversa; isto é, $k'_i = k_{n-i}$
2. As SBoxs $\mathbf{R}_i$ são substitídas elas suas inversas algébricas e são apresentadas pela ordem inversa; isto é, $\mathbf{R}'_i = \mathbf{R}^{-1}_{n-i-1}$
3. O estado, representado por w_i , calcula-se fazendo variar $i \in 0..n-1$

$$w_0 = w \qquad w_{i+1} = \mathbf{R}'_i(w_i \oplus k'_i) \qquad z = w_n \oplus k'_n$$

Para ver que realmente estas duas funções são a inversa uma da outra basta notar que se preserva o invariante

$$w_i \oplus k'_i = z_{n-i} \qquad \text{ou} \qquad z_i \oplus k_i = w_{n-i}$$

Presupõe-se que uma mesma chave k é usada para cifrar um grande número de mensagens[39]. Presupõe-se também que são conhecidos alguns (poucos) pares mensagem+criptograma $\langle z_i, w_i \rangle$ gerados com essa chave.

Um ataque é uma algoritmo tratável que, a partir desta informação, descobre a chave k ou, equivalentemente, um algoritmo tratável que, a partir de um w arbitrário (distincto dos w_i conhecidos), descobre o elemento z

[39] Se a chave k fosse usada apenas uma vez então a cifra mais segura é simplesmente $w = z \oplus k$; esta é a chamada **segurança perfeita** de Shanon.

que cifrado com k reconstrói w. Numa cifra ideal a sua entropia está limitada pelo tamanho da chave em *bits*. Qualquer quebra em relação a este valor máximo é um ataque.

A segurança da cifra depende crucialmente do "design" dos "rounds" $\mathbf{R}_i$ e, normalmente, tem-se em vista dois objectivos principais:

- Não deve existir propagação de diferenças e, por isso, cada $\mathbf{R}_i$ deve manifestar um elevado grau de não-linearidade[40].

  Mais precisamente deve existir uma boa **mistura** entre a chave e a entrada: não deve ser possível "separar", à saída, os efeios individuais de cada um destes items.
- Não devem existir correlações entre visões particulares (paridades) da entrada e da saída. A existência de tais correlações implicaria a existência de relações priveligiadas entre alguns *bits* da entrada com alguns *bits* da saída, o que equivale a uma quebra na entropia da cifra.

  Deve existir uma boa **difusão** da influência de qualquer *bit* da entrada por toda a saída.

Idealmente os objectivos de "boa mistura" e "boa difusão" deveriam ser assegurados por um "design" único das SBoxes. No entanto, se atendermos que é necessário assegurar também boas propriedades computacionais numa função não-linear invertivel, este "design único" torna-se muito difícil.

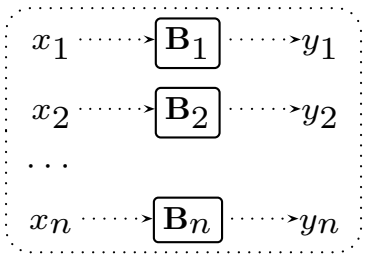
[40] Como cada "round" tem de ser uma função algebricamente invertível (para ser possível decifrar) se fosse linear as técnicas usuais de álgebra linear permitiriam um ataque trivial.

Por isso é usual decompor as SBox em componentes cada uma das quais com um objectivo próprio. Por exemplo é usual escolher uma componente linear com propriedades óptimas em termos de difusão mas muito má (por ser linear) em termos de mistura. Outro tipo de componente são as "bricklayer functions" (que veremos em seguida) que têm boas propriedades em termos de mistura e eficiência computacional mas que apresentam elevadas correlações e, por isso, são más em termos de difusão.

Funções "bricklayer"

A entrada x e a saída y são vectores de $n \times s$ *bits* agrupados em n blocos de s *bits*.

Cada bloco é transformado independentemente dos restantes por uma $s \times s$-SBox $\mathbf{B}_i$. A SBox global $\mathbf{B}$ (de tamanho $(n\,s) \times (n\,s)$) obtém-se fraccionando a entrada x em blocos x_i, aplicando a SBox $\mathbf{B}_i$ ao bloco x_i e agregando os resultados parciais y_i num único vector y.



$$y = \mathbf{B}(x) \qquad y, x \in (\mathbb{B}^s)^n$$
$$y = (y_1, \ldots, y_n) \qquad x = (x_1, \ldots, x_n)$$
$$y_i = \mathbf{B}_i(x_i) \qquad x_i, y_i \in \mathbb{B}^s$$

Se todas as SBoxes $\mathbf{B}_i$ forem invertíveis, também $\mathbf{B}$ é invertível. Adicionalmente $\mathbf{B}^{-1}$ é também uma função "bricklayer" determinada pelas n inversas $\mathbf{B}_i^{-1}$.

A vantagem destas funções reside na sua eficiência computacional; isto deriva do facto de lidar com SBox de uma dimensão s que é muito menor do que a simensão $(n\, s)$ exigida para a SBox global.

Para além de ausências de diferenças e correlações (ligadas aos objectivos de "boa mistura" e "boa difusão"), surge um outro tipo de objectivo que deriva da existência de uma outra forma de ataque.

Considere-se, de novo, as relações em (103) que traduzem as relações essenciais entre os vários items de informação que caracterizam a cifra. Delas resulta um sistema de equações com incógnitas z_i e k_i conhecida a entrada z e o criptograma w.

$$\begin{cases} z_0 & = z \\ z_n \oplus k_n & = w \\ z_{i+1} \oplus \mathbf{R}_i(z_i \oplus k_i) & = 0 \qquad i \in 0..n-1 \end{cases} \tag{104}$$

Uma tentativa de resolver estas equações esbarra com a forma complicada das SBoxes $\mathbf{R}_i$. No entanto é geralmente possível inserir estas equações numa estrutura algébrica adequada (num corpo finito, específicamente) de tal modo que as equações possam ser escritas

$$\begin{cases} z_0 & = z \\ z_n + k_n & = w \\ \mathbf{F}_i(z_{i+1}, z_i + k_i) & = 0 \qquad i \in 0..n-1 \end{cases} \tag{105}$$

em que as funções $\mathbf{F}_i(\cdot,\cdot)$ têm uma estrutura algebricamente muito mais símples do que os $\mathbf{R}_i(\cdot)$.

EXEMPLO 33: O exemplo paradigmático é SBox AES determinada por $y = x^{254}$ ou $y = x^{-1}$ (quando $x \neq 0$) que, explicitamente, é uma função não-linear bastante complexa mas que, implicitamente, se escreve como uma forma bilinear

$$x \cdot y = 1 \quad , \; x \neq 0$$

A forma indicada em (99) e (100) (página 245) para as funções bileneares permite-nos ver esta forma gerada pela matrizes

$$\mathbf{A} = \begin{bmatrix} 10000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \\ 00000000 \end{bmatrix} \qquad \boldsymbol{\alpha} = \begin{bmatrix} 01 & 02 & 04 & 08 & 10 & 20 & 40 & 80 \\ 02 & 04 & 08 & 10 & 20 & 40 & 80 & 1B \\ 04 & 08 & 10 & 20 & 40 & 80 & 1B & 36 \\ 08 & 10 & 20 & 40 & 80 & 1B & 36 & 6C \\ 10 & 20 & 40 & 80 & 1B & 36 & 6C & D8 \\ 20 & 40 & 80 & 1B & 36 & 6C & D8 & AB \\ 40 & 80 & 1B & 36 & 6C & D8 & AB & 4D \\ 80 & 1B & 36 & 6C & D8 & AB & 4D & 9A \end{bmatrix}$$

Uma vez mais, o elementos genérico a em ambas as matrizes é descrito pela palavra de bits $a^{\tilde{}}$ em notação hexadecimal.

Recordemos, nesta representação da forma bilinear $x \cdot y$, a equação $1 = x \cdot y$ se descreve como

$$1 \;=\; \sum_{i,j} \boldsymbol{\alpha}_{ij} \cdot \; x_i \, y_j$$

É possível olhar para esta representação de uma outra forma: considera-mos, por momentos, os pares $x_i \, y_j$ como uma única incógnita e a matriz $\boldsymbol{\alpha}$ como um vector de coeficientes. Vamos chamar X ao vector destas "incógnitas duplas" e continuamos a representar por $\boldsymbol{\alpha}$ a forma linearizada da matriz.

Com $64 = 8 \times 8$ incógnitas booleanas a equação é escrita

$$1 \;=\; \boldsymbol{\alpha} \cdot X \tag{106}$$

Esta equação base dá origem e outras; nomeadamente

$$x = x^2 \, y, \quad x^2 = x^4 \, y^2, \quad x^4 = x^8 \, y^4, \quad \cdots x^{64} = x^{128} \, y^{64}$$

e a versão dual para y

$$y = x \, y^2, \quad y^2 = x^2 \, y^4, \quad \cdots \quad y^{64} = x^{64} \, y^{128}$$

Cada uma destas duas sequências é gerada a partir da equação base ($x = x^2 \cdot y$ ou $y = x \cdot y^2$) por aplicação sucessiva da transformação linear $\sigma : z \mapsto z^2$ a ambos os lados da equação; as equações resultantes são, portanto, linearmente dependentes.

De facto σ gera transformações lineares nos vectores $x\tilde{}$ e $y\tilde{}$ que permitem ir transformando uma equação na seguinte.

Portanto temos três equações em GF(2^8) (das quais resultam 3×8 equaçoes em GF(2)) que podem ou não ser linearmente independentes e que derivam de

$$x \cdot y = 1 \quad (x \neq 0) \qquad x^2\, y + x = 0 \qquad x\, y^2 + y = 0$$

Existem ainda outras formas bilineares; por exemplo, multiplicando ambos os lados da 2ª equação por x^2 e ambos os lados da 3ª equação por y^2 constrói-se

$$x^4\, y + x^3 = 0 \qquad x\, y^4 + y^3 = 0$$

As formas bilineares em todas estas 5 equações são por matrizes¡ $\mathbf{A}$ apropriadas que vão dar origem a matrizes $\boldsymbol{\alpha}$.

Os restnates constituintes dos lados esquerdos das equações (para além das formas bilineares) são monímios x , y , que são formas lineares, ou os monómios x^3 , y^3 , que são formas quadráticas.

As formas bilineares têm exactamente a mesma forma que (106) e dão origem a somas de monómios do tipo $x_i\, y_j$ calculadas por $x\tilde{} \cdot \boldsymbol{\alpha}\, y\tilde{} = \sum_{ij} \boldsymbol{\alpha}_{ij}\, x_i\, y_j$.

Para a forma $x^2\, y$ tem-se a matriz $\mathbf{A}'$ com $\mathbf{A}'_{10} = 1$ e $\mathbf{A}'_{ij} = 0$ para os restantes índices.

A forma $x\,y^2$ é determinada por uma matriz $(\mathbf{A}')^t$ que é transposta da anterior.

A forma $x^4\,y$ é definida pela matriz $\mathbf{A}''$ com $\mathbf{A}''_{20} = 1$ e $\mathbf{A}'_{ij} = 0$ para os restantes índices. A forma $x\,y^4$ é definida pela transposta desta matriz.

As matrizes $\boldsymbol{\alpha}$ correspondentes serão

$$\boldsymbol{\alpha}' = \begin{bmatrix} 01 & 02 & 04 & 08 & 10 & 20 & 40 & 80 \\ 04 & 08 & 10 & 20 & 40 & 80 & 1B & 36 \\ 10 & 20 & 40 & 80 & 1B & 36 & 6C & D8 \\ 40 & 80 & 1B & 36 & 6C & D8 & AB & 4D \\ 1B & 36 & 6C & D8 & AB & 4D & 9A & 2F \\ 6C & D8 & AB & 4D & 9A & 2F & 5E & BC \\ AB & 4D & 9A & 2F & 5E & BC & 63 & C6 \\ 9A & 2F & 5E & BC & 63 & C6 & 97 & 35 \end{bmatrix}$$

$$\boldsymbol{\alpha}'' = \begin{bmatrix} 01 & 02 & 04 & 08 & 10 & 20 & 40 & 80 \\ 10 & 20 & 40 & 80 & 1B & 36 & 6C & D8 \\ 1B & 36 & 6C & D8 & AB & 4D & 9A & 2F \\ AB & 4D & 9A & 2F & 5E & BC & 63 & C6 \\ 5E & BC & 63 & C6 & 97 & 35 & 6A & D4 \\ 97 & 35 & 6A & D4 & B3 & 7D & FA & EF \\ B3 & 7D & FA & EF & C5 & 91 & 39 & 72 \\ C5 & 91 & 39 & 72 & E4 & D3 & BD & 61 \end{bmatrix}$$

Juntanto todas estes vectors $\boldsymbol{\alpha}$ como linhas de uma matriz constrói-se uma nova matriz com $64 = 8 \times 8$ colunas

e $5 \times 8 = 40$ linhas. Daqui resulta um sistema de equações "aparentemente linear" nas incógnitas X

$$\begin{cases} 1 & = \boldsymbol{\alpha} \cdot X \\ x & = \boldsymbol{\alpha}' \cdot X \\ y & = (\boldsymbol{\alpha}')^t \cdot X \\ x^3 & = \boldsymbol{\alpha}'' \cdot X \\ y^3 & = (\boldsymbol{\alpha}'')^t \cdot X \end{cases} \quad \Longrightarrow \quad \begin{bmatrix} 1 \\ x \\ y \\ x^3 \\ y^3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\alpha}' \\ (\boldsymbol{\alpha}')^t \\ \boldsymbol{\alpha}'' \\ (\boldsymbol{\alpha}'')^t \end{bmatrix} X$$

Não se trata realmente de um sistema de equações lineares porque os $x\tilde{}, y\tilde{}$ aparecem por si no lado esquerdo das equações ao mesmo tempo que aparecem nas incógnitas X .

A questão essencial está no número de equações que são linearmente independentes e sobre a forma como estes sistemas podem ser resolvidos.

No exemplo anterior resultaram 5 formas bilineares

$$x \cdot y \qquad x^2 \cdot y \qquad x^4 \cdot y \qquad x \cdot y^2 \qquad x \cdot y^4$$

para as quais foi possível construir equações lineares no binómios $x_i y_j$. Levantam-se imediatamente duas questões:

1. As equações resultantes são linearmente independentes?

2. Será possível acrescentar outras formas (por exemplo, $x^8 \cdot y$) que gerem equações linearmente independentes das existentes?

4.Criptografia Simétrica

Neste capítulo procuraremos dar uma visão introdutória da família de técnicas criptográficas cujo uso requer o conhecimento, partilhado por todos os legítimos intervenientes, de uma única chave. Esta família de técnicas designa-se, genericamente, por **criptografia simétrica**.

Dada a limitação no número de chaves, a gama de técnicas que é possível definir é bastante limitada; essencialmente são cifras, funções de "hash" e combinações destes dois tipos base. Em contrapartida estas técnicas fornecem:

. **Elevada eficiência computacional**
Um parâmetro fundamental para aferição de uma técnica criptográfica é o seu **"throughput"**; isto é, a quantidade de informação que consegue processar numa unidade de tempo. As técnicas simétricas são as que possuem o maior "throughput" tendo capacidade para processar, em tempo real, sinais de áudio e vídeo. Nomeadamente permitem implementações em "hardware" dedicado, o que é factor essencial para aumentar o "throughput". contribui fortemente para aumentar a eficiência computacional.

. **Elevada eficiência de segurança**
A incerteza quando à viabilidade de um ataque com sucesso, expresso em bits, mede o *grau de segurança* de uma

técnica criptográfica. Nas técnicas simétricas a incerteza está sempre limitada pelo comprimento da chave[41] mas pode, no entanto, ser menor; a diferença entre o tamanho da chave e o grau de segurança da técnica, mede a *eficiência* de segurança da técnica.

As técnicas simétricas têm uma elevada eficiência de segurança já que (pelo menos, nas boa técnicas) o grau de segurança está muito próximo do tamanho da chave; por exemplo, mesma na sua versão mais limitada, o AES usa chaves de 128 bits e tem um grau de segurança muito próximo desse valor. Em contraste uma cifra assimétrica como o RSA precisa de uma chave de 1024 bits para ter um grau de segurança da ordem dos 80 bits.

. **Elevada flexibilidade e robustez**

É frequente que vários contextos de processamento de informação imponham restrições e condicionalismos de segurança específicos a uma mesma técnica criptográfica. As técnicas simétricas têm a capacidade de se adaptar a estas condições através de modos de funcionamento específicos. Também têm a flexibilidade de se metamorfosear adaptando-se às necessidades do contexto; por exemplo, as mesmas primitivas criptográficas básicas podem ser usadas para cifras de blocos, cifras sequenciais e funções de "hash".

Outro aspecto essencial é a sua robustez a ataques; quando existe a hipótese de um ataque é possível, quase sempre, efectuar ligeiras modificações à técnica básica de modo a o tornar inviável. A evolução do DES para o 3DES, que estendeu efectivamente o tempo de vida útil do DES para 30 anos, foi uma resposta simples ao

[41] Existe sempre um ataque trivial, o chamado **ataque por força bruta**, que consiste em percorrer todo o espaço de chaves até que uma delas satisfaça uma determinada condição; por exemplo, verificar se o criptograma obtido com a chave teste coincide com um determinado valor observado.

aparecimento das técnicas de criptoanálise; do mesmo modo, quando foi detectado um ataque ao SHA logo no início da sua existência, uma ligeira alteração conduziu ao SHA-1 que é, já há 15 anos, a função de "hash" mais popular.

Por estas razões as técnicas simétricas são os alicerces de todo o edifício de segurança num sistema de informação. Especificações de segurança complexas e subtis exigem, normalmente, técnicas assimétricas; no entanto, dado que as técnicas assimétricas, devido à sua elevada complexidade computacional, conseguem processar apenas pequenas quantidades de informação (chaves, "hashs", etc.), o tratamento de elevadas quantidades de dados exige sempre uma combinação de uma técnica assimétrica com uma técnica simétrica capaz de processar o volume de dados.

4.1 Nomenclatura de Cifras

Permutação

Uma permutação (também designada por **substituição simples**) é uma função bijectiva de domínio finito; isto é, uma função $f : \mathbb{B}^n \longrightarrow \mathbb{B}^n$ que seja injectiva.

As permutações são a construção invertível mais simples e estão na base das cifras ditas **determinísticas**: a permutação, aplicada duas vezes, ao mesmo "input" produz sempre o mesmo "output".

Transformação holomórfica

Uma função $h : \mathbb{N} \longrightarrow \mathbb{N}$ tal que a família de conjuntos $\left\{h^{-1}(n)\right\}_{n\in\mathbb{N}}$ é uma cobertura disjunta de $\mathbb{N}$: isto é, os conjuntos são disjuntos dois a dois e a sua união cobre todo $\mathbb{N}$.

A transformação é usada para construção das cifras homomóficas: um algoritmo probabilístico que, sob "input" x, produz um $y \in h^{-1}(x)$. A função h é usada para decifrar o criptograma y.

Cifras por blocos

São primitivas criptográficas representáveis por funções boolenas $f : \mathbb{B}^t \times \mathbb{B}^n \longrightarrow \mathbb{B}^n$ tais que, para cada $k \in \mathbb{B}^t$, a função $f(k, \cdot) : \mathbb{B}^n \longrightarrow \mathbb{B}^n$ é uma permutação. Os inteiros n e t são os comprimentos, respectivamente, do **bloco** e da **chave**.

Cifras iteradas

São cifras por blocos em que a função f é construída por composição de um número finito p de cifras por blocos $r_1, r_2, ..., r_p : \mathbb{B}^t \times \mathbb{B}^n \longrightarrow \mathbb{B}^n$, designadas por **"rounds"**, da seguinte forma:

1. Existe uma função $g : \mathbb{B}^t \longrightarrow \left(\mathbb{B}^t\right)^p$ chamada **programador de chaves** que, a cada chave $k \in \mathbb{B}^t$, associa um vector com p chaves $g(k) = (k_1, k_2, ..., k_p)$ em $\mathbb{B}^t$.
2. Para uma determinada **chave** k e **texto** x, o **criptograma** correspondente $y = f(x, k)$ é calculado através da sequência
$$y_0 = x \quad , \quad y_i = r_i(k_i, y_{i-1}) \quad i = 1, ..., t \quad , \quad y = y_t$$
em que $k_i = g(k)_i$.

Desta forma a função f é definida por uma iteração de funções $f_0, f_1, \cdots, f_t$

$$f_0(x, k) = x \qquad , \qquad f_i(x, k) = r_i(f_{i-1}(x, k), g(k)_i) \quad i = 1, \cdots, t \qquad , \qquad f = f_t$$

Numa cifra iterada o acto de decifrar é também iterado: dado o criptograma y, a sequência y_i é reconstruída iterativamente do fim para o princípio,

$$y_t = y \qquad , \qquad y_{i-1} = r_i^{-1}(y_i, k_i) \quad i = t, ..., 1 \qquad , \qquad x = y_0$$

Cifras sequenciais ("stream ciphers"):

São primitivas criptográficas descritas por funcionais $f : \mathbb{B}^t \times \mathbb{B}^\infty \longrightarrow \mathbb{B}^\infty$ tais que:

(i) Para toda a chave k e todo i, o valor $f(k,u) \restriction i$ depende apenas de $u \restriction i$; isto é, se for $u \restriction i = v \restriction i$, então será $f(k,u) \restriction i = f(k,v) \restriction i$.

(ii) Para toda a chave k e todo i, a função $x \in \mathbb{B}^i \mapsto f(k, \uparrow x) \restriction i$ é uma permutação.

Este conceito de cifra sequencial efectua um processamento sequencial de mensagens *orientado ao bit*. Em cada estado constrói-se o bit de ordem i do criptograma a partir dos bits do texto de ordem $j \leq i$.

Nomeadamente, uma forma comum de cifra sequencial orientada ao bit, é descrita por uma função f da forma

$$f(k,u) \;=\; s(k) \oplus u \tag{107}$$

sendo $s : \mathbb{B}^m \longrightarrow \mathbb{B}^\infty$ um **gerador pseudo-aleatório** de bits.

Neste caso o bit de ordem i do criptograma depende apenas do bit com a mesma ordem i do texto. Cifras com esta propriedade designam-se por **mono-alfabéticas**. Cifras onde o bit de ordem i depende efectivamente de todos os bits do texto de ordem $j \leq i$ e, eventualmente, de alguns de ordem $j > i$, designam-se por **poli-alfabéticas**.

Cifras sequenciais por blocos

Por vezes é necessário generalizar o conceito de cifra sequencial para o de uma cifra sequencial *orientada ao bloco*. Neste modelo tanto o texto como o criptograma são vistos como sequências infinitas de blocos de tamanho fixo n; o cálculo do bloco de ordem i do criptograma recorre ao conhecimento de todos os blocos de ordem $j \leq i$ do texto.

Formalmente, uma cifra sequencial orientada a blocos de tamanho n, com chaves de tamanho m, é uma funcional $f : \mathbb{B}^t \times \mathbb{B}^* \longrightarrow \mathbb{B}^*$ tal que:

(i) Para toda a chave k e todo i múltiplo de n, se for $u \restriction i = v \restriction i$, então $f(k,u) \restriction i = f(k,v) \restriction i$.

(ii) Para toda a chave k e todo i múltiplo de n, a função $x \in \mathbb{B}^i \mapsto f(k, \uparrow x) \restriction i$ é uma permutação.

Essencialmente, em relação à definição inicial de cifra sequencial, nesta definição substitui-se o quantificador "*todo* i" por "*todo* i *múltiplo de* n". Isto implica que um bit do criptograma de ordem i arbitrária pode depender de bits do texto que têm ordems superiores.

Uma cifra sequencial mono-alfabética orientada ao bloco de tamanho n, pode ser definida à custa de uma qualquer cifra de blocos $h : \mathbb{B}^t \times \mathbb{B}^n \longrightarrow \mathbb{B}^n$ e de um gerador de chaves $s : \mathbb{B}^t \longrightarrow (\mathbb{B}^t)^\infty$.

Para tal texto e o criptograma são sequências de blocos e define-se a funcional $f : \mathbb{B}^t \times (\mathbb{B}^n)^\infty \longrightarrow (\mathbb{B}^n)^\infty$ por

$$f(k\,,\,v) \quad = \quad h(s_0, x_0)\; h(s_1, x_1)\; \cdots\; h(s_i, x_i)\; \cdots \tag{108}$$

em que $v = x_0\, x_1 \cdots x_i \cdots$ é a sequência de blocos de "input" e $s(k) = s_0\, s_1 \cdots s_i \cdots$ é a sequência de chaves geradas a partir de k. Ou seja, o bloco de ordem i no criptograma é determinado por

$$y_i \quad = \quad f(k, v)_i \quad = \quad h(s_i, x_i) \tag{109}$$

Para construir uma cifra poli-alfabética necessita-se de uma estrutura um pouco mais complexa. Uma forma frequente, chamada **cifra com "trail" l** usa o mesmo gerador de chaves $s : \mathbb{B}^t \to (\mathbb{B}^t)^\infty$ da cifra mono-alfabética, mas introduz uma nova função $H : \mathbb{B}^* \times \mathbb{B}^n \to \mathbb{B}^n$ que é invertível no 2º argumento; isto é, para todo y, $H(y, \cdot)$ é uma permutação.

Então a sequência $\{y_i\}$ de blocos do criptograma é gerada por

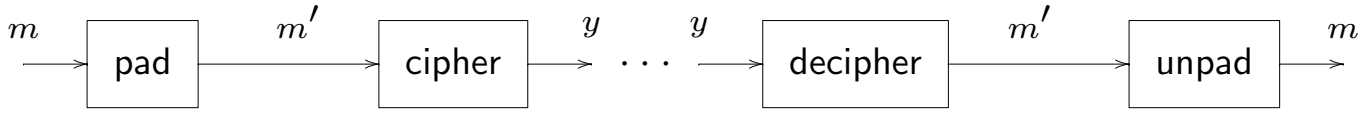
$$y_i \;=\; h(s_i, u_i) \qquad \text{com} \qquad u_i \;=\; H(y_{i-1} \| \cdots \| y_{i-l}\,,\; x_i) \tag{110}$$

Os l blocos do criptograma, anteriores a i, funcionam como chave que, com H, cifra x_i. O valor resultante u_i é, posterormente, cifrado (usado h) com a chave s_i gerada a partir de k.

Este tipo de construção levanta a questão de se saber como calcular os primeiro l blocos do criptograma. Note-se que, para calcular y_0, por exemplo, precisariamos de conhecer um "pseudo"-bloco y_{-l}. O problema resolve-se introduzindo l constantes que, independentemente do texto v, funcionam como blocos $y_{-l} \cdots y_{-1}$. Estas constantes são designadas por **vectores iniciais (iv)**.

"Padding"

Quando uma mensagem m, de comprimento arbitrário, está destinada a ser cifrada por uma cifra de blocos de tamanho fixo n, há necessidade de embeber a mensagem numa outra mensagem m' cujo comprimento é um múltiplo exacto do tamanho do bloco. Se o tamanho da nova mensagem m' for $t \cdot n$, então pode-se cifrar m' invocando t vezes a cifra de bloco n.



Este esquema exige uma função injectiva para embeber m em m' e, no final, uma função que recupere m sem ambiguidades. O mecanismo de "padding" exige duas funções:

1. Uma função injectiva pad : $\mathbb{B}^* \longrightarrow \mathbb{B}^*$ tal que, para todo $u \in \mathbb{B}^*$, $|\text{pad}(u)| = t \cdot n$, para algum $t > |u|/n$.
2. Uma função parcial unpad : $\mathbb{B}^* \rightharpoonup \mathbb{B}^*$ tal que $\text{unpad}(\text{pad}(u)) = u$, para todo u.

Modos de Cifras por blocos

Considere-se o problema de cifrar bit-strings infinitas usando uma cifra de blocos $h : \mathbb{B}^t \times \mathbb{B}^n \longrightarrow \mathbb{B}^n$ com chaves de tamanho k e blocos de tamanho n.

A forma mais simples de construir uma tal cifra, consiste em ver o "input" e o "output" como strings infinitas de blocos. e usar as construções (109) e (110) escolhendo formas particulares do gerador de chaves s e do "trail" H.

Formalmente queremos construir, usando h, uma funcional $f : \mathbb{B}^t \times (\mathbb{B}^n)^\infty \to (\mathbb{B}^n)^\infty$ que se comporte como uma cifra sequencial orientada a um bloco de tamanho n.

Numa cifra da forma (108) e (109) considere-se um gerador de chaves que se limita a repetir a chave k *ad infinitum*; isto é $s(k) = k^\infty$. Neste caso diz-se que a cifra h está em modo **electronic code book (ecb)**. No modo **ecb**, o bloco de orden i no criptograma é determinado por

$$y_i = h(k, x_i)$$

Em alternativa pode-se usar uma construção da forma (110), com o mesmo gerador de chaves $s(k) = k^\infty$, e um "trail" de tamanho 1, dado por $H(y_{i-1}, x_i) = y_{i-1} \oplus x_i$. Nesta construção diz-se que a cifra h está em modo **cipher-block chaining (cbc)**; bloco y_i, no modo **cbc**, é calculado por

$$y_i = h(k, x_i \oplus y_{i-1})$$

Uma abordagem alternativa consiste em ver o "input" e o "output" como "streams" de blocos com um tamanho m que é menor do que o tamnaho n do bloco usado por h. Os blocos y_i são gerados pela dupla recorrência

$$z_i = H(y_{i-1} \| z_{i-1}) \quad , \quad s_i = h(k, z_i) \restriction m \quad , \quad y_i = x_i \oplus s_i$$

em que $H : \mathbb{B}^* \to \mathbb{B}^n$ é uma qualquer função de "hash". Modos baseados nesta estrutura chamam-se **"feedback modes"**.

4.2 Arquitectura de Cifras por Blocos

Na maioria das cifras por blocos iteradas, os diversos "rounds", com eventual excepção do primeiro e do último, são iguais. São também formados por algumas componentes "standard". Dado que uma característica essencial das cifras por blocos é a sua eficiência computacional, essas componentes têm de ser de implementação muito simples.

O desafio essencial de qualquer arquitectura de cifra é o de construir um edifício extremamente seguro a partir de componentes muito simples e muito eficientes. Individualmente, cada uma das componentes pode não ser particularmente segura, mas a sua combinação produz a segurança pretedida.

A seguir indicam-se algumas das componentes básicas que ocorrem na construção de cifras.

Branqueamento ("**whitening**")

Sabe-se que, se $k \in \mathbb{B}^n$ for, nalgum sentido do termo, aleatório então todo $x \oplus k$ é também aleatório. Por isso a cifra $f(k, x) = x \oplus k$ dá segurança perfeita para ataques sem texto conhecido; não é possível conhecer x a partir de $y = x \oplus k$ sem conhecer a chave k.

No entanto, num ataque de texto conhecido (onde tanto x como y são conhecidos) determinar a chave k é trivial; basta fazer $k = y \oplus x$.

A cifra $f(k,x) \doteq x \oplus k$ é extremamente eficiente já que o `xor` de duas palavras de bits é directamente implementado num ciclo de máquina.

Funções Lineares

O "branqueamento" $x \oplus k$ determina, para cada k, uma função linear. Como sabemos, uma função $f : \mathbb{B}^n \longrightarrow \mathbb{B}^n$ é linear se verifica, para todo $x, y \in \mathbb{B}^n$ e $b \in \mathbb{B}$

$$f(x \oplus y) \;=\; f(x) \oplus f(y) \qquad , \qquad f(b\,x) \;=\; b\,f(x)$$

Funções lineares têm muitas representações; nomeadamente em $\mathrm{GF}(2)$ a função f é representada por uma matriz de bits $F \in \mathrm{GF}(2)^{n\times n}$; nessa representação tem-se $f(x) = F\,x$ usando as operações de álgebra matricial no corpo $\mathrm{GF}(2)$.

Também nesta representação, se a função $f(\cdot)$ for injectiva, então a matriz F tem uma inversa F^{-1} que é a representação matricial da função inversa f^{-1}

Normalmente uma função linear f é de implementação mais eficiente do que uma não linear e algumas formas particulares de funções lineares (como o "branqueamento") são mesmo extremamente eficientes.

Uma cifra $f(k,x)$, em que $f(k,\cdot)$ fosse linear para algum k, é vulnerável a um ataque muito simples. Sopunhamos que se conhecem vários criptogramas $y_i = f(k,x_i)$ e os respectivos textos x_i; seja $x = a_1\,x_1 \oplus a_2\,x_2 \cdots \oplus a_l\,x_l$,

com $a_i \in \mathrm{GF}(2)$, uma qualquer combinação linear dos diversos x_i. Então, mesmo sem conhecer a chave k, é possível calcular $y = f(k, x)$ usando a mesma combinação linear nos criptogramas; isto é, $y = a_1\, y_1 \oplus a_2\, y_2 \cdots \oplus a_l\, y_l$.

Nomeadamente, se os x_i formarem uma base do espaço vectorial $\mathrm{GF}(2)^n$, então o conhecimento de n pares (x_i, y_i) permite construir o criptograma de todos os 2^n possíveis pares (texto,criptograma), sem necessidade de conhecer a chave.

No entanto as funções lineares têm uma grande importância como componente de cifras porque uma construção da forma $y = f(x)$, sendo f linear, permite preservar a aleatoriedade de x em y.

□

Alguns exemplos de funções lineares de implementação muito eficiente.

Linear Feedback Shift Register (LFSR)

Sejam x_i, y_i os bits de ordem i dos vectores $x, y \in \mathrm{GF}(2)^n$; defina-se

$$y_0 = a_0\, x_0 + a_1\, x_1 + \cdots + a_{n-1}\, x_{n-1} \quad , \quad y_i = x_{i-1} \quad \text{para} \quad i = 1, \cdots, n-1$$

Esta transformação traduz-se no produto $y = F\, x$ em que a matriz F verifica $F_{1,i} = a_i$, $F_{i,i-1} = 1$ e $F_{i,j} = 0$ para os restantes índices. O vector $(a_0, \cdots, a_{n-1}) \in \mathrm{GF}(2)^n$ desina-se por **vector característico** do LFSR.

LFSR são uma componente muito importante de cifras sequenciais e têm a vantagem de ser implementadas em poucos ciclos máquina e directamente implementáveis em *hardware*.

Foram, durante muito tempo, usadas como componente de geradores de bit-strings pseudo-aleatórios. Um exemplo simples usa um LFSR definido por uma matriz F, do tipo atrás descrito, e a função traço $\mathrm{tr}(\cdot)$.

Gera-se uma sequência de bits b_i a partir da recorrência linear

$$x_0 \;=\; k \qquad , \qquad x_i \;=\; F\,x_{i-1} \quad \text{para } i = 1, \cdots \qquad , \quad b_i = \mathrm{tr}(x_i)$$

Esta construção pode-se generalizar usando t LFSR´s definidos por matrizes $F_1, \cdot, F_t$ e uma função booleana não-linear $\sigma : \mathrm{GF}(2)^t \longrightarrow \mathrm{GF}(2)$. Cada LFSR actua sobre um estado próprio e os traços dos diversos estados são combinados pela função σ.

A sequência de bits b_i é gerado pela recorrência

$$\begin{cases} x_{0,j} \;=\; k_j & j = 1, \cdots, t \\ x_{i,j} \;=\; F_j\, x_{i-1,j} & j = 1, \cdots, t \qquad i = 1, \cdots \\ b_i \;=\; \sigma(\mathrm{tr}(x_{i,1}), \cdots, \mathrm{tr}(x_{i,t})) \end{cases}$$

Transformações Pseudo-Hadamard (pHT)

Pode-se ver palavras $x, y \in \mathbb{B}^{2n}$ como pares de inteiros $(x_1, x_2), (y_1, y_2) \in \mathbb{Z}_n \times \mathbb{Z}_n$.

A transformação PHT define-se por $(y_1, y_2) = (2\,x_1 + x_2, x_1 + x_2)$ em que a adição $+$ e a multiplicação são feitas em $\mathbb{Z}_n$.

Assim $\begin{bmatrix} y_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ e esta transformação é, normalmente, efectuada num único ciclo máquina.

□

Vamos agora examinar funções não-lineares $f\mathbb{B}^n \longrightarrow \mathbb{B}^n$.

"Substitution boxes"

Funções não lineares genéricas $f : \mathbb{B}^n \longrightarrow \mathbb{B}^m$ são, normalmente, designadas por **substituion boxes** (ou, simplesmente, **S-boxes**).

Para um mesmo tamanho de bloco n, uma S-box $n \times n$ é muito mais difícil de implementar que uma função linear com a mesma dimensão. Por isso S-boxes são normalmente usadas com pequenos valores de n; enquanto que é relativamente simples implementar uma transformação linear para um bloco de 128 bits ou superior, as S-boxes não ultrapassam os 16 bits. Aliás, quase sempre as S-boxes são de 8 ou menos bits.

Quando n é pequeno é fácil implementar uma função f genérica como uma tabela em memória ROM; constrói-se uma memória com n linhas para o endereço de célula e cada célula contém m bits. Usando x como endereço de acesso à memória, e supondo que a célula de endereço x contém o valor $f(x)$, então, com um simples acesso à memória, calcula-se $f(x)$.

Obviamente que, atendendo à pequena capacidade de memória dos dispositivos que têm de executar estas cifras, esta estratégia só é viável se n for pequeno; por exemplo, uma S-box 8×8 requer `256 bytes` de memória; se for 16×16 requer `128 Kbytes`; se for 24×24 requer `48 Mbytes`; etc.

□

Quando são realmente necessárias funções não-lineares com elevados valores de n, então as únicas implementações possíveis são aquelas que aproveitam a forma particular destas funções.

Funções "bricklayer"

Considere-se palavras $x, y \in \mathbb{B}^n$ e considere-se um valor l que é divide n; seja $p = n/l$. Pode-se representar cada um dos elementos de $\mathbb{B}^n$ como um vector com p componentes em $\mathbb{B}^l$; isto é, usa-se o isomorfismos $\mathbb{B}^n \sim \left(\mathbb{B}^l\right)^p$.

Considere-se agora p S-boxes $f_1, \cdots, f_p \colon \mathbb{B}^l \longrightarrow \mathbb{B}^l$. Então, pode-se definir uma função não linear por

$$f \;:\; x_1 \| x_2 \| \cdots \| x_p \quad \mapsto \quad f_1(x_1) \| f_2(x_2) \| \cdots \| f_p(x_p) \tag{111}$$

Neste tipo de função as S-boxes têm dimensão $l \times l$ e podem ser razoavelmente implementáveis. Cada uma destas transformações actua apenas sob uma das componentes do "input" x e produz a componente, com a mesma ordem, no "output" y. O "output" é construído bloco a bloco e não é de estranhar que este tipo de funções se designe por "bricklayer".

Suponhemos que f é esta função "bricklayer" e que um atacante conhece x e y e procura determinar k tal que $y = f(x \oplus k)$. Normalmente, para uma função não-linear bem escolhida, este problema teria complexidade exponencial com n. Porém, para uma função "bricklayer", o problema resume-se a p problemas diferentes de tamanho l. Ou seja, passamos de uma complexidade $O((2^l)^p)$ para uma complexidade $O(p\,2^l)$.

Por este motivo, funções "bricklayer" não podem ser usadas isoladamente como "rounds" de cifras. O seu "output" têm de ser pós-processado (normalmente por uma transformação linear) de modo que, no final, se tenha uma função $n \times n$ cujo comportamento se aproxime de uma função ideal.

Circuitos de Feistel

Uma dificuldade importante na escolha de funções não-lineares em "rounds" de cifras está no cumprimento do requisito de que os "rounds" têm de ser invertíveis. Construir e implementar eficientemente funções não-lineares invertíveis é um problema difícil.

Com o DES (que serviu de standard criptográfico no sistema financeiro durante quase 30 anos) surgiu uma construção que tomou o nome do principal autor do DES: os *circuitos de Feistel*.

Vamos tomar uma função $h\colon \mathbb{B}^n \longrightarrow \mathbb{B}^n$, não-linear mas não necessaramente invertível. Vamos contruir uma função $f\colon \mathbb{B}^{2n} \longrightarrow \mathbb{B}^{2n}$ do seguinte modo

$$f \quad : \quad (x_1, x_2) \quad \mapsto \quad (x_2 \oplus h(x_1) \ , \ x_1) \tag{112}$$

A 1ª componente do "input" é transferida para a 2ª componente do "output" sem qualquer transformação; a 2ª componente do "input" é misturado com $h(x_1)$ para produzir a 1ª componente do "output".

Das equações

$$y_1 \ = \ x_2 \oplus h(x_1) \qquad , \qquad y_2 \ = \ x_1$$

concluímos

$$x_1 \ = \ y_2 \qquad , \qquad x_2 \ = \ y_1 \oplus h(y_2)$$

Desta forma a inversa de f é a função

$$f^{-1} \quad : \quad (y_1, y_2) \quad \mapsto \quad (y_2 \ , \ y_1 \oplus h(y_2)) \tag{113}$$

que é uma função análoga a f. De facto, com a ajuda da função "troca" definda por $t\colon (x, y) \mapsto (y, x)$, vemos que

$$f^{-1} \quad = \quad t \circ f \circ t \tag{114}$$

4.3 Rijndael-AES

Em 2/10/00 o *National Institute of Standards and Technology* (NIST) do *US Department of Commerce* recomendoa adopção do algoritmo designado por **Rijndael**como **Advanced Encryption Standard** (AES).

O AES foi adoptado em 2002 como *standard* para criptografia simétrica e substituiu *standard* designado por DES (Data Encryption Standard) que, na sua versão inicial ou na versão modificada designada como TRIPLEDES, tem vindo a ser usado desde 1977 na segurança de todo o sector financeiro bem como em inúmeras outras situações.

□

Alguns pontos fundametais sobre a arquitectura do AES

1. O Rijndael é uma cifra iterada com blocos de tamanho variável ($N_b \times 32$ bits, com $N_b = 4, 6$ ou 8) e chaves de tamanho variável ($N_k \times 32$ bits, com $N_k = 4, 6$ ou 8). O algoritmo tem 10 iterações (*rounds*) quando $N_b = N_k = 4$, tem 14 iterações quando $N_b = 8$ ou $N_k = 8$ e tem 12 iterações nos restantes casos.

2. O Rijndael **não é** um circuito de Feistel. É uma cifra orientada ao byte em que cada iteração (*round*) é uma substituição composta por três transformações invertíveis designadas por **camadas** (*layers*):
   - **Camada não linear (*non-linear layer*)**
     aplicação paralela de S-*boxes* destinada a evitar correlações cruzadas,

- **Camada linear (*linear mixing layer*)**
  garante a difusão dos bits transformados,
- **Mistura de chaves (*key addition layer*)**
  mistura das sub-chaves por simples operações **XOR**

3. Antes da primeira iteração é aplicada uma mistura de chaves prévia. A motivação para este tipo de operação (conhecido por *whitening*) reside no facto que qualquer transformação invertível que ocorra antes da primeira mistura de chaves pode ser retirada sem afectar as propriedades de segurança da cifra; é, portanto, redundante.

   De modo a melhorar a fase de decifragem a camada linear da última iteração é diferente da dos restantes.

Para facilitar a apresentação a descrição do algoritmo será aqui feita para o caso mais simples: blocos e chaves de 128 bits ($N_b = N_k = 4$) e 10 iterações.

Round do Rijndael

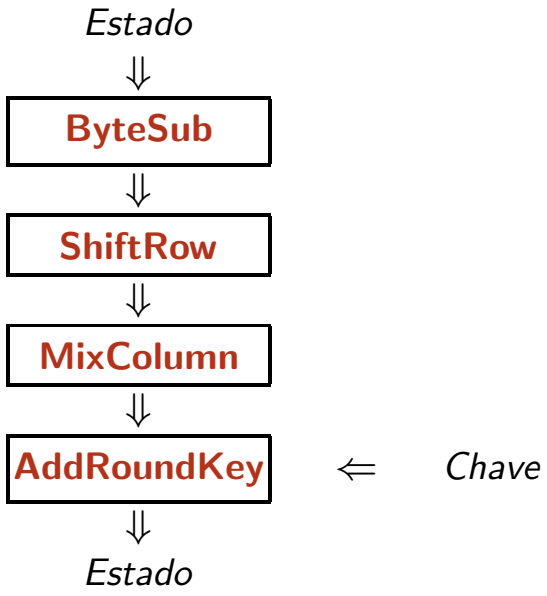
A definição de *round* é uma transformação sobre um **estado** de 128 bits organizado como uma matriz 4×4 de bytes. Cada *round* usa uma única sub-chave de 128 bits também organizada como uma matriz 4×4 de bytes. Ambas matrizes são sequencialmente organizadas "em colunas": i.e. a posição (i, j) da matriz corresponde à posição $i + 4 \times j$ num vector uni-dimensional.

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

Estado

$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

Chave



Cada uma dos primeiros 9 *rounds* do Rijndael é uma sequência de 4 transformações sobre o estado acima representadas. O último *round* é análogo mas não tem a transformação **MixColumn**. Adicionalmente, antes do 1º *round* há uma transformação **AddRoundKey**.

Globalmente a sequência de transformações é

$$\begin{array}{l}\texttt{AddRoundKey} \Rightarrow \\ (\ \texttt{ByteSub} \Rightarrow \texttt{ShiftRow} \Rightarrow \texttt{MixColumn} \Rightarrow \texttt{AddRoundKey} \Rightarrow\ ) \quad \times 9 \\ \texttt{ByteSub} \Rightarrow \texttt{ShiftRow} \Rightarrow \texttt{AddRoundKey}\end{array}$$

As duas transformações ByteSub e ShiftRow formam a *Non Linear Layer*. A transformação MixColumn forma a *Linear Mixing Layer* enquanto que AddRoundKey implementa a *Key Addition Layer*.

Representação dos *bytes*

Cada byte (organizado do bit mais significativo para o menos significativo) $b_7\, b_6\, \cdots\, b_2\, b_1\, b_0$ é interpretado como um polinómio em Y do 7º grau

$$b_7 \times Y^7 + b_6 \times Y^6 + \cdots + b_2 \times Y^2 + b_1 \times Y + b_0$$

São definidas operações de soma e multiplicação de bytes do seguinte modo:

Soma $A \oplus B$ calcula-se fazendo o **XOR** bit-a-bit dos respectivos coeficientes.

Multiplicação $A \otimes B$ calcula-se multiplicando os polinómios e reduzindo o resultado módulo $Y^8+Y^4+Y^3+Y+1$.

EXEMPLO

$$\begin{aligned}\texttt{ff} \otimes \texttt{03} &= (Y^7+Y^6+Y^5+Y^4+Y^3+Y^2+Y+1) \otimes (Y+1) = \\ &= (Y^8+1) \pmod{Y^8+Y^4+Y^3+Y+1} = Y^4+Y^3+Y = \\ &= \texttt{1a}\end{aligned}$$

As operações de multiplicação e similares (como o cálculo da inversa) podem ser realizadas usando uma *tabela de logaritmos*: cada byte $b \neq 0$, pode ser escrito na forma

$$b = g^e \pmod{Y^8+Y^4+Y^3+Y+1} \qquad \text{com} \quad g = \texttt{ff}$$

para um único expoente $e \in \{0 \cdots 254\}$ determinado pela tabela 4 apresentada na página 292. Para $e > 254$, tem-se $g^e \equiv g^{e \bmod 255}$.

EXEMPLO

Vamos usar a tabela para calcular o produto $\texttt{ff} \otimes \texttt{03}$ e o inverso $\texttt{aa}^{-1}$.

Vemos que $\texttt{ff} = g^1$ e $\texttt{03} = g^{73}$. Logo $\texttt{ff} \otimes \texttt{03} = g^1 \otimes g^{73} = g^{74} = \texttt{1a}$.

Vemos que $\texttt{aa} = g^{223}$; logo $\texttt{aa}^{-1} = g^{-223 \pmod{255}}$. Como $-223 \equiv (255-223) \equiv 32 \pmod{255}$ o inverso será $\texttt{aa}^{-1} = g^{32} = \texttt{12}$.

Uma tabela de logaritmos e a sua inversa podem ser facilmente armazenadas numa S-Box (que se resume a um vector de 256 bytes). A implementação de operações de multiplicação ou calculo de inversas resumem-se a algumas **XOR** de bytes e procuras na tabela; deste modo são computacionalmente muito eficientes.

Transformação `ByteSub`

Actua em paralelo e de modo uniforme sobre cada um dos bytes da matriz do estado transformando-o do modo seguinte

$$b(Y) \quad \Longrightarrow \quad u(Y) \oplus v(Y) \times b^{-1}(Y) \pmod{Y^8+1}$$

$$\text{com} \quad u \equiv \texttt{c6} \quad \text{e} \quad v \equiv \texttt{f1}$$

Notas

1. A característica não linear da cifra é basicamente implementada na transformação $b \Longrightarrow b^{-1}$.
2. A multiplicação desta inversa por v usa um polinómio (Y^8+1) diferente do usado nas operações algébricas gerais sobre os bytes (aumentando a não-linearidade intrínseca). v é escolhido de modo que $v^{-1} \pmod{Y^8+1}$ exista.
3. A transformação pode ser implementada eficientemente numa S-Box 8×8 invertível. Na tabela 4 (página 291) é apresentada essa S-Box. A transformação inversa implementa-se usando esta mesma S-Box mas em sentido oposto.

4. A constante u é escolhida de modo que a S-Box não tenha pontos fixos (pontos b onde $S(b) = b$) nem anti-pontos fixos (pontos b onde $S(b) = \bar{b}$).

Transformação `ShiftRow`

As linhas da matriz que representa o estado sofrem uma rotação circular: a primeira não é rodada, a segunda roda uma posição, a terceira roda duas posições e a última roda 3 posições

$$\begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} \Longrightarrow \begin{array}{|c|c|c|c|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,3} & a_{1,0} & a_{1,1} & a_{1,2} \\ a_{2,2} & a_{2,3} & a_{2,0} & a_{2,1} \\ a_{3,1} & a_{3,2} & a_{3,3} & a_{3,0} \\ \hline \end{array}$$

A transformação inversa efectua-se rodando as linhas o mesmo número de posições mas em sentido inverso.

Transformação `MixColumn`

Nesta transformação do estado cada coluna da matriz que representa o estado é interpretada como um polinómio em X do 3º grau

$$\mathbf{a}_j(X) \equiv a_{0,j} + a_{1,j} \times X + a_{2,j} \times X^2 + a_{3,j} \times X^3$$

Nestes polinómios a soma é efectuada componente a componente e a multiplicação é efectuada módulo (X^4+1). As operações básicas são a multiplicação de cada $\mathbf{a}(X)$ por uma matriz $\mathbf{C}$ ou pela sua inversa $\mathbf{C}^{-1}$ dadas por

$$\mathbf{C}\ \mathbf{a}_j \equiv \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \otimes \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix} \qquad \mathbf{C}^{-1}\ \mathbf{a}_j \equiv \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \otimes \begin{bmatrix} a_{0,j} \\ a_{1,j} \\ a_{2,j} \\ a_{3,j} \end{bmatrix}$$

A operação **MixColumn** multiplica por **C** cada uma das 4 colunas do estado

$$\begin{bmatrix} \mathbf{a}_0 & \mathbf{a}_1 & \mathbf{a_2} & \mathbf{a}_3 \end{bmatrix} \implies \begin{bmatrix} \mathbf{C}\,\mathbf{a}_0 & \mathbf{C}\,\mathbf{a}_1 & \mathbf{C}\,\mathbf{a}_2 & \mathbf{C}\,\mathbf{a}_3 \end{bmatrix}$$

A operação inversa de **MixColumn** é análoga mas usa a matriz $\mathbf{C}^{-1}$.

Transformação `AddRoundKey`

É efectuado um **XOR** byte-a-byte do estado e da sub-chave

$$\begin{array}{|cccc|} \hline a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \\ \hline \end{array} \implies \begin{array}{|cccc|} \hline a_{0,0} \oplus k_{0,0} & a_{0,1} \oplus k_{0,1} & a_{0,2} \oplus k_{0,2} & a_{0,3} \oplus k_{0,3} \\ a_{1,0} \oplus k_{1,0} & a_{1,1} \oplus k_{1,1} & a_{1,2} \oplus k_{1,2} & a_{1,3} \oplus k_{1,3} \\ a_{2,0} \oplus k_{2,0} & a_{2,1} \oplus k_{2,1} & a_{2,2} \oplus k_{2,2} & a_{2,3} \oplus k_{2,3} \\ a_{3,0} \oplus k_{3,0} & a_{3,1} \oplus k_{3,1} & a_{3,2} \oplus k_{3,2} & a_{3,3} \oplus k_{3,3} \\ \hline \end{array}$$

A transformação AddRoundKey coincide com a sua inversa.

A cifra inversa

Como cada transformação de um *round* é invertível, a sua inversa constrói-se por aplicação das inversas das diversas componentes pela ordem inversa. Notando que $\texttt{AddRoundKey}^{-1} = \texttt{AddRoundKey}$ a transformação será

$$\texttt{AddRoundKey} \Rightarrow \texttt{MixColumn}^{-1} \Rightarrow \texttt{ShiftRow}^{-1} \Rightarrow \texttt{ByteSub}^{-1}$$

O primeiro *round* não tem a transformação $\texttt{MixColumn}^{-1}$ e o último *round* é seguido de uma aplicação extra de AddRoundKey. As sub-chaves são usadas pela ordem inversa da usada na cifragem.

Note-se que

1. A ordem relativa das transformações ByteSub e ShiftRow é indiferente porque ByteSub do mesmo modo para qualquer byte e ShiftRow apenas muda a ordem relativa dos bytes.
2. A ordem relativa de MixColumn e AddRoundKey é indiferente porque MixColumn é uma transformação linear, AddRoundKey é uma soma e as transformações lineares $f(\cdot)$ verificam $f(a+k) = f(a) + f(k)$.

Se notar-mos que o 1º *round* não tem a transformação $\texttt{MixColumn}^{-1}$ vemos então (usando estas duas transposições)

que a decifragem é dada pela sequência exactamente análoga à da cifragem

$$\begin{array}{l} \texttt{AddRoundKey} \Rightarrow \\ (\ \texttt{ByteSub}^{-1} \Rightarrow \texttt{ShiftRow}^{-1} \Rightarrow \texttt{MixColumn}^{-1} \Rightarrow \texttt{AddRoundKey} \Rightarrow\ ) \quad \times 9 \\ \texttt{ByteSub}^{-1} \Rightarrow \texttt{ShiftRow}^{-1} \Rightarrow \texttt{AddRoundKey} \end{array}$$

4.4 Programador de Chaves

O programador de chaves gera, a partir da chave principal, tantas sub-chaves (com o mesmo tamanho da chave principal) quantos os *rounds* mais 1.

Note-se que cada *round* contém uma transformação **AddRoundKey** e, adicionalmente, antes do primeiro round existe uma operação **AddRoundKey** extra.

Vamos ilustrar a geração das sub-chaves para o caso em que $N_b = N_k = 4$ e são usados 10 *rounds*. Os princípios gerais são os seguintes:

- A `RoundKey` é um vector unidimensional de palavras de 32 bits (4 bytes) e de tamanho 4×11 . Sequencialmente são usadas 4 palavras deste vector para formar cada uma das sub-chaves.
- A chave principal (designada `CipherKey`) é expandida para a `RoundKey` segundo o algoritmo seguinte.
  1. O vector `RoundKey` é inicializado com 11 cópias do vector `CipherKey`.
  2. Para $i = 4$ até 44 a palavra na posição i é substituída pelo **XOR** da palavra na posição $i - 4$ e pela palavra w calculada da seguinte forma.
     (a) Se i não é múltiplo de 4 usa-se para w a palavra na posição $i - 1$,
     (b) Se i é múltiplo de 4 e se for $k = i/4$ e s a palavra na posição $i - 1$, então

$$w = \texttt{ByteSub4}(\texttt{rot4}(s)) \oplus \texttt{cons}(k)$$

em que:

i. $\texttt{cons}(k)$ é uma palavra constante

$$(Y^k, 00, 00, 00)$$

em que o 1º byte é representado pelo polinómio Y^k e os restantes três bytes são 00.

ii. $\texttt{rot4}(s)$ roda os 4 bytes de s uma posição:

$$(b_0, b_1, b_2, b_3) \Rightarrow (b_1, b_2, b_3, b_0)$$

iii. $\texttt{ByteSub4}(\cdot)$ aplica a S-Box ByteSub a cada um dos 4 bytes da palavra.

Tabelas AES

S-Box da cifra Rijndael

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	c6	37	87	47	df	46	6	ac	f3	e0	86	42	1f	8d	4a	97
1	5c	d8	6c	27	5f	65	84	ff	2a	bd	da	a	39	ba	d7	fc
2	8b	2f	c9	92	93	3	8f	3c	b3	aa	ae	ef	e7	7d	e3	a1
3	b0	8c	c2	cc	71	99	a0	59	80	d1	f8	de	4e	82	db	a7
4	60	c8	32	51	41	16	55	fa	d5	43	9d	cb	62	ce	2	b8
5	c5	ed	f0	2e	f2	3f	eb	45	56	4c	1b	63	54	34	75	c
6	fd	e	5a	4f	c4	24	c3	a8	a4	6f	d0	7	f5	33	9	7a
7	e5	ca	f4	8	d9	29	73	af	3b	9b	5d	e2	f1	f	cf	dd
8	2c	30	c1	3e	5	89	b4	81	bc	8a	17	23	b6	25	61	c7
9	f6	e8	4	3d	d2	52	f9	78	94	1e	7b	b1	1d	15	40	4d
a	fe	d3	53	50	64	90	b2	35	dc	cd	3a	d6	e9	a9	be	67
b	8e	7c	83	26	28	ad	14	6a	36	95	bf	5e	a6	57	1a	70
c	5b	77	a2	12	31	9a	bb	9c	7e	2d	b7	1	44	2b	48	58
d	f7	13	ab	96	74	c0	9f	10	e6	a3	85	6b	98	ec	21	19
e	ee	7f	79	e1	66	6d	18	b9	49	11	88	6e	1c	a5	72	d
f	38	ea	68	20	b	9e	d4	76	e4	69	22	0	fb	b5	4b	91

Tabela de logaritmos de gerador $g = \texttt{ff}$ em GF(2^8) com polinómio carcaterístico $Y^8 + Y^4 + Y^3 + Y + 1$.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	-	0	40	73	80	146	113	174	120	247	186	197	153	34	214	219
1	160	37	32	2	226	93	237	107	193	89	74	65	254	15	4	64
2	200	137	77	208	72	245	42	88	11	69	133	162	22	138	147	90
3	233	116	129	110	114	75	105	78	39	180	55	192	44	21	104	166
4	240	239	177	27	117	139	248	94	112	86	30	253	82	10	128	8
5	51	68	109	189	173	183	202	14	62	151	178	131	187	238	130	148
6	18	125	156	210	169	26	150	228	154	161	115	12	145	19	118	63
7	79	163	220	3	95	132	232	211	84	176	61	142	144	235	206	231
8	25	49	24	230	217	252	67	53	157	207	179	249	33	215	134	106
9	152	140	126	236	70	76	38	35	122	29	50	98	168	97	48	205
a	91	111	108	198	149	28	229	143	213	46	223	225	242	199	54	99
b	102	136	191	195	218	123	171	92	227	121	23	234	170	85	188	241
c	58	244	165	57	196	100	250	36	209	167	66	175	190	9	13	212
d	194	81	201	45	155	96	52	83	185	6	59	159	158	71	103	243
e	119	221	203	31	5	41	43	56	135	127	172	224	17	204	251	60
f	124	47	216	141	101	7	182	222	184	87	20	164	246	181	16	1

Na talela 4 a A entrada $(\texttt{d}_1, \texttt{d}_2)$ contém o expoente e tal que $b = g^e$ se escreve, em hexadecimal $\texttt{d}_1\texttt{d}_2$.

4.5 Outras Cifras por Blocos

Nos últimos anos, para além da cigra Rijndael adoptada como cifra AES, outras cifras mereceram atenção. Nomeadamente

Twofish cifra patrocinada por Bruce Schneier ao concurso AES e, durante bastante tempo, considerada como favorita. É uma cifra de construção que se pode classificar como "tradicional". Usa o conhecimento experimental para propor uma arquitectura de Feistel dupla.

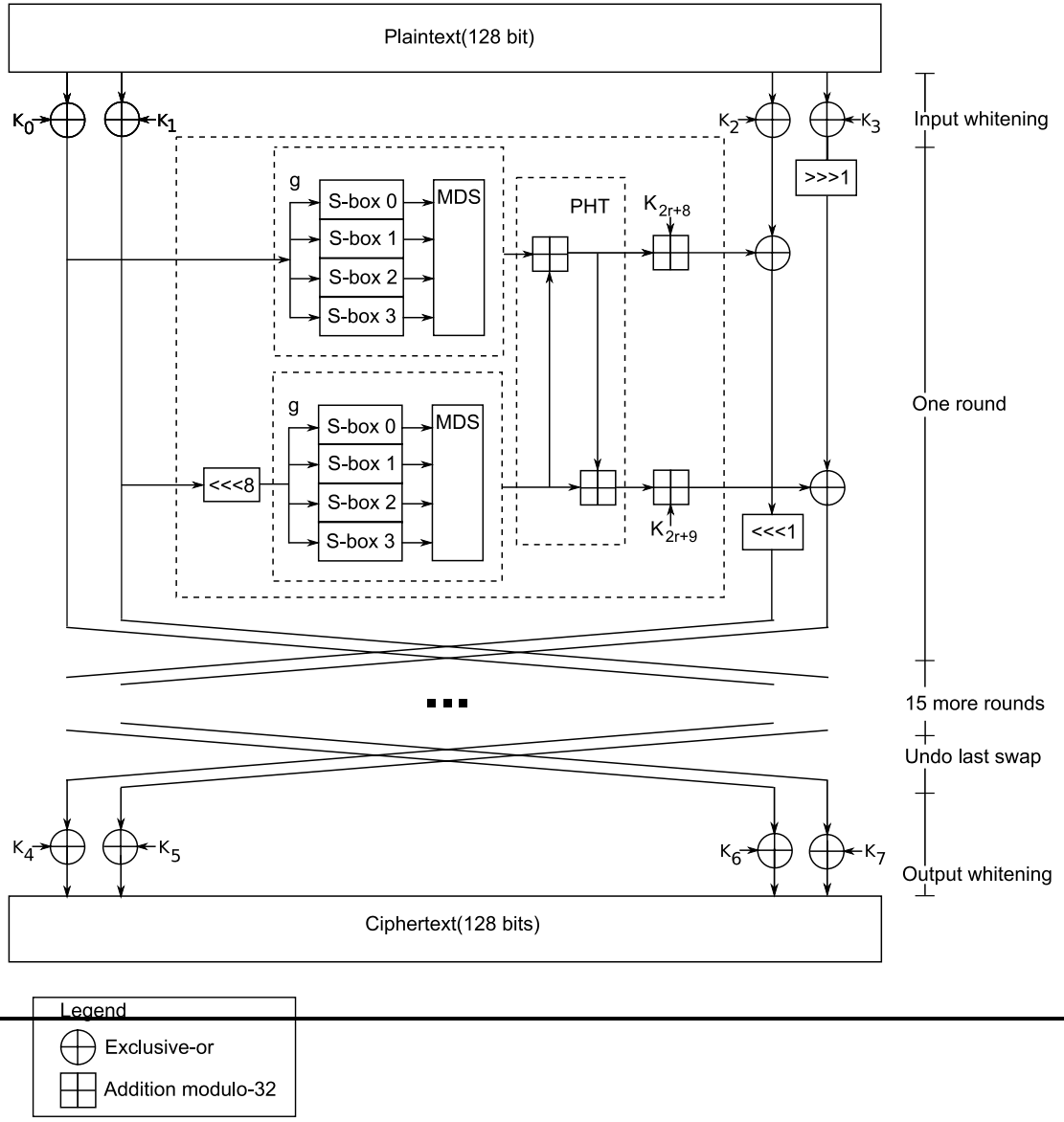
Kasumi Retirado do "ETSI 3rd Generation Partnership Project (3GPP) Tecnical Specification"

> *"The 3GPP Confidentiality and Integrity Algorithms f8 & f9 have been developed through the collaborative efforts of the European Telecommunications Standards Institute (ETSI), the Association of Radio Industries and Businesses (ARIB), the Telecommunications Technology Association (TTA), the T1 Committee. The f8 & f9 Algorithms Specifications may be used only for the development and operation of 3G Mobile Communications and services."*

Twofish é uma cifra que, na versão base, opera com blocos de 128 bits e chaves de 128 bits. Cada "round" é um circuito de Feistel duplo, cuja arquitectura geral está respresentada na figura seguinte.

MDS é uma transformação linear (matriz difusora) destinada a distribuir os "outputs" das S-Boxes.

PHT (*pseudo Hadamard Transformation*) é uma transformação de mistura dos dois circuitos de Feistel.



Kasumi

Na terminologia KASUMI, `f8` designa o algoritmo que implementa 8 "rounds" de cifra enquanto que `f9` é uma função de *hash* derivada de `f8`.

KASUMI é uma cifra de Feitsel com 8 "rounds" que usa blocos de 64 bits e chaves de 128 bits. A representação esquemática de um "round" Kasumi é feita nos seguintes diagramas.

Annex 1 (informative):
Figures of the KASUMI Algorithm

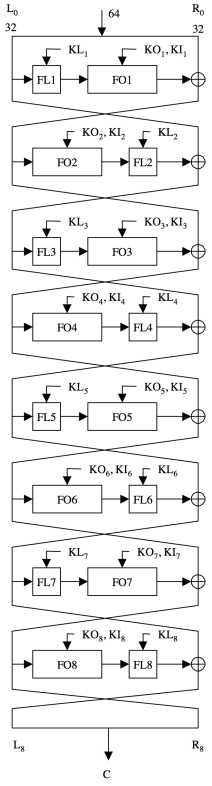


Fig. 1: KASUMI

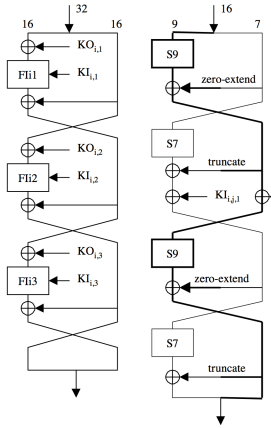


Fig.2: FO Function

Fig.3: FI Function

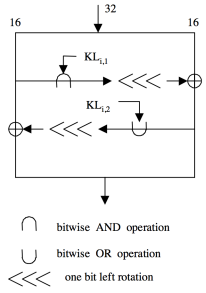


Fig.4: FL Function

4.6 Condições de Segurança em Cifras Simétricas

Segurança Orientada à Incerteza

A abordagem clássica à noção de segurança assenta nos conceitos de preservação de incerteza; essencialmente traduz a ideia de que a observação de um determinada valor do criptograma não dá qualquer informação sobre o eventual texto que lhe tenha dado origem.

Seja ψ uma função parcial recursiva e X, Y conjuntos finitos. Defina-se a relação $X \overset{\psi}{\hookrightarrow} Y$ por

$$X \overset{\psi}{\hookrightarrow} Y \quad = \quad \{\, x\|y \mid x \in X \wedge y \in Y \wedge (\exists k)\,[\,\psi(k\|x) \simeq y\,]\,\} \tag{115}$$

Tome-se um qualquer medida de incerteza ϑ definível em conjuntos finitos. Neste contexto

170 Noção
*A função ψ é (ϑ, X, Y) -**perfeitamente segura** se e só se $\vartheta(X \overset{\psi}{\hookrightarrow} Y) \geq \vartheta(Y) + \vartheta(X)$. A função é ϑ-**perfeitamente segura** se e só se for (ϑ, X, Y) -segura para todos conjuntos X, Y finitos.*

Exemplo 34:
Considere-se cifras por blocos $f : \mathbb{B}^t \times \mathbb{B}^n \to \mathbb{B}^n$. Temos $X = Y = \mathbb{B}^n$ e, usando ϑ como a incerteza de Hartley ($\vartheta(A) = \log_2[\![A]\!]$), será $\vartheta(X) = \vartheta(Y) = n$.

Vamos analisar dois exemplos de tais cifras vendo em que condições podem ou não ser ϑ-perfeitamente seguras.

Para simplificar a notação vamos designar por C_f a relação $\mathbb{B}^n \xhookrightarrow{f} \mathbb{B}^n$; f será ϑ, X, Y-perfeitamente segura se $\vartheta(C_f) = \vartheta(X) + \vartheta(Y) = 2\,n$.

1. **Cifra de Vernam** ("one-time pad") $f(k, x) = k \oplus x$.

$$C_f \quad \sim \quad \{\, (x, y) \in \mathbb{B}^n \times \mathbb{B}^n \mid \exists k \cdot y = x \oplus k \,\}$$

   Como, para todo $x, y \in \mathbb{B}^n$, existe sempre um k tal que $y = k \oplus x$ (nomeadamente $k = x \oplus y$), tem-se $C_f \sim \mathbb{B}^{2\,n}$. Portanto $\vartheta(C_f) = 2\,n$ e a cifra é perfeitamente segura.

2. **Cifra "perfeitamente insegura"** $f(k, x) \simeq g(x)$ para alguma permutação g.

   Tem-se $(\exists k)\,[f(k, x) \simeq y] \equiv [g(x) \simeq y]$; i.e. a "cifra" f não usa qualquer chave. Logo

$$C_f \sim \{\, (x, y) \in \mathbb{B}^n \times \mathbb{B}^n \mid \exists k \cdot f(k, x) \simeq y \,\} = \{\, (x, g(x)) \mid x \in \mathbb{B}^n \,\} \sim \mathbb{B}^n$$

   Portanto $\vartheta(C_f) = n < 2\,n$; como seria de esperar, f não é perfeitamente segura.

Nota

Estes dois casos descrevem situações extremas: segurança perfeita e insegurança perfeita. Note-se que, porque f é uma cifra, é

uma função total e, para cada x, existe sempre um valor y (pelo menos) no qual o predicado $(\exists k)\,[f(k\|x) \simeq y]$ é válido. Por isso, tem-se sempre $\vartheta(X \overset{f}{\hookrightarrow} Y) \geq \vartheta(X)$; por outro lado, como $(X \overset{f}{\hookrightarrow} Y) \subseteq X \times Y$, a incerteza de Hartley garante que $\vartheta(X \overset{f}{\hookrightarrow} Y) \leq \vartheta(X) + \vartheta(Y)$. Desta forma, em cifras por blocos, $\vartheta(X \overset{f}{\hookrightarrow} Y) - \vartheta(X)$ é sempre um valor compreendido entre 0 e $\vartheta(Y)$. Na primeira hipótese temos insegurança perfeita e na segunda temos segurança perfeita.

O exemplo anterior dá algumas pistas sobre o papel da dimensão do espaço e chaves. Começamos por ver o seguinte resultado

171 LEMA *Para todos conjuntos X, Y e toda a função parcial recursiva ψ tem-se*

$$X \overset{\psi}{\hookrightarrow} Y \;\sim \biguplus_{x \in X} \mathrm{rng}(\psi_x) \cap Y \tag{116}$$

em que $\uplus$ denota a união disjunta de conjuntos e $\{\psi_x\}$ é a sequência de funções parciais recursivas definidas por $\psi_x(k) = \psi(k\|x)$.

Esboço de prova
O lado direito em (115) pode-se reescrever $\{\, x\|y \mid x \in X \wedge y \in \mathrm{rng}(\psi_x)\,\} \cap Y$.

172 FACTO
Se ϑ é uma medida de incerteza monotónica[42] *e ψ é ϑ, X, Y-perfeitamente seguro, então* $\mathrm{rng}(\psi_x) \supseteq Y$, *para*

[42] Isto é, $A \subseteq B$ implica $\vartheta(A) \leq \vartheta(B)$.

todo $x \in X$. Adicionalmente, se ϑ for a incerteza de Hartley, verifica-se $\vartheta(\mathrm{dom}(\psi_x)) \geq \vartheta(Y)$.

Prova A primeira asserção é consequência do lema 171 e da hipótese $\vartheta(X \overset{\psi}{\hookrightarrow} Y) \geq \vartheta(X) + \vartheta(Y)$. A segunda asserção resulta do facto de $[\![\mathrm{dom}(\psi_x)]\!] \geq [\![\mathrm{rng}(\psi_x)]\!]$, para todo ψ_x.

Como consequência

173 TEOREMA
Seja ϑ a incerteza de Hartley. Uma cifra por blocos $f : \mathbb{B}^t \times \mathbb{B}^n \to \mathbb{B}^n$ só é ϑ-perfeitamente segura se for $t \geq n$.

Prova
A cifra é modelada pela função parcial recursiva ψ que verifica $\psi(k\|x)\downarrow \Leftrightarrow |k| = t \wedge |x| = n$ e, ainda, $\psi(k\|x)\downarrow \Rightarrow \psi(k\|x) \simeq f(k,x)$. Dado que f é total, tem-se, para todo x, $\mathrm{dom}(\psi_x) \sim \mathbb{B}^t$ e $\vartheta(\mathrm{dom}(\psi_x)) = t$. O facto 172 diz-nos que f só pode ser perfeitamente segura quando for $t \geq n$.

□

Quando se determina a incerteza $\vartheta(A)$ de um conjunto A, estamos a colocar um limite superior ao esforço computacional para encontrar um elemento $a \in A$ que verifique uma **propriedade discriminadora** arbitrária ρ; por exemplo, uma propriedade que verifique $\rho(a) \simeq 1 \Leftrightarrow [a = x]$, para alguma x desconhecido.

A incerteza $\vartheta(A)$ deve traduzir a intuição de que

Os algorimos que decidem $(\exists a \in A) \cdot \rho(a)$ *têm, independentemente de* ρ*, um limite de complexidade* $O(2^{\vartheta(A)})$.

Por isso faz sentido associar a incerteza do conjunto vazio, $\vartheta(\emptyset)$, ao valor $-\infty$: a complexidade de procurar num conjunto vazio deve ser sempre $0 = 2^{-\infty}$.

Quando se escreve $\vartheta(X \overset{\psi}{\hookrightarrow} Y) - \vartheta(Y)$ estamos a caracterizar a complexidade de, dado um específico $y \in Y$, discriminar o $x \in X$ particular cujo criptograma é y.

EXEMPLO 35:
Sopunhamos que do conjunto dos textos X seleccionamos um subconjunto de X com apenas dois elementos; $\{x, x'\}$. Do conjunto de criptogramas Y, seleccionamos um só elemento y. Usando a incerteza de Hartley , tem-se $\vartheta(\{x, x'\}) = 1$ e $\vartheta(\{y\}) = 0$.

Ao determinar $\vartheta(\{x, x'\} \overset{\psi}{\hookrightarrow} \{y\})$, assume-se que são conhecidos os três valores x, x', y e procura-se seleccionar um de dois textos (x ou x') que tenha y como criptograma. Informalmente, $\vartheta(\{x, x'\} \overset{\psi}{\hookrightarrow} \{y\})$ mede a incerteza de seleccionar "aleatoriamente" um dos dois textos e de ser esse, de facto, o texto que originou o criptograma y.

Este exemplo motiva a análise de como se relaciona segurança perfeita com a capacidade de seleccionar, de entre dois textos, aquele que origina um criptograma arbitrário.

174 **Teorema**

A função ψ é ϑ, X, Y-perfeitamente segura se e só se, para todo $\{x, x'\}$, verifica-se

$$\vartheta(\{x, x'\} \overset{\psi}{\hookrightarrow} Y) - \vartheta(Y) \;\geq\; 1$$

Esboço de prova

Se for $\{x, x'\}$ um conjunto formado por dois textos distintos em X, tem-se

$$\{x, x'\} \overset{\psi}{\hookrightarrow} Y \quad \sim \quad \mathrm{rng}(\psi_x) \cap Y \;\uplus\; \mathrm{rng}(\psi_{x'}) \cap Y$$

Se suposermos que ψ é X, Y-perfeitamente segura, então o facto 172 garante-nos tanto $\mathrm{rng}(\psi_x)$ como $\mathrm{rng}(\psi_x)$ contêm Y; portanto $\{x, x'\} \overset{\psi}{\hookrightarrow} Y \;\sim\; Y \uplus Y$ e, por isso, $\vartheta(\{x, x'\} \overset{\psi}{\hookrightarrow} Y) - \vartheta(Y) \geq 1$.

Inversamente vamos supor que ψ não era ϑ, X, Y-perfeitamente seguro. Pelo lema 171 isto significa que existe pelo menos um $x \in X$ para o qual $\mathrm{rng}(\psi_x)$ é um subconjunto próprio de Y; ou seja existe um y tal que $\forall k \cdot \psi(k\|x) \neq y$. Escolha-se um outro x' tal que $\mathrm{rng}(\psi_{x'}) \supseteq Y$. Então, para este par de textos $\{x, x'\}$ tem-se $[\![\{x, x'\} \overset{\psi}{\hookrightarrow} Y]\!] < 2\,[\![Y]\!]$ e, consequentemente, $\vartheta(\{x, x'\} \overset{\psi}{\hookrightarrow} Y) - \vartheta(Y) < 1$.

Segurança Orientada à Aleatoriedade

A "perfeita segurança" pode não ser a forma perfeita de definir segurança.

Suponhamos, por exemplo, que ψ era tal que, para um valor x particular, se verificava $\psi(k\|x) \simeq k$. Pode acontecer esta condição se verifique mesmo que ψ seja "perfeitamente segura"; veja-se o *"one-time pad"* onde o texto $x = 0$ faz com que o criptograma $y = k \oplus x$ reproduza a chave. A intuição diz-nos que esta cifra não é segura se a mesma chave k for usada mais do que uma vez[43].

Por isso, uma noção de segurança que esteja de acordo com esta intuição, tem de considerar a possibilidade de um atacante escolher uma sequência de textos $\{x_n\}$ e observar a sequência de criptogramas correspondente $\{y_n\}$.

[43] Daí o qualificativo "one-time".

5.Criptografia de Chave Pública

5.1 Funções "one-way" e Sistemas "trapdoor"

A segurança de muitas técnicas criptográficas depende, crucialmente, da existência de certas funções matemáticas $f : X \to Y$ invertíveis que têm uma implementação computacionalmente eficiente no sentido "directo" mas cuja inversa é computacionalmente intratável.

São funções paras as quais a **computação directa**

$$(dado\ qualquer\ x \in X,\ determinar\ y \in Y\ \ tal\ que\ \ f(x) = y\ )$$

é computacionalmente simples, enquanto que a **computação inversa**

$$(dado\ qualquer\ y \in Y,\ determinar\ x \in X\ \ tal\ que\ \ f(x) = y\ )$$

é computacionalmente intratável com os recursos usuais.

Estas funções designam-se por **funções *one-way*** ou **funções unidireccionais**.

EXEMPLO 36: Embora não exista nenhuma prova formal de que existam funções unidireccionais é credível, com a experiência existente, que realmente existam tais funções.

As inversas dessas funções possuem implementações que, quase sempre, têm **complexidade sub-exponencial**. Recordemos que para descrever esse tipo de complexidade se usava a notação

$$L_n[p, c] \ = \ O(2^{c\, n^p\, (\log_2 n)^{1-p}})$$

para $p \in [0,1]$ e $c > 0$.

São candidatos a funções unidireccionais os seguintes exemplos:

Multiplicação/Factorização

A **multiplicação** de inteiros (*dados* $x, y \in \mathbb{N}$ *calcular* $z \in \mathbb{N}$ *tal que* $z = x \cdot y$) é implementável de forma eficiente mesmo para valores muito grandes de x e y. A complexidade de uma implementação comum da multiplicação é $O(n^2)$ (n o número de *bits* dos argumentos).

Quando x e y são números primos, a multiplicação tem um problema inverso: *dado* z *determinar* x *e* y *tais que* $z = x \cdot y$. Este problema chama-se **factorização** de z e os valores x e y chama-se **factores primos** de z.

Ao contrário da multiplicação, não é conhecido actualmente nenhuma implementação da factorização que possa ser considerada computacionalmente tratável com os recursos usuais. Os melhores algoritmos são sub-exponenciais: o melhor algoritmo genérico conhecido tem complexidade $L_n[1/3, c]$ com $c = (64/9)^{1/3}$ apesar de os que mais são usados terem complexidade ligeiramente superior $L_n[1/2, 1]$

Exponencial/Logaritmo Discreto

Dados um primo p grande e $0 < a < p$, a função $\exp_{p,a} : \mathbb{Z}_{p-1} \rightarrow \mathbb{Z}_p^*$ [44]

$$\exp_{p,a} : x \mapsto a^x \pmod{p}$$

é implementável de forma eficiente; de facto pode-se implementar com um algoritmo de complexidade linear ou melhor.

Prova-se também que, escolhendo uma base a apropriada, a função é um isomorfismo entre $\mathbb{Z}_{p-1}$ e $\mathbb{Z}_p^*$. Porém o problema inverso (dado y encontrar x tal que $y = a^x \mod p$) é, quanto muito, sub-exponencial. O melhor algoritmo conhecido está relacionado com o melhor algoritmo para resolver o problema da factorização e tem também complexidade $L_n[1/3, c]$ com $c = (64/9)^{1/3}$.

Raiz quadrada modular

Dado um $n = p \cdot q$, em que p, q são primos grandes, o problema é

dado um qualquer $x < n$ *determinar, se existir, um* y *tal que*

$$x = y^2 \pmod{m}$$

Prova-se que este problema é redutível ao problema da factorização de n e, por isso, a sua complexidade é idêntica à da factorização.

[44] $\mathbb{Z}_n$ designa o corpo finito definido pelos inteiros no intervalo $[0, n-1]$ equipados com as operações de soma e multiplicação. $\mathbb{Z}_p^*$ – com p primo – designa o grupo multiplicativo dos inteiros no intervalo $[1, p-1]$.

Estes exemplos apontam para funções inteiras $f\colon \mathbb{N} \longrightarrow \mathbb{N}$ mas, como iremos ver, é conveniente considerar funcionais em primeiro lugar. A formalização da noção de unidireccionalidade de uma funional $h\colon \mathbb{N} \longrightarrow \wp(\mathbb{N})$ traduz, essencialmente, a incapacidade de extrair do resultado $A = h(x)$ qualquer tipo de informação sobre o argumento x.

Outra forma de exprimir esta ideia consiste em impor, como condição de unidireccionalidade, a exigência de que os eventuais argumentos x que conduzam ao resultado $h(x)$, se "distribuam aleatoriamente".

175 Definição
Dada uma funcional $h\colon \varpi \longrightarrow 2^{\varpi}$ *e* $A \in 2^{\varpi}$, *o* **rastro** *é o conjunto*

$$A^h = \{\, x \in \varpi \mid h(x) \supseteq A \,\}$$

Os "eventuais argumentos" x que produzem $A \subseteq h(x)$ são descritos pelo rastro A^h que sejam, de alguma forma, algoritmicamente aleatórios. Isto implica que é necessário especificar as computações que se consideram efectivamente computáveis e que (quer através da abordagem de Kolmogorov ou de Martin-Löf) determinam a noção de aleatoriedade algorítimica.

Outro tipo de especificação incide sobre a família U dos "resultados" u cujos rastros se pretende testar.

Usamos U para escolher o tipo de unidireccionalidade que é conveniente para cada aplicação específica do conceito.

Por exemplo, pode-se tomar uma colecção U só formada por u que sejam aleatórios. Então se h for aleatório estará garantida a unidireccionalidade em U.

Outras vezes toma-se U como um conjunto de conjuntos singulares $\{y\}$ quando y percorre um qualquer domínio enumerável. Neste caso a unidireccionalidade procura que, para qualquer y nesse domínio, o rastro $\{y\}^h$ seja aleatório.

176 Definição
A funcional h é **unidireccional** *se é efectivamente computável e, para todo A suficientemente aleatório, é aleatório o rastro A^h.*

□

Vamos agora caracterizar a unidireccionalidade das funções.

177 Definição
A função $f : \mathbb{N} \to \mathbb{N}$ é unidireccional quando é computável e, para todo $y \in \mathbb{N}$ e toda a funcional efectivamente computável F, o rastro $F(y^f)$ é pelo menos tão aleatório quanto $F(y)$.

A unidireccionalidade é uma propriedade específica dessa função e traduz, essencialmente, o facto de dado um qualquer eventual resultado y não é computacionalmente tratável encontrar um x (mesmo que existam varios) tal que $f(x) \simeq y$.

Outro tipo de problema ocorre quando se tem uma colecção destas funções $\mathcal{F} = \{f\}$ e, fixando um qualquer par de valores (x, y), procura-se encontrar um $f \in \mathcal{F}$ tal que $f(x) \simeq y$. Isto requer a noção de *sistema de chaves*

178 DEFINIÇÃO
Um ***sistema de chaves*** *é uma classe efectivamente enumerável* $K = \{k\}$ *de funções unidireccionais* $k \colon \mathbb{N} \longrightarrow \mathbb{N}$ *de tal forma que, para cada* $v \in U$, *a funcional* $k \mapsto k^{-1}(v)$ *é unidireccional.*

A funcional $k \mapsto k^{-1}(v)$ associa a cada $u \in U$ o rastro $\{\, k \in K \mid k(u) \subseteq v \,\}$. Portanto, afirmar que esta funcional é unidireccional é equivalente a exigir que, para todo $u, v \in U$, a string

$$\{\, k \in K \mid k(u) \subseteq v \,\}$$

seja aleatória[45].

Particularizando para os casos em que u, v são conjuntos singulares, com $u = \{x\}$ e $v = \{y\}$, isto significa que, para x, y arbitrários, não é possível obter informação sobre qual é a chave k que verifica $k(x) = y$.

EXEMPLO 37(FUNÇÕES HASH):
A noção de unidireccionalidade permite caracterizar formalmente as propriedades de segurança aliadas às funções de "hash". Para representar uma função de hash arbitrária $H \colon \mathbb{N} \longrightarrow \mathbb{B}^t$ usaremos duas funcionais:

[45] Esta condição exprime exprime aquilo que vulgarmente se designa por "não-invertibilidade das chaves".

1. A funcional h mapeia $x \in \mathbb{N}$ no conjunto singular $\{H(x)\}$.
2. Assume-se uma codificação sobrejectiva de pares de naturais em naturais $(x, y) \mapsto x\|y$. A funcional g define-se como

$$g(x\|y) \;=\; \begin{cases} \emptyset & \text{se} \quad H(x) = H(y) \\ \{0\} & \text{se} \quad H(x) \neq H(y) \end{cases}$$

Com estas duas funcionais as condições de segurança da função de hash H exprimem-se da seguinte forma

unidireccionalidade	$\Rightarrow$	para todo resultado $z \leq 2^t$, é aleatório o rastro $\{z\}^h$
inexistência do 2º representado	$\Rightarrow$	para todo texto $x \in \mathbb{N}$ é aleatório o rastro $h(x)^h$
inexistência de colisões	$\Rightarrow$	$\emptyset^g$ é aleatório.

Sistemas "trapdoor"

179 DEFINIÇÃO

Seja K um sistema de chaves e h uma qualquer função de hash. Um sistema de chaves T é **sistema trapdoor** *de K para h, se existe uma* **funcional geradora** $\mathcal{G}: T \to 2^K$ *que é sobrejectiva, unidireccional e, para cada $t \in T$ e $k \in \mathcal{G}(t)$, verifica $t(k(x)) \simeq h(x)$ para todo x.*

A condição "$\mathcal{G}$ é sobrejectiva" implica que a cada chave $k \in K$ corresponde pelo menos uma função t (designada por "trapdoor" de k) que verifica a relação $t \circ k = h$. O facto de se exigir que $\mathcal{G}$ seja unidireccional implica que é intratável determinar qual é esse t a partir do conhecimento de k.

Pode acontecer que cada $\mathcal{G}(t)$ é singular; isto é, existe um única chave $k \in K$ que verifica $t \circ k \simeq h$. Neste caso $\mathcal{G}$ é uma **função geradora** do sistema "trapdoor".

Porém é mais frequente que $\mathcal{G}(t)$ seja um conjunto da forma $\{ G_p(t) \mid p \in \varpi \}$ em que os G_p formam uma sequência efectivamente computável de funções unidireccionais. O tamanho $|p|$ designa-se por **parâmetro de segurança**.

Neste caso, escolher $k \in \mathcal{G}(t)$ equivale a escolher um índice p e computar $k = G_p(t)$.

□

A aplicação dos sistemas "trapdoor" reside essencialmente nos sistemas de criptografia de chave pública como veremos adiante.

Um dos desenvolvimentos mais importantes em Criptografia foi a criação de técnicas criptográficas que usam dois tipos de chaves: chaves ditas "privadas", cuja segurança assenta do facto de serem ou não conhecidas pelos agentes apropriados, e chaves ditas "privadas", cuja segurança assenta no facto de terem sido emitidas pelas autoridades apropriadas.

Essas técnicas tomam o nome genérico de **técnicas assimétricas** (devido à assimetria nas propriedades das chaves) mas são designadas também por **técnicas de chave pública**.

As técnicas assimétricas tornaram-se possíveis a partir do momento em que foram identificadas certas funções que, não só parecem ser unidireccionais[46], como um comportamento assimétrico que conduz à assimetria de chaves.

A forma como vamos formalizar as técnicas de chave pública assenta precisamente no uso de sistemas "trapdoor" $\langle \mathcal{G}, K, T, h \rangle$. Nestas técnicas o sistema trapdoor é público; portanto são públicamente conhecidas a função de "hash" h, os sistemas de chaves T e K e a funcional geradora $\mathcal{G}$.

Uma instância particular da técnica torna público um $k \in \mathcal{G}(t)$ mas mantém privado valor de t.

Antes de vermos exemplos destas técnicas, convém examinar algumas situações concretas cujo comportamento computacional é, pelo menos, semelhante ao de um sistema trapdoor teórico. Como já foi referido não está provado que existe alguma função que seja unidireccional; existe apenas famílias de funções que mostram "promessa" de vir a ser unidireccionais.

Grupos cíclicos

Um dos exemplos de estruturas algébricas que aparentemente conduzem a sistemas *trapdoor* assenta nas propriedades dos grupos cíclicos.

Como exemplo vamos considerar grupos cíclicos da forma $\mathbb{Z}_p^*$.

[46] Não existe prova de que realmente existam funções unidireccionais; existe apenas uma boa possibilidade de que certas classes de funções exibam esse comportamento.

Seja p um primo representado com, pelo menos, 512 *bits* e que verifica certas condições de segurança que analisaremos num dos capítulos seguintes.

Nestas circunstâncias, existe sempre um elemento $g \in \mathbb{Z}_p^*$ tal que, quando x percorre os inteiros $\mathbb{Z}_{p-1}$, os elementos $g^x \pmod p$ percorrem todos os inteiros $\mathbb{Z}_p^*$; tais g chamam-se *elementos primitivos* ou *geradores primitivos*.

Assim, a candidata a função de hash é, neste caso,

$$h : x \mapsto g^x \pmod p \tag{117}$$

Acredita-se que a função h seja unidireccional; isto é, é implementável mas tem uma inversa (conhecida por *logaritmo discreto*) que é supostamente intratável. Adicionalmente deve ser livre de colisões.

Os diferentes h vão ser os índices da funcional geradora.

O sistema de chaves K é indexado pelos $\beta \in \mathbb{Z}_p^*$. O sistema trapdoor T é indexado pelos $a \in \mathbb{Z}_{p-1}$ tais que $h(a)$ é também um elemento primitivo. As respectivas chaves são

$$k_\beta : x \mapsto \beta^x \pmod p \tag{118}$$

$$t_a : x \mapsto x^a \pmod p \tag{119}$$

A funcional geradora $\mathcal{G}$ é determinada pela família de funções G_h com

$$G_h\colon\ a \quad \mapsto \quad h(a^{-1}) \tag{120}$$

Facilmente se verifica que a classe $K = \{k_\beta\}$ é um sistema de chaves e, para o *hash* h, a classe $T = \{t_a\}$ é a trapdoor correspondente a k_a. De facto:

(i) Enumerando T pelo índice a, cada G_h mapeia o índice a em T no índice β da chave k_β de tal forma que $\beta = G_h(a)$ se e só se $t_a(k_\beta(x) = h(x)$ para todo x.

De facto, sempre $(\mathrm{mod}\ p)$, fazendo $\beta = h(a^{-1}) = g^{a^{-1}}$, tem-se $k_\beta(x) = g^{x\,a^{-1}}$. Donde

$$t_a(k_\beta(x)) \ = \ \left(g^{x\,a^{-1}}\right)^a \ = \ g^x \ = \ h(x)$$

(ii) A função $h : x \mapsto g^x \ (\mathrm{mod}\ p)$ é unidireccional mas, em geral, não é uma função de *hash* porque apresenta colisões. De facto, dado um qualquer x é sempre possível encontrar $x' \neq x$ que tem a mesma imagem: basta escolher um x' que verifique $x' = x \ (\mathrm{mod}\ p)$.

No entanto, se restringirmos o domínio da função a $\mathbb{Z}_{p-1}$, a função torna-se injectiva e, por isso, não pode apresentar colisões.

(iii) Cada $k_\beta : x \mapsto \beta^x \ (\mathrm{mod}\ p)$ é unidireccional; é, de facto, injectiva e a intratabilidade do problema do logaritmo discreto está na base desta afirmação.

(iv) A colecção $K = \{k_\beta\}$ "forma" um sistema de chaves porque, dado quaisquer x, y não é possível descobrir qual é o valor de β que verifica $y = \beta^x \pmod{p}$. Mais uma vez a intratabilidade do logaritmo discreto justifica esta afirmação.

(v) É intratável, dado k_β (ou, equivalentemente, dado β) determinar t_a (ou, equivalentemente, determinar a) tal que $\beta = h(a^{-1})$; tal implicaria calcular o logaritmo discreto de β.

Sistemas RSA

Se $p > q$ são dois primos grandes e sejam $n \doteq p\,q$ e $\phi = (p-1)\,(q-1)$. Determinar ϕ conhecendo n é equivalente a conhecer-se a factorização de n que, supostamente, é intratável.

Um resultado muito importante é o chamado *teorema RSA* que, para todo $a, b \in \mathbb{Z}_n$ e todo $x \in \mathbb{Z}_m$, afirma

$$a = b \pmod{\phi} \implies x^a = x^b \pmod{n}$$

Neste contexto, escolhe-se uma função de *hash* arbitrária h e define-se um sistema *trapdoor* da seguinte forma:

Funcional geradora A funcional geradora $\mathcal{G}$ é definida pela família de funções $G_\phi : \mathbb{Z}_\phi^* \to \mathbb{Z}_\phi^*$ em que

$$G_\phi(a) = a^{-1} \pmod{\phi}$$

Chaves A família $K = \{k_r\}$, com $r \in \mathbb{Z}_\phi^*$, definida por

$$k_r : x \mapsto h(x)^r \pmod{n}$$

Trapdoors A família $T = \{t_a\}$, com $a \in \mathbb{Z}_\phi^*$, definida por

$$t_a : x \mapsto x^a \pmod{n}$$

A família $K = \{k_r\}$ é (a menos de um eventual estratégia para factorizar n) um sistema de chaves porque:

(i) Todos os k_r são unidireccionais (não só devido à função de hash $h(\cdot)$ mas também a exponenciação $x \mapsto x^r$).

(ii) As funções são indistinguíveis no sentido visto no caso anterior. Isto é, dados quaisquer x, y não é computacionalmente tratável determinar o r tal que $y = h(x)^r \pmod{n}$.

Por outro lado a colecção $T = \{t_a\}$ é também um sistema de chaves porque

(i) Cada $t_a : x \mapsto x^a \pmod{n}$ é supostamente unidireccional.

(ii) Os vários t_a são indistinguíveis, no sentido em que dados quaisquer x, y não é tratável determinar a tal que $y = x^a \pmod{n}$.

Finalmente, para mostrar que T é um sistema trapdoor de K para h temos de verificar

(i) Verifica-se a relação $t \circ \mathcal{G}(t) = h$ para todo t. Cada $a \in \mathbb{Z}_\phi^*$ determina um $t_a \in T$ e o índice $r = \mathcal{G}(a) = a^{-1} \pmod{\phi}$ (ou, equivalentemente, $r\,a = 1 \pmod{\phi}$) determina o correspondente k_r. Portanto

$$t_a\left(k_r(x)\right) \;=\; \left(h(x)^r\right)^a \pmod{n} \;=\; h(x)^{r\,a} \pmod{n} \;=\; h(x)$$

(ii) É intratável determinar t_a (ou a) a partir do conhecimento de k_r (ou r) a menos que ϕ seja conhecido.

Nota muito importante

Existe uma diferença fundamental entre este dois exemplos.

No primeiro caso (problemas de logaritmo discreto) a função geradora G_h era, como exige a definição teórica, unidireccional. Por isso não compromete a segurança do sistema se for publicamente conhecida.

No segundo exemplo a função geradora G_ϕ não é unidireccional. A não invertibilidade da chave k_r para a respectiva trapdoor t_a só é possível se o valor de ϕ for mantido secreto.

Por isso, nas técnicas RSA torna-se público apenas o parâmetro de segurança $|\phi|$ mas nunca ϕ.

5.2 Técnicas Criptográficas Básicas

Determinadas técnicas criptográficas podem ser definidas directamente a partir da noção de *sistema trapdoor* abstraindo completamente em relação aos detalhes das funções usadas.

No que se segue vamos assumir uma sistema de chaves K, uma função de hash h e um sistema de trapdoors T de K para h.

Cifra assimétrica

É possível definir uma classe muito geral de cifras assimétricas (isto é, cifras que usam chaves distintas para cifrar e decifrar) de forma bastante eficiente.

Neste tipo de cifras, o criptograma de um texto $x \in \mathbb{B}^*$ é um par $y \,||\, \sigma$ em que y é a *imagem* do texto e tem o mesmo comprimento que o texto, enquanto que σ se designa-se por *redundância*.

Todas estas técnicas têm uma fase de "setup" onde são geradas as chaves privadas t e públicas k. A segurança da técnica obriga a que, a partir do conhecimento de k, não seja computacionalmente tratável determinar t.

A técnica tem, depois, os processos ("esquemas") que são executados pelos dois agentes intervenientes: o agente que "cifra" a mensagem e o agente que "decifra" a mensagem.

Protocolo 1 :

Setup

Objectivo: definir o par de chaves (k, t) , usado nos esquemas de cifrar e decifrar,

(g.1) Gerar um t aleatório, gerar um $k \in \mathcal{G}(t)$ aleatório e publicita-o.

Cifrar

Objectivo: construir o criptograma $y \| \sigma$, conhecidos o texto x e a chave k

(c.1) gerar uma string aleatória ω e calcular $\sigma \leftarrow k(\omega)$ e $\lambda \leftarrow h(\omega)$
(c.2) calcular $y \leftarrow x \oplus \lambda$, calcular e publicitar o criptograma $z \leftarrow y \| \sigma$

Decifrar

Objectivo: reconstruir o texto limpo x conhecidos a trapdoor t e o criptograma z

(d.1) recuperar as componentes individuais do criptograma $(y, \sigma) \leftarrow z$
(d.2) recuperar λ calculando $\lambda' \leftarrow t(\sigma)$.
(d.3) recuperar x calculando $x' \leftarrow y \oplus \lambda'$.

Note-se que:

(i) O passo (g.2) recupera a chave de sessão λ. De facto tem-se $\lambda = h(\omega)$, $\lambda' = t(\sigma)$ e $\sigma = k(\omega)$. Logo, usando a identidade $t \circ k \simeq h$, temos

$$\lambda' = t(\sigma) = t(k(\omega)) = h(\omega) = \lambda$$

(ii) Como $\lambda = \lambda'$ o passo (g.3) recupera x; de facto tem-se $y = x \oplus \lambda$ e $x' = y \oplus \lambda'$; logo

$$x' = y \oplus \lambda' = x \oplus \lambda \oplus \lambda' = x$$

(iii) Se a incerteza em λ for igual à incerteza em x, a cifra é equivalente à cifra de Vernam ("one-time pad") e, por isso, é perfeitamente segura.

A incerteza em λ depende directamente apenas de dois factores: a incerteza em ω e as propriedades da função de "hash" h.

A incerteza em ω depende da forma como é gerado em (c.1) mas também da unidireccionalidade de k. Se k não for unireccional, partindo do conhecimento de σ (que é público) reduz-se a incerteza em ω.

A função h pode "perder incerteza" se a incerteza no seu contradomínio for inferior à incerteza no domínio.

Assinatura de 1 *bit*

É possível definir uma classe muito geral de assinaturas digitais sobre mensagens de tamanho 1 *bit*[47] em que

o agente A tem intensão de enviar 1 bit de informação para o agente B provando a autoria da mensagem (A é o único que a pode ter enviado) e a sua integridade (o bit não é alterado).

Ao contrário dos esquemas simples de assinaturas, vamos definir um **protocolo** de 3 passos onde *B* manifesta previamente a sua disposição para receber mensagens de *A*.

Protocolo 2 :

Geração: um *TA* procede como no protocolo 1, distribui t a *A* e torna público k

Inicialização *B* gera duas strings aleatórias ω_0 e ω_1; em seguida calcula e torna públicos $\sigma_0 \leftarrow k(\omega_0)$ e $\sigma_1 \leftarrow k(\omega_1)$.

Assinatura *A* calcula $s \leftarrow t(\sigma_b)$ e torna público o par $\langle b, s\rangle$.

Verificação *B* aceita a mensagem b sse for válido $(s = h(\omega_b))$.

[47] Podem não parecer muito úteis(!) mas é possível extender este mecanismo.

Identificação

Um protocolo de identificação de um agente *A* perante um agente *B* é

uma prova apresentada a B de que A conhece um segredo s sem que s possa vir a ser, em qualquer momento, conhecido por B.

Neste protocolo a prova é representada por uma chave pública k e o segredo representado pelo *trapdoor* t correspondente.

Protocolo 3 :

Inicialização *TA* gera, como no protocolo 1, chaves k e t e distribui t a *A*.

Intensão *A* manifesta a intensão de ser identificado publicando k

Desafio *B* gera um desafio σ no seguinte esquema:

(1) gera uma string aleatória ω ,

(2) calcula $\sigma \leftarrow k(\omega)$ e publicita-o.

Resposta *A* gera a resposta adequada ao desafio calculando $r \leftarrow t(\sigma)$ e publicando este valor.

Verificação *B* aceita a identificação sse $r = h(\omega)$.

Acordo de chaves com autenticação mútua

Dois agentes A e B, que possuem (cada um) um segredo na forma de trapdoors t_A *e* t_B*, sendo públicas as respectivas chaves* k_A *e* k_B*, acordam num segredo comum* λ *e, simultaneamente, certificam-se que estão a comunicar com o interlocutor legítimo.*

O protocolo tem duas partes: primeiro cada um dos participantes gera uma chave, λ_A e λ_B que, em princípio, coincidem. Na segunda parte verifica-se que as duas chaves coincidem e que cada agente está a interagir com o agente certo.

1ª Parte - Geração da Chave

Protocolo 4 :

segredo e desafio *A* gera o segredo comum λ com o esquema:

(1) gera ω aleatório e calcula $\lambda_A \leftarrow h(\omega)$

(2) calcula e publica $\sigma_1 \leftarrow k_B(\omega)$.

assume λ_A *como chave acordada*

resposta e desafio *B* calcula $\lambda_B \leftarrow t_B(\sigma_1)$

assume λ_B *como chave acordada*

▷ se a execução for correcta $\lambda_B = t_B(k_B(\omega)) = h(\omega) = \lambda_A$

2ª Parte - Autenticação da Chave

Protocolo 5 :

Desafio *B* calcula $\sigma_2 \leftarrow k_A(\lambda_B)$ e publicita-o.

Resposta *A* reconhece o interlocutor e responde ao desafio seguindo o esquema

(1) calcula $\lambda_1 \leftarrow t_A(\sigma_2)$

(2) *aceita prosseguir se* $\lambda_1 = h(\lambda_A)$

▷ se o protocolo for bem executado será $\lambda_1 = t_A(k_A(\lambda_B)) = h(\lambda_B) = h(\lambda_A)$

(3) calcula $\sigma_3 \leftarrow k_B(\lambda_1)$ e publicita-o.

Verificação *B* reconhece o interlocutor usando o esquema

(1) calcula $\lambda_2 \leftarrow t_B(\sigma_3)$

(2) *aceita prosseguir se* $\lambda_2 = h(h(\lambda_B))$

▷ se o protocolo for bem executado, será $\lambda_2 = t_B(k_B(\lambda_1)) = h(\lambda_1) = h(h(\lambda_B))$.

(3) *Termina com sucesso*

5.3 Grupos Diffie-Hellman

Em 1976, foi publicada pela primeira vez uma técnica criptográfica de chave pública: o protocolo de acordo de chaves que passou a ser designado por Diffie-Hellman.

Resumidamente o protocolo é formado por uma sequência de mensagens entre dois agentes, *A* e *B*, usando um canal aberto e com o objectivo de ambos partilharem um segredo λ.

Previamente os agentes concordam um primo p e num inteiro $g > 1$ cuja ordem[48] seja um primo r. Estas escolhas fixam um grupo multiplicativo $\mathbb{G}_g = \langle G, *, 1 \rangle$, de ordem r, em que o suporte G é o conjunto dos inteiros da forma $(g^n \mod p)$, com $n \in \mathbb{Z}_r$, e a operação de grupo $*$ é multiplicação módulo p.

Protocolo 6 : Diffie-Hellman

1. *A* gera um segredo aleatório $a \in \mathbb{Z}_r^*$, calcula g^a e envia este valor para *B*.
2. *B* gera um segredo aleatório $b \in \mathbb{Z}_r^*$, calcula g^b e envia este valor para *A*.
3. *A* calcula $(g^b)^a$; *B* calcula $(g^a)^b$; os valores coincidem e definem λ.

[48] A ordem de um qualquer $g \neq 0$ no grupo multiplicativo $\mathbb{Z}_p^*$ é o menor inteiro $n > 0$ tal que $(g^n = 1 \mod p)$. Todas as possíveis ordens de eventuais g são divisores de $(p-1)$.

A *correcção* do protocolo assenta nas propriedades algébricas dos sub-grupos cíclicos de $\mathbb{Z}_p^*$. O gerador g determina um destes sub-grupos, nomeadamente o grupo $\mathbb{G}_g$.

A *segurança* assenta na eventual intratabilidade do problema do logaritmo discreto ("discret-log problem" ou DLP) no grupo $\mathbb{G}_g$: se, no protocolo DH, for intratável recuperar a ou b a partir das mensagens g^a ou g^b, um atacante não consegue reconstituir o segredo λ.

Desde esse trabalho pioneiro, muitas mais técnicas criptográficas, com objectivos de segurança diversos, foram desenvolvidas assentes na mesma base: a correcção justificada pelas propriedades dos grupos cíclicos de inteiros e a segurança justificada pela intratabilidade do DLP nesses grupos.

No entanto cedo se notou que uma técnica criptográfica, baseada num determinado grupo cíclico, pode ser transferida para qualquer outro grupo cíclico desde que as condições de segurança se mantenham; isto é, o DLP continue intratável nesse grupo.

Sendo assim faz sentido considerar grupos multiplicativos arbitrários $\mathbb{G} = \langle G, *, 1 \rangle$. Note-se que pode ser importante considerar grupos infinitos.

Neste grupo os *elementos de torção* são aqueles $g \in \mathbb{G}$ para os quais existe um $n > 0$ tal que $g^n = 1$. O menor de todos estes n chama-se **ordem** de g. Os elementos de $\mathbb{G}$ da forma g^n formam um sub-grupo cíclico, cuja ordem é a ordem de g, e que se chama **órbita** de g.

Num tal grupo, o problema do logaritmo discreto, representado por $\mathrm{DLP}(\mathbb{G})$, define-se como

Problema do Logaritmo Discreto (DLP)
Conhecidos um elemento de torção $g \in \mathbb{G}$, a sua ordem r e um elemento g^a da sua órbita, determinar o valor de a.

Sabe-se que a complexidade computacional média do DLP num grupo cíclico é determinada pela dimensão do maior factor primo da ordem do grupo. Por isso, em tecnicas criptográficas assentes nos grupo cíclicos, *só interessa usar grupos cuja ordem seja um número primo*.

Se a ordem não for primo, um grupo com muitos elementos apenas introduz complexidade excessiva nas operações que devem ser eficientes sem que tal conduza a mais segurança para as operações supostamente intratáveis.

180 Definição
Os sub-grupos cíclicos de $\mathbb{G}$ cuja ordem é um primo, designam-se por **grupos primos** *em $\mathbb{G}$. A* **dimensão** *de $\mathbb{G}$, representada por $\|\mathbb{G}\|$, é $\lceil \log_2 r \rceil$, sendo r a ordem do maior grupo primo de $\mathbb{G}$.*

□

Simultaneamente, com o desenvolvimento de técnicas criptográficas com segurança assente no DLP, foi sendo claro que em certos grupos cíclicos era possível desenvolver outro tipo de técnicas. refinando as condições de segurança. Indo além da simples exigência de que o DLP seja intratável, introduziu-se uma gama mais detalhada de problemas

semelhantes ao DLP (mas que não coincidem necessariamente com o DLP) e introduziu-se a noção de "gap" para comparar a complexidade computacional destes vários problemas.

Isto permite explicitar formas mais complexas de segurança que, por seu lado, permitem definir técnicas com uma crescente gama de objectivos.

181 Definição

Genericamente, diz-se que existe um **gap** *do problema P para o problema P' quando P é redutível a P' (isto é, existe um algoritmo PPT que converte qualquer eventual solução de P' numa solução de P) mas P' não é redutível a P.*

Quando existe um algoritmo PPT que resolve o problema P mas não existe um algoritmo PPT que resolva o problema P', este "gap" pode ser explorado para definir alguma forma de "trapdoor".

Repare-se no problema que um eventual atacante ao protocolo Diffie-Hellman quer resolver: ele conhece o gerador g, conhece as mensagens g^a e g^b e quer descobrir $\lambda = (g^a)^b = (g^b)^a = g^{a\,b}$.

Este problema é tão importante que merece uma designação própria: chama-se

Problema da Computação Diffie-Hellman (CDHP)

Conhecidos um elemento de torção g, a sua ordem r (supostamente um primo), e elementos g^a e g^b da sua órbita, determinar $g^{a\,b}$.

Note-se que para atacar o protocolo DH só é necessário resolver o CDHP; o ataque pode não exigir que se tenha de resolver DLP.

Portanto, como se comparam os dois problemas, DLP e CDHP?

Claramente, em qualquer grupo $\mathbb{G}$, CDHP é redutível a DLP: se se souber calcular a e b a partir de g^a e g^b, sabe-se facilmente calcular $g^{a\,b}$ a partir destes dois valores.

Já não é evidente que o DLP não seja redutível ao CDHP nem é evidente que, sendo o DLP computacionalmente intratável, o CDHP continue a ser computacionalmente intratável. Ambas as implicações dependem do grupo $\mathbb{G}$ usado[49].

Por isso faz sentido destacar os grupos onde seja seguro usar o protocolo DH,

182 Definição
Seja $\mathbb{G}$ um grupo multiplicativo e $g \in \mathbb{G}$ um elemento de torsão cuja ordem r é um número primo. O triplo $\langle \mathbb{G}, g, r \rangle$ designa-se por ***Grupo Diffie-Hellman (GDH)*** *se não existe nenhum algoritmo polinomial em $|r|$ que resolva CDHP.*

Num GDH $\langle \mathbb{G}, g, r \rangle$ faz sentido definir os problemas DLP e CDHP como oráculos. Assim define-se

[49] Num grupo primo de ordem r, sabe-se que DLP e CDHP são equivalente quando existe um grupo auxiliar definido algebricamente sobre $\mathbb{Z}_r$ que tenha pequena dimensão. Os grupos auxiliares que têm sido mais testados são as curvas elípticas definidos sobre $\mathbb{Z}_r$.

Oráculo DLP Idealiza um algoritmo que recebe como *input* um $x \in G$ e produz como resultado o único $a \in \mathbb{Z}_r$ tal que $x = g^a$ ou então falha se x não está na órbita de g.

Oráculo CDHP Idealiza o algoritmo que recebe como *input* $\langle x, y \rangle \in G^2$ e falha se x ou y não pertencem à órbita de g; se for $x = g^a$ e $y = g^b$ o oráculo produz o resultado $z = g^{a\,b}$.

O problema da computação Diffie-Hellman pode-se generalizar para dois grupos DH com a mesma ordem r. Considere-se um segundo GDH $\langle \Gamma, \gamma, r \rangle$ de ordem r. Então, conhecidos ambos GDH´s,

Problema da Computação Diffie-Hellman Generalizada (co-CDHP)
Dado $g^a \in G$ *determinar* $\gamma^a \in \Gamma$

É óbvio que co-CDHP generaliza o CDHP: basta fazer $\Gamma = G$ e $\gamma = g^b$. É também óbvio que co-CDHP é redutível a DLP: consultando uma vez um oráculo DLP, dado $x \in G$ obtém-se o valor a que permite calcular γ^a.

O oráculo respectivo define-se do mesmo modo

Oráculo co-CDHP Idealiza o algoritmo que recebe $x \in G$, falha se x não está na órbita de g, e, caso seja $x = g^a$, produz γ^a.

□

Dado um GDH $\langle \mathbb{G}, g, r \rangle$ que técnicas criptográficas é possível definir?

Como primeira aplicação criptográfica de grupos DH temos a construção de um **sistema trapdoor** tal como foi sugerido na secção 1. Daí resultam um conjunto de técnicas básicas como foi visto nessa altura.

No entanto outras técnicas criptográficas usam esta estrutura básica dos grupos DH explorando as propriedades algébricas do grupo sem que possam ser expressas num enquadramento trapdoor genérico.

Dentro destas destacamos as técnicas criptográficas afins ao esquema de assinaturas digitais **DSA** e as técnicas criptográficas ditas **orientadas à identidade**.

Protocolo 7 : Componentes Comuns

Sejam

1. $\langle \mathbb{G}, g, r \rangle$ um grupo Diffie-Hellman; G denota a órbita de g.
2. $H : \mathbb{B}^* \longrightarrow \mathbb{Z}_r^*$ é uma função de hash
3. $f : G \longrightarrow \mathbb{Z}_r^*$ é uma função chamada *função de redução*.

As técnicas criptográficas baseadas na identidade, para além de grupos DH com características particulares, usam outro tipo de componentes.

Técnicas básicas trapdoor num grupo DH

A partir das componentes comuns define-se um *sistema trapdoor* com

- A *função de hash*: $h : \mathbb{B}^* \longrightarrow G$ define-se como

$$h \ : \ x \ \mapsto \ g^{H(x)}$$

- O *sistema de chaves públicas*: para cada $\beta \neq 1 \in G$ define-se $k_\beta : \mathbb{B}^* \longrightarrow G$ como

$$k_\beta \ : \ x \ \mapsto \ \beta^{H(x)}$$

- O *sistema de chaves privadas*: para cada $a \in \mathbb{Z}_r^*$, define-se $t_a : G \longrightarrow G$ como

$$t_a \ : \ z \ \mapsto \ z^a$$

- A *função geradora*: $\mathcal{G} : \mathbb{Z}_r^* \longrightarrow G$

$$\mathcal{G} \ : \ a \ \mapsto \ g^{a^{-1}}$$

Este sistema verifica claramente a relação

$$\beta = \mathcal{G}(a) \quad \Leftrightarrow \quad t_a(k_\beta(x)) \ = \ h(x) \qquad \forall a, x \in \mathbb{Z}_r^*$$

e, por isso, é um presumível sistema trapdoor.

Nestas circunstâncias todas as técnicas criptográficas básicas descritas na secção 2 podem ser usadas. A título ilustrativo uma cifra assimétrica seria

Protocolo 8 : ElGamal Generalizado

geração do par de chaves

Gerar $u \in \mathbb{B}^*$ aleatório; calcular $a \leftarrow H(u)$ e $\beta \leftarrow \mathcal{G}(a)$; a chave privada é t_a ; a chave pública é k_β .

cifrar $x \in \mathbb{Z}_r$

Gerar $\upsilon \in \mathbb{B}^*$ aleatório; calcular a redundância $\sigma \leftarrow k_\beta(\upsilon)$, a imagem $y \leftarrow x \oplus f(h(\upsilon))$ e o criptograma $z \leftarrow \sigma \,\|\, y$. Publicitar z.

decifrar $z = \sigma \,\|\, y$

Recuperar $(\sigma, y) \leftarrow z$. Recuperar $x' \leftarrow y \oplus f(t_a(\sigma))$.

Correcção As variáveis x e x' são indistinguíveis.

Segurança A família de funcionais $C_a \colon \mathbb{Z}_r \to \wp(\mathbb{Z}_r \times \mathbb{Z}_r)$, indexada por $a \in \mathbb{Z}_r^*$ e definida por $C_a(x) = \{\, \sigma \| y \mid \upsilon \in \mathbb{B}^* \,\}$, com $\sigma = k_{\mathcal{G}(a)}(\upsilon)$ e $y = x \oplus f(h(\upsilon))$, é um sistema de chaves.

A condição de segurança significa que todas as funcionais são unidireccionais e não é possível recuperar a mesmo que se conheça o texto claro x e o criptograma $\sigma \,\|\, y$.

5.4 O esquema de assinaturas DSA generalizado

Dentro das técnicas criptográficas mais importantes construídas com grupos DH estão, sem dúvida, os esquemas de assinaturas digitais que nasceram do **Digital Signature Algorithm (DSA)**.

O DSA foi proposto em 1991 pelo "National Institute of Standards and Tecnhology (NIST)" e foi aceite em 1994 como "standard" de assinatura digitais pelo organismo de standardização NIST. Conjuntamente com a função de hash SHA ("Standard Hash Algorithm")[50] constituem o "Digital Signature Standadrd" (DSS).

O DSA é uma técnica baseada nas propriedades algébricas do mais comum dos grupos cíclicos criptográficos: o grupo $\mathbb{Z}_p^*$. Com o aparecimento de algoritmos eficientes para lidar com curvas elípticas sobre corpos finitos, foi proposto em 1992 o **Elliptic Curve Digital Signature Algorithm (ECDSA)** como adaptação para o grupo cíclico daí resultante.

De momento o ECDSA é aceite como standard ISO (ISO 14888-3) desde 1998, como standard ANSI (ANSI X9.62) desde 1999 e, desde 2000, como standard comum IEEE e FIPS (IEEE 1363-2000 e FIPS 186-2).

A versão aqui apresentada é geral e usa um qualquer GDH e as componentes comuns apresentadas na página 331.

[50] Posteriormente substituído pelo SHA-1.

Protocolo 9 : DSA Generalizado

geração do par de chaves

1. Gerar $u \in \mathbb{B}^*$ aleatório e calcular $a \leftarrow H(u)$ e $\beta \leftarrow h(u)$.
2. A chave privada é a e a chave pública é β.

assinar a mensagem $m \in \mathbb{B}^*$

1. Gerar $\upsilon \in \mathbb{B}^*$ aleatório; calcular $\gamma \leftarrow h(\upsilon)$ e $\sigma \leftarrow f(\gamma)$.
2. Determinar s como solução da equação $s\,H(\upsilon) = a\,\sigma + H(m)$ em $\mathbb{Z}_r$.
3. Publicitar a assinatura $z \leftarrow \sigma \,\|\, s$

verificar a assinatura z para o texto m

1. Recuperar $(\sigma, s) \leftarrow z$; calcular $\lambda \leftarrow (\beta^{\sigma} \cdot h(m))^{s^{-1}}$
2. Aceitar a assinatura sse $\sigma = f(\lambda)$ se verifica em $\mathbb{Z}_r$,

As várias instâncias deste esquema genérico resultam de se fazer uma escolha apropriada do grupo cíclio $\mathbb{G}$, da função de hash H e da função de redução f. As duas instâncias mais importantes são o DSA original e o ECDSA que correspondem às seguintes escolhas:

DSA

O grupo cíclio é um grupo primo do grupo multiplicativo $\mathbb{Z}_p^*$ cuja ordem r divide $p-1$; o standard DSA exige que a dimensão de r seja $\geq$ 160 bits e a dimensão de p seja $\geq$ 512 bits.

A função de redução $f : G \longrightarrow \mathbb{Z}_r$ é $x \mapsto x \mod r$. A função de hash H é a função SHA-1 com os resuldados reduzidos módulo r.

ECDSA

Os elementos de um grupo cíclico definido por uma curva elíptica são (como veremos adiante) pontos $P = \langle x, y \rangle$ do plano em que as coordenadas x, y são elementos de um determinado corpo finito $\mathbb{F}_q$. Sobre estes pontos está definida uma operação de grupo que, neste caso, é representado de forma aditiva; isto é $P + Q$ é a aplicação da operação de grupo a dois pontos P e Q e existe um elemento neutro (um zero) representado por $\mathcal{O}$.

Como a operação de grupo é aditiva, a "exponenciação" é aqui chamada **multiplicação escalar** e representa-se por $n\,P$. A *ordem* de P é o menor n tal que $n\,P = \mathcal{O}$.

O ECDSA usa, como grupo cíclico G, a órbita de um ponto P de ordem prima r. A função de hash H é, como no DSA, a função SHA-1 com os resultados reduzidos módulo r.

A função redução f mapeia um ponto $P \in G$ de coordenadas $\langle x, y\rangle \in \mathbb{F}_q \times \mathbb{F}_q$ num inteiro $f(P) \in \mathbb{Z}_r$ que é a redução módulo r de uma representação inteira[51] da componente x do ponto P.

□

Sumariamente a correcção da assinatura deriva da seguinte cadeia de argumentos.

1. A equação

$$s\,H(v) = a\,\sigma + H(m) \tag{121}$$

verifica-se em $\mathbb{Z}_r$ sse $g^{s\,H(v)} = g^{a\,\sigma} \cdot g^{H(m)}$ se verifica em G.

2. Como $\beta = g^a$, $\gamma = g^{H(v)}$ e $h(m) = g^{H(m)}$, a equação (121) verifica-se sse $\gamma^s = \beta^\sigma \cdot h(m)$.
3. Como $\lambda = (\beta^\sigma \cdot h(m))^{s^{-1}} = (\gamma^s)^{s^{-1}} = \gamma$ e $\sigma = f(\gamma)$, a assinatura será aceite sse for $\sigma = f(\lambda)$.

Este argumento de **correcção** apenas prova que, caso a assinatura esteja bem construída, a verificação tem sucesso. Falta provar a implicação em sentido contrário: assume-se que a a verificação tem sucesso e quere-se provar que a assinatura foi bem construída.

[51] Se $\mathbb{F}_q$ for um corpo primo, a representação inteira de $x \in \mathbb{F}_q$ coincide com x; se $\mathbb{F}_q$ for um corpo binário, a representação inteira de x será o inteiro que tem a mesma representação em bits do que x.

A noção de grupo cíclico implica que uma equação da forma $g^x = g^y$ é válida em G se e só se $x = y$ for válida em $\mathbb{Z}_r$. Portanto temos a certeza que se verifica $\gamma^s = \beta^\sigma \cdot h(m)$ se e só se a equação (121) se verifica.

Porém a intervenção da função de redução f neste esquema, traz problemas.

Note-se que, se f não for injectiva, é possível ocorrer $f(\gamma) = f(\gamma')$ com $\gamma \neq \gamma'$. Portanto a verificação de $f(\gamma) = f(\lambda)$ não implica necessariamente que seja válido $\gamma^s = \beta^\sigma \cdot h(m)$.

Se facto, se percorrermps todos os possíveis $\gamma_1 \in f^{-1}(\sigma)$ e os respectivos b_1 tais que $\gamma_1 = g^{b_1}$, é possível que surjam vários tipos de situações:

1. existam mensagens m_1 cujo hash $H(m_1)$ verifique a equação $s\,b_1 = a\,\sigma + H(m_1)$; assim, surgem mensagens diferentes (m e m_1) com *hashs* diferentes que verificam a mesma assinatura s.
2. existam assinaturas $s_1 \neq s$ que verifiquem $s_1\,b_1 = s\,b$; neste caso, o mesmo $\lambda \leftarrow \beta^\sigma\,h(m)$, agora levantado a outro expoente ${s_1}^{-1}$, continuaria a verificar $\sigma = f\left(\lambda^{{s_1}^{-1}}\right)$; temos, então, duas assinaturas diferentes (s e s_1) que o esquema de verificação aceita para a mesma mensagem m.

Estas duas situações indicam-nos que, para um mesmo σ (e uma mesma geração aleatória υ) podem existir várias assinaturas para a mesma mensagem e várias mensagens com a mesma assinatura. Portanto os pares

mensagem+assinatura legitimamente gerados pelo esquema de assinatura não coincidem exactamente os pares análogos que são verificados pelo esquema de verificação.

No entanto as construções apresentadas indicam que não será probabilisticamente viável distinguir a duas situações; para fazer tal distinção não só seria necessário inverter a função de hash como resolver o problema do logaritmo discreto para, dado $\gamma \in f^{-1}(\sigma)$, encontrar o valor b tal que $\gamma = g^b$.

□

Para uma análise da **segurança**, e sob o ponto de vista de um atacante (aqui designado por **falsificador**), vamos assumir que ele:

(i) conhece o grupo cíclico $\langle \mathbb{G}, g, r \rangle$,
(ii) escolhe um assinante legítimo seleccionando a sua chave pública β, e
(iii) escolhe também uma uma família finita de textos $M = \{m_i\}$ e conhece assinaturas para cada um desses textos; isto é, conhece $S = \{\langle \sigma_i, s_i \rangle\}$ com $(m_i, \langle \sigma_i, s_i \rangle) \in \text{Ver}_\beta$.

Colocam-se-lhe dois tipos de desafios/ataques:

falsificação da mensagem
: Escolher um dos textos em M, digamos $m_i \in M$, e gerar um outro texto $m' \neq m_i$ que verifique a mesma assinatura $\langle \sigma_i, s_i \rangle$ com a mesma chave pública β.

falsificação da assinatura

Gerar um qualquer texto $m \notin M$ e uma assinatura $\langle \sigma, s \rangle$ que seja verificável com a chave pública do assinante legítimo.

Uma análise *ad-hoc* dos ataques começa por calcular a família $\Gamma = \{\gamma_i\}$ fazendo $\lambda_i \leftarrow \beta^{\sigma_i} h(m_i)$ e resolvendo $\gamma_i^{s_i} = \lambda_i$.

Para o primeiro ataque ("falsificação da mensagem"), a forma mais simples seria encontrar uma mensagem m' que verifique $H(m') = H(m_i)$ para algum dos $m_i \in M$. Isto equivale a encontrar colisões na função de *hash* H.

Assumindo que não existem tais colisões, a alternativa passa por encontrar um i e um $\gamma' \neq \gamma_i$ tais que $f(\gamma') = f(\gamma_i) = \sigma_i$. Então faz-se $h(m') \leftarrow \beta^{-\sigma_i} \left(\gamma'\right)^{s_i}$ e consegue-se localizar uma *hash* $h(m') \neq h(m_i)$ que é aceite pelo esquema de verificação.

Esta computação é realizável por um algoritmos PPT. Será possível, a partir de $h(m')$, determinar a desejada mensagem m' também por um algoritmo PPT ?

Sabendo que $h(m') = g^{H(m')}$, se fosse possível determinar m', seria também possível resolver o problema do logaritmo discreto para o valor $h(m')$: bastaria calcular $H(m')$. Se $h(m')$ for "suficientemente aleatório", essa solução para o PLD é, por hipótese, intratável.

Uma possível realização ao ataque de "falsificação de assinatura" parte, simplesmente, do conhecimento da chave privada a; se for possível determinar a a partir de $\beta = g^a$ (resolvendo o problema do logaritmo discreto) seria possível, obviamente, determinar qualquer assinatura para qualquer mensagem.

Uma questão bastante mais complexa é a de saber se é possível falsificar a assinatura de uma mensagem $m \notin M$ sem o conhecimento da chave privada a.

5.5 Transformação de Fiat-Samir

A transformação de Fiat-Shamir converte um qualquer protocolo de identificação "desafio-resposta" na forma canónica, num esquema de assinaturas.

O protocolo de identificação lida com dois agentes, normalmente designados por "prover" (representado por $\mathcal{P}$) e "verifier" (representado por $\mathcal{V}$). O seu objectivo é fazer com que o "prover" prove ao "verifier" que conhece um segredo s sem que, neste processo, o "verifier" fique com qualquer conhecimento sobre esse segredo. Nomeadamente o "verifier" só deve ser capaz, no fim do protocolo, de conhecer exactamente os mesmo items de informação que conheceria se não participasse no protocolo.

Neste e nos seguintes protocolos usaremos a seguinte convenção:

- A notação $A : x \leftarrow \phi$ representa um passo de protocolo em que o agente A fica a conhecer um item x a partir de uma computação ϕ.
- A notação $A : x \rightarrow \psi$ denota um passo de protocolo onde o agente A, a partir do conhecimento x, calcula um valor ψ e publicita-o. A notação $A : x \rightarrow y = \psi$ é semelhante mas atribui o nome y à informação calculada.
- O titular público é representado por $\cdot$ e o conhecimento vazio por $*$.

De uma forma esquemática o protocolo canónico "desafio-resposta" escreve-se

Protocolo 10 : Identificação Canónica

$\mathcal{P}: * \longrightarrow \text{id}$	$\mathcal{P}$ mostra **intensão** de ser identificado tornando público id
$\mathcal{V}: * \longrightarrow d$	$\mathcal{V}$ gera um **desafio** aleatório d com um grau de incerteza adequado
$\mathcal{P}: s, d \longrightarrow r$	$\mathcal{P}$ usa o segredo s e o desafio d para calcular uma **resposta** r
$\mathcal{V}: \rho(\text{id}, d, r) =? 1$	$\mathcal{V}$ **verifica**, recorrendo a uma decisão ρ, se é válido o triplo $\langle \text{id}, d, r \rangle$

Assim, uma instância específica deste protocolo necessita de precisar os três algoritmos e a decisão que nele intervêm. Nomeadamente:

1. O algoritmo PPT que, a partir da identidade de $\mathcal{P}$, gera um valor id representativo;
2. O gerador de desafios aleatórios d;
3. O algoritmo PPT que, sob *input* do segredo s e do desafio d, produz a resposta adequada r
4. A decisão ρ que reconhece os triplos $\langle \text{id}, d, r \rangle$ válidos.

A decisão ρ é um teste raro que diferencia duas linguagens: a linguagem dos triplos $\langle \text{id}, d, r \rangle$ válidos gerada fazendo d percorrer o domínio dos desafios aceitáveis, e a linguagem dos peseudo-triplos $\langle \text{id}, d, u \rangle$, quando u é aleatório no domínio das respostas aceitáveis.

EXEMPLO 38: Um exemplo paradigmático de um protocolo com esta estrutura assente em grupos Diffie-Hellman é o chamado **protocolo de identificação de Schnorr**.

Nesse protocolo vamos considerar os elementos comuns de um grupo DH (ver página 331) e ainda um gerador $\mathcal{U}_r$ de inteiros uniformemente distribuídos em $\mathbb{Z}_r$.

Protocolo 11 : Identificação de Schnorr

SetUp geração de chaves, privada s e pública $\beta = g^{-s}$

(1) $\mathcal{P}: s \leftarrow \mathcal{U}_r \quad ; \quad \mathcal{P}: s \rightarrow \beta = g^{-s}$

Instância o "prover" $\mathcal{P}$ e o "verifier" $\mathcal{V}$ executam os seguintes passos:

(1)	$\mathcal{P}: \upsilon \leftarrow \mathcal{U}_r \quad ; \quad \mathcal{P}: \upsilon \rightarrow \gamma = g^{\upsilon}$	*intensão*
(2)	$\mathcal{V}: d \leftarrow \mathcal{U}_r \quad ; \quad \mathcal{V}: d \rightarrow d$	*desafio*
(3)	$\mathcal{P}: \upsilon, s, d \rightarrow r = \upsilon + s\,d$	*resposta*
(4)	$\mathcal{V}: g^r\,\beta^d \stackrel{?}{=} \gamma$	*verificação*

Note-se que, sendo o protocolo cumprido, $r - s\,d = \upsilon$ e, por isso, $g^r\,(g^{-s})^d = g^{\upsilon}$ o que conduz a $g^r\,\beta^d = \gamma$.

Considere-se de novo a estrutura geral do protocolo de identificação (pag. 343) e as componentes a ele associadas.

A transformação de Fiat-Shamir acrescenta a este "setup" uma função de *hash* H e a operação concatenação de códigos representadas pelo operador $\|$. Implementa um esquema de assinaturas da seguinte forma:

Protocolo 12 : Assinatura de Fiat-Shamir

Assinatura O "prover" $\mathcal{P}$, titular do segredo s, produz a assinatura σ para a mensagem m

(1) $\mathcal{P}: d \leftarrow H(\mathsf{id} \| m)$
(2) $\mathcal{P}: d, s \rightarrow \sigma = \mathsf{id} \| r(s, d)$

Verificação O "verifier" $\mathcal{V}$ verifica a validade de $\sigma = \mathsf{id} \| r$ como assinatura de m.

(1) $\mathcal{V}: \mathsf{id}, r \leftarrow \sigma$
(2) $\mathcal{V}: d \leftarrow H(\mathsf{id} \| m)$
(3) $\mathcal{V}: \rho(\mathsf{id}, d, r) =? 1$

5.6 Assinaturas com recuperação de mensagem e o esquema de assinaturas Hermes

O esquema de assinaturas HERMES é um projecto público do Ministério da Defesa de França para um esquema de assinaturas com recuperação de mensagens, assente em grupos Diffie-Hellman, que segue a estrutura de uma transformação de Fiat-Shamir sobre um protocolo de identificação análogo ao protocolo de Schnorr.

Para caracterizar HERMES temos de olhar separadeamente para as suas duas componentes essenciais: o que é um assinatura com recuperação de mensagens e qual é o protocolo "desafio-resposta" a que se vai aplicar a transformação FS.

Assinaturas com recuperação de mensagem (ARN)

Designemos por $\mathcal{M}$ o espaço das mensagens e por $\mathcal{S}$ o espaço das assinaturas.

Quando uma mensagem $m \in \mathcal{M}$ é assinada digitalmente e enviada para um destinatário é necessário enviar não só a assinatura construída $\sigma \in \mathcal{S}$ mas também o texto da mensagem m.

Por vezes a estrutura das mensagens é tal que é possível ter uma solução mais eficiente (em termos de bits transmitidos) num esquema onde a própria assinatura contenha informação sobre a mensagem.

Um tal esquema é formado pelas seguintes componentes:

Função de redundância É uma função $\xi : \mathcal{M} \rightarrow \mathbb{N}$ com as seguintes proprieades:

1. ξ é uma função injectiva implementável por um algoritmo PPT determinístico.
2. É implementável PPT a função parcial ξ^{-1} que verifica $\xi^{-1}(u) = x$, quando $u = \xi(x)$, e $\xi^{-1}(u) = \perp$ quando $u \notin \xi(\mathcal{M})$.
3. ξ "espalha" os seus resultados "uniformemente" por todo $\mathbb{N}$. Formalmente, existe um $l > 0$ suficientemente grande tal que, para todo $n > 0$, $|\xi(\mathcal{M}) \cap \mathbb{B}^n|/2^n \leq 2^{-l}$.

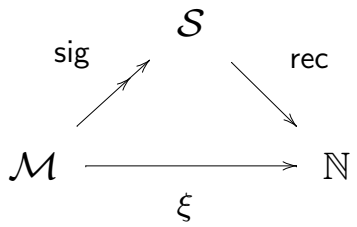
Funcional de assinatura

$\mathsf{sig}_s : \mathcal{M} \rightarrow \wp(\mathcal{S})$, é uma funcional que depende de uma chave privada s, e é unidireccional.

Função de recuperação

$\mathsf{rec}_\beta : \mathcal{S} \rightarrow \mathbb{N}$ é uma função que depende da chave pública β. e que verifica a condição de correcção

$$\sigma \in \mathsf{sig}_s(m) \quad \Leftrightarrow \quad \xi(m) = \mathsf{rec}_\beta(\sigma) \tag{122}$$



Num tal esquema a assinatura σ é aceite quando de verifica

$$\xi^{-1}\left(\mathsf{rec}_\beta(\sigma)\right) \neq \perp \tag{123}$$

e, se for aceite, o valor de $\xi^{-1}\left(\text{rec}_\beta(\sigma)\right)$ recupera a mensagem m.

Esquema de assinaturas Hermes

O esquema de assinaturas HERMES actua sobre mensagems m decompostas em duas componentes, $m = m_1 \| m_2$ e só trata a primeira m_1 num esquema ARM.

Assume-se que ambos agentes conhecem a segunda parte m_2.

HERMES assume as seguintes componentes

- Um grupo de Diffie-Hellman $\langle \mathbb{G}, g, r \rangle$;
- Duas funções de *hash*: $H_1 : G \to \mathbb{Z}_t$, com $t \gg r$, e $H_2 : \mathbb{Z}_t \to \mathbb{Z}_r$;
- Uma função de redundância $\xi : \mathbb{Z}_r \to \mathbb{Z}_t$;
- Um gerador $\mathcal{U}_r$ de números aleatórios uniformemente distribuídos em $\mathbb{Z}_r$.

A correcção deste esquema é facilmente verificado; note-se que, com $e = -H_2(z\|m_2)$, se tem $g^x = g^\upsilon \beta^{-e}$. Portanto $H_1(g^\upsilon) = H_1(g^x \beta^e) = h$. Como $r = H_1(g^\upsilon) \oplus \xi(m_1) = h \oplus \xi(m_1)$ e como $\tau = h \oplus z$, temos de concluir que $\tau = \xi(m_1)$.

Protocolo 13 : Hermes: Assinatura com Recuperação de Mensagem

Geração de chaves $\mathcal{P}$ gera a chave privada s e a chave pública $\beta = g^s$

(1) $\mathcal{P}: s \leftarrow \mathcal{U}_r \quad ; \quad \mathcal{P}: s \rightarrow \beta = g^s$

Assinatura $\mathcal{P}$ gera uma assinatura ARM σ da componente m_1, usando m_2 como chave

(1) $\mathcal{P}: \upsilon \leftarrow \mathcal{U}_r \quad ; \quad \mathcal{P}: z \leftarrow H_1(g^{\upsilon}) \oplus \xi(m_1)$
(2) $\mathcal{P}: x \leftarrow \upsilon + s\, H_2(z \,\|\, m_2) \quad ; \quad \mathcal{P}: x, z \rightarrow \sigma = z \| x$

Recuperação $\mathcal{V}$ recupera, de σ e m_2, um valor $\tau = \xi(m_1)$

(1) $\mathcal{V}: z, x \leftarrow \sigma \quad ; \quad \mathcal{V}: e \leftarrow -H_2(z \,\|\, m_2)$
(2) $\mathcal{V}: h \leftarrow H_1(g^x\, \beta^e) \quad ; \quad \mathcal{V}: h, z \rightarrow \tau = h \oplus z$

Verificação $\mathcal{V}$ aceita a mensagem sse τ estiver na imagem $\xi(\mathbb{Z}_r)$,

(1) $\mathcal{V}: \xi^{-1}(\tau) \neq \perp$

O esquema Hermes pode também ser obtido aplicando a transformação de Fiat-Shamir à seguinte variante do

protocolo de identificação de Schnorr.

Protocolo 14 : Identificação Schnorr-Hermes

Geração de chaves como no protocolo 13

Instância

(1)	$\mathcal{P}: v, v' \leftarrow \mathcal{U}_r, \mathcal{U}_r \quad ; \quad \mathcal{P}: \tau \leftarrow \xi(v')$	
(2)	$\mathcal{P}: v, \tau \rightarrow z = H_1(g^v) \oplus \tau$	*intensão*
(3)	$\mathcal{V}: d \leftarrow \mathcal{U}_r \quad ; \quad \mathcal{V}: d \rightarrow d$	*desafio*
(4)	$\mathcal{P}: d, s, v \rightarrow r = v + s\, d$	*resposta*
(5)	$\mathcal{V}: \tau' \leftarrow H_1(g^r\, \beta^{-d}) \oplus z$	
(6)	$\mathcal{V}: \xi^{-1}(\tau') \neq \perp$	*verificação*

5.7 Assinaturas Cegas

Uma das aplicações mais interessantes e complexas da criptografia são os protocolos de votação electrónica. Sem querer apresentar aqui uma descrição completa deste tipo de técnicas, parece ser claro que dois dos seus objectivos nucleares são as garantias de *anonimato do voto* e de *autenticidade do votante*.

Se, por um lado, o votante deve ser autenticado por uma qualquer autoridade (a *mesa* de voto) também se deve garantir que essa autoridade não tem conhecimento do voto específico do votante. Quando a mesa reconhece a legitimidade do votante, deve autenticar o voto respectivo sem o conhecer; em seguida deve passar essa informação ao *escrutinador* que faz a contagem dos votos.

Este esboço de protocolo é suficiente para se perceber que a sua componentes fundamental é uma técnica de autenticação designada por **assinatura cega**.

Numa assinatura cega (conceito introduzido por CHAUM nos anos 80) existem dois agentes: o emissor $\mathcal{E}$ e o "prover" $\mathcal{P}$. O objectivo de $\mathcal{E}$ é obter uma assinatura de $\mathcal{P}$ sobre uma mensagem m por si escolhida, sem que $\mathcal{P}$ conheça essa mensagem.

Uma solução "aparentemente óbvia" consistiria em esconder a mensagem através de uma função de *hash* H: o emissor calcularia $x \leftarrow H(m)$ e fornecia ao "prover" apenas x; ao assinar x, $\mathcal{P}$ continuaria sem conhecer m.

O protocolo de votação electrónica ilustra imediatamente porque é que esta não é uma solução aceitável; se a mensagem m for um voto, dado que existe um número limitado de alternativas de voto, ao "prover" bastaria

percorrer todas essas alternativas até encontrar aquela que tivesse um "hash" igual a x; desta forma determinaria o voto. Portanto, *esconder m de $\mathcal{P}$* exige também esconder o respectivo *hash*.

O protocolo original de Chaum baseia-se nas assinaturas RSA.

Protocolo 15 : Assinatura Cega (Chaum)

"Setup" RSA O "prover" $\mathcal{P}$ gera um módulo RSA produto de dois primos, $n = p \cdot q$ e $\phi \leftarrow (p-1)(q-1)$. Gera a chave privada s e a respectiva chave pública k.

(1) $\mathcal{P}: s \leftarrow \mathbb{Z}_\phi^* \quad ; \quad s \rightarrow k = s^{-1} \pmod{\phi}$

Assinatura $\mathcal{E}$ cifra o *hash* $x = H(m)$; $\mathcal{P}$ assina o criptograma respectivo; $\mathcal{E}$ recupera a assinatura e verifica-a.

(1) $\mathcal{E}: x \leftarrow H(m) \quad ; \quad \mu \leftarrow \mathbb{Z}_n^* \quad ; \quad \mu, h \rightarrow y = \mu^k \cdot h \pmod{n}$
(2) $\mathcal{P}: s \rightarrow x = y^s \pmod{n}$
(3) $\mathcal{E}: \sigma \leftarrow \mu^{-1} \cdot x \pmod{n} \quad ; \quad \sigma^k \pmod{n} \stackrel{?}{=} h$

Correcção

Porque $y = \mu^k \cdot h$ tem-se $x = y^s = \mu^{k\,s} \cdot h^s = \mu \cdot h^s$ (pelo teorema RSA, $\mu^{k\,s} = \mu$); portanto, $\sigma = \mu^{-1} \cdot x = h^s$. Desta forma $\mathcal{E}$ recupera a assinatura h^s da mensagem M. O "prover" $\mathcal{P}$ conhece $\mu^k \cdot h$ e consegue calcular $\mu \cdot h^s$; nenhum destes valores permite-lhe determinar μ ou h.

Um segundo protocolo de assinaturas usa as componentes comuns dos grupos Diffie-Hellman apresentados no protocolo 7 (página 331).

Protocolo 16 : Assinatura Cega (Schnorr)

Setup O "prover" $\mathcal{P}$ escolhe uma instância de um grupo DH, publicita-o; gera uma chave privada e publicita a respectiva chave pública obtendo a sua autenticação de um TA.

(1) $\mathcal{P}$: $\quad \rightarrow \langle \mathbb{G} \ , \ g \ , \ r \rangle \quad ; \quad a \leftarrow \mathbb{Z}_r^* \quad ; \quad a \rightarrow \beta = g^a$

Assinatura O emissor $\mathcal{E}$ obtém uma assinatura σ para a mensagem m usando o segredo de $\mathcal{P}$

(1) $\mathcal{P}$: $\upsilon \leftarrow \mathbb{Z}_r^* \quad ; \quad \upsilon \rightarrow \gamma = g^{\upsilon}$
(2) $\mathcal{E}$: $w, u, c \leftarrow \mathbb{Z}_r^* \quad ; \quad \mu \leftarrow \gamma^w \, g^u \, \beta^c \quad ; \quad y \leftarrow H(\mu \| m) \quad ; \quad \rightarrow x = w^{-1} \, (y - c)$
(3) $\mathcal{P}$: $a, \upsilon \rightarrow s = \upsilon - a \, x$
(4) $\mathcal{E}$: $\gamma \stackrel{?}{=} g^s \cdot \beta^x \quad ; \quad z \leftarrow w \, s + u \quad ; \quad \mu, m \rightarrow \sigma = z \, \| \, y$

Verificação

(1) $\mathcal{V}$: $(z, y) \leftarrow \sigma \quad ; \quad \mu' \leftarrow g^z \cdot \beta^y \quad ; \quad y' \leftarrow H(\mu' \| m) \quad ; \quad y \stackrel{?}{=} y'$

Alguns pontos sobre os objectivos do protocolo através do grau de conhecimento dos agentes envolvidos e das suas crenças quanto à autenticidade da informação.

1. A produção da assinatura σ é um protocolo síncrono que envolve colaboração entre o emissor $\mathcal{E}$, que conhece a mensagem a assinar m, e o "prover" $\mathcal{P}$, que conhece a chave privada a usada na assinatura.

   O emissor usa o segredo do "prover" (a sua chave privada a) sem nunca o conhecer em qualquer fase do protocolo. Pelo seu lado o "prover" não deve conhecer m em qualquer fase do protocolo ou o seu "hash".

   Na verificação, o verificador $\mathcal{V}$ conhece a mensagem m; assume-se que a recebeu do emissor via um canal privado (por exemplo, usando uma cifra) e conhece a informação pública do "prover" (a sua chave pública).
2. O verificador tem de acreditar na autenticidade da chave oública do "prover". Por isso um TA tem de fornecer um certifificado que associe o "prover" à sua chave pública.

   O emissor tem de ter garantias que o "prover" agiu de boa fé antes de publicitar a assinatura da mensagem.

Por estas razões, o protocolo de assinatura inclui:

- Um acto de compromisso – passo (1) – onde o "prover" gera um segredo segredo υ (uma "chave de sessão") e compromete-se com a respectiva "chave pública" γ.
- A produção pelo emissor – passo (2) – de um "hash" da mensagem m que é cifrado; só o respectivo criptograma x é conhecido pelo "prover".

- A assinatura σ é construída em duas fases; na primeira – passo (3) – é executada pelo "prover"; a segunda – passo (4) – é executada pelo emissor.
  O "prover" produz, no passo (3), uma pré-assinatura s do "hash" cifrado x, usando o segredo υ, com que se comprometeu, e a sua chave privada a. O emissor, no passo (4), assegura-se da autenticidade de s usando o compromisso γ, antes de, com segredos próprios, usar s para construir a assinatura final.

Para analisarmos a **correcção** deste protocolo convém notar que:

1. Dada a forma como γ e β são gerados, verificam-se, no passo (2), as seguintes igualdades
$$\mu \;=\; g^{w\,\upsilon+u+a\,c} \qquad , \qquad w\,x+c \;=\; y$$
2. As igualdades que resultam dos passos (3) e (4) são, respectivamente,
$$a\,x+s \;=\; \upsilon \qquad , \qquad w\,s+u \;=\; z$$
3. No passo (4), a verificação da pré-assinatura calcula $g^s\,\beta^x = g^{s+a\,x}$ e compara-o com $\gamma = g^{\upsilon}$. Se o "prover" agiu de boa fé no passo (3), produz um s que satisfaz a igualdade $a\,x+s=\upsilon$; portanto esta "boa fé" consta-se na comparação $\gamma \;=?=\; g^s\,\beta^x$.
4. A verificação da assinatura σ calcula $\mu' = g^{z+a\,y}$. Atendendo às igualdades anteriores, tem-se
$$z+a\,y \;=\; w\,s+u+a\,w\,x+a\,c \;=\; w\,(s+a\,x)+u+a\,c \;=\; w\,\upsilon+u+a\,c$$

Portanto

$$\mu' = g^{z+a\,y} = g^{w\,v+u+a\,c} = \mu$$

Os valores y e y' são "hashs" de m calculados, respectivamente, com as chaves μ e μ'; dado que as chaves coincidem, os "hashs" também têm de coincidir.

Quanto à **segurança** convém referir a alguns pontos:

Poderia parecer, à primeira vista, que o segredo c é irrelevante. A correcção estaria assegurada se, no passo (2), c não fosse gerado aleatoriamente e se fixasse uma constante c usada em todas as instâncias do protocolo. Note-se porém que os valores de x e y são ambos públicos após a produção da assinatura; se c fosse conhecido, então seria trivial, a um atacante, calcular o segredo w, dada a relação $w\,x + c = y$. Em seguida, dado que s e z são públicos, o atacante pode calcular o outro segredo u, usando a relação $z = w\,s + u$, e finalmente μ.

Do mesmo modo, quaisquer circunstâncias que permitam descobrir um dos três segredos w, u, c, pemitem descobrir os dois restantes segredos e, por isso, permitem construir um ataque semelhante a este.

Conhecendo w, u, c, o atacante dispõe de informação que lhe permite gerar mensagens arbitrárias $m_1 \neq m$, o respectivo "hash" cifrado y_1 e a assinatura $z_1 \| y_1$, sem passar pela intervenção do "prover". Para isso, basta escolher um w_1 que verifique $w_1\,x + c = y_1$ e calcular $z_1 = w_1\,s + u$. Como o valor de $s + a\,x$ se mantém inalterado, o algorimo de verificação continua a ter sucesso com esta nova assinatura e com a mensagem m_1.

5.8 Protocolos de Acordo de Chaves

O protocolo Diffie-Hellman (protocolo 6, página 325) foi a primeira técnica criptográfica de chave pública publicada. Foi introduzido em 1976 e está na base do interesse em grupos cíclicos como suporte à especificação de técnicas de chave pública.

O protocolo Diffie-Hellman pertence à classe dos "protocolos de acordo de chaves" que, em linhas gerais, são

183 Noção

Um **protocolo de acordo de chaves** *é um protocolo síncrono envolvendo dois agentes legítimos (designados por* **principais***) que, através de uma troca sucessiva de mensagens usando um canal público num meio hostil, obtêm informação suficiente para lhes permitir calcular um segredo comum* λ.

Esta definição genérica tem de ser concretizada numa série de condições de correcção e segurança. As condições de correcção determinam como se devem comportar os agentes numa execução legítima do protocolo. As condições de segurança estipulam propriedades que devem ser verificadas em qualquer execução do protocolo (legítima ou não).

Para as definir é necessário concretizar um pouco mais o que significa alguns conceitos que acabámos de referir: agente, canal público, mensagem, meio hostil, segredo, etc.

Para isso começamos por concretizar um pouco mais o que é um *"protocolo"*.

Cada execução específica do protocolo designa-se por **instância**. Cada instância é uma sequência finita de **passos**; cada passo de protocolo tem um *autor* (identificado pelo seu nome) e é uma computação que produz uma *mensagem*.

Cada protocolo especifica um conjunto finito de computações disponíveis para a criação de mensagens e disponibiliza um determinado número de constantes designadas por "parâmetros de execução". Cada mensagem é criada, com estas computações, a partir dos parâmetros de execução, do nome do autor, da sua informação privada, de mensagens anteriores e de consultas a oráculos.

Os oráculos disponíveis são específicos de cada protocolo mas, tipicamente, incluem:

- Um gerador $\mathfrak{N}$ que, em cada invocação, gera um novo inteiro. Cada "output" de $\mathfrak{N}$ designa-se por **nounce**[52]. A sequência de "nounces" é suficientemente aleatória e sem repetições.
- Um "trusted agent" `TA` que, ao receber um nome A, determina uma chave pública de A.

O facto da computação executado no passo p usar uma mensagem calculada no passo q, estabelece naturalmente uma **relação de precedência** entre estes passos: p é **anterior** ou **precede** q. Uma **história** do protocolo é uma sequência (eventualmente infinita) de passos de protocolo, resultantes de uma sequência de instâncias, mantendo a relação de precedência específica de cada instância.

[52] Number Only Used onCE.

Nota

Ou seja: se numa instância particular, o passo p precede o passo q, então na história p ocorre antes de q. Isto não impede que ocorram, entre p e q, passos provenientes de outras instâncias do protocolo envolvendo, eventualmente, outros agentes.

Um **canal público** é uma memória de longo prazo onde são depositadas as diferentes mensagens de uma história do protocolo conjuntamente com informação sobre a relação de precedência entre as mensagens. A menos que tal informação conste na mensagem, o canal público não conhece o seu autor.

Cada agente é determinado pelo seu **nome**, pela sua **informação privada** e pelos seus **privilégios** de acesso ao canal público. Estes privilégios podem ser:

- **Escuta:** o agente tem conhecimento de todas ou parte das mensagens no canal público. Este privilégio pode ser *local*, se se refere apenas à instância presente do protocolo, ou *global*, se se refere a todas as instâncias presente e passadas do protocolo.
- **Execução:** o agente tem possibilidade de executar novos passos de protocolo ou repetir passos anteriores.
- **Controlo:** o agente tem a capacidade de negar, selectiva ou globalmente, privilégios de outros agentes.

Nos protocolos de acordo de chave entende-se que os **agentes principais** são caracterizados pelo seu nome, pela sua informação privada e têm privilégio de escuta local e execução de novos passos do protocolo. O *"meio hostil"* é personificado num agente designado por **atacante** que tem privilégios global de escuta, execução e tem privilégio controlo selectivo de escuta e execução de um determinado conjunto de agentes.

□

Para caracterizar, dentro da noção genérica de protocolo, o que é um protocolo de acordo de chaves começamos por definir uma condição que especifica o seu objectivo.

Correcção

O segredo é comum no sentido em que, em cada instância do protocolo, o mesmo valor de λ é determinado por ambos os agentes principais.

As condições de segurança vão ser expressas em termos de uma história do protocolo envolvendo um único atacante C e, em cada instância, dois agentes principais A e B.

Confidencialidade

Em nenhum momento, C conhece a chave λ acordado pelo par de agentes A, B.

Autenticidade dos agentes

Em cada instância, o agente A tem prova que o outro interveniente no protocolo é B e, vice-versa, B tem prova que o outro interveniente é A.

Autenticidade do segredo

Cada um dos agentes (A ou B) tem prova que o outro calculou o mesmo segredo λ que ele próprio calculou.

O protocolo determina uma **chave estática** λ quando, para o mesmo par de agentes A e B, toda a instância do protocolo envolvendo estes agentes determinam a mesma chave. O segredo é uma **chave efémera** (ou "de sessão") quando, para o mesmo par de agentes, cada instância do protocolo determina uma chave uniformemente distribuída dentro de um domínio vasto de possibilidades. Isto é, a chave é efémera quando a sua incerteza é igual (ou ligeiramente inferior) ao seu comprimento e é estática se a sua incerteza for nula.

A efemeridade da chave está ligada à sua confidencialidade. Uma chave estática pode sempre, por motivos alheios à execução do protocolo, passar a ser conhecida pelo atacante C. Por exemplo, um dos agentes pode ser indiscreto no seu uso. Uma chave efémera é menos sensível a essas "fugas de informação" no sentido em que o seu eventual conhecimento pelo atacante deixa de ser relevantes para utilizações futuras. A efemeridade da chave levanta também a questão da autenticidade temporal; isto é, a autenticidade relativa às instâncias particulares do protocolo.

Surgem assim variantes temporais das condições de segurança anteriores.

Novidade
: Numa determinada instância do protocolo, A e B têm a certeza que λ não pode ser determinada a partir de instâncias anteriores do mesmo protocolo.

Autenticidade temporal dos agentes
: Numa determinada instância do protocolo, A tem a certeza que o outro agente é B participando na mesma instância do protocolo. Analogamente, para B.

Autenticidade temporal do segredo

Numa determinada instância do protocolo, cada agente tem prova que o outro calculou, na mesma instância, o mesmo segredo que ele próprio determinou.

A indiscrição coloca-se também em relação à informação privada dos agentes principais. Isto levanta a questão da manutenção da confidencialidade dos segredos acordados com informação privada. Temos assim uma condição de segurança adicional.

Confidencialidade a longo prazo ("foward secrecy")

O conhecimento, pelo atacante, da chaves privada de um conjunto de agentes não lhe permite ter conhecimento das chaves de sessão acordadas em instâncias anteriores onde algum desses agentes tenha intervido.

□

Muitos protocolos de acordo de chaves não necessitam de uma concretização específica das estruturas matemáticas subjacentes; são definidos, de forma abstracta, recorrendo a cifras, funções de "hash" e assinaturas com segurança perfeita. Nomeadamente, muitos protocolos importantes usam exclusivamente técnicas simétricas (cifras e funções de "hash").

Porém estes protocolos exigem um mecanismo prévio de distribuição das chaves simétricas o que conduz, naturalmente, às questões de autenticidade desse mecanismo.

Nesta secção vamos considerar protocolos baseados em técnicas de chave pública e, em particular, protocolos assentes em grupos cíclicos. Vamos começar por considerar o protocolo de Diffie-Hellman, e algumas variantes simples, e tentar verificar quais destas condições são satisfeitas e quais são violadas. Vamos assumir um grupo Diffie-Hellman $\langle \mathbb{G}, g, r \rangle$ de ordem prima; vamos assumir os elementos comuns referidos na página 331. Vamos, finalmente, assumir que todos os passos de protocolo lidam com valores apenas em dois domínios: $\mathbb{Z}_r$ ou numa órbita G de g de ordem r. Estas condições garantem que os protocolos não são vulneráveis a ataques baseados na pequena ordem dos grupos cíclicos.

Protocolo 17 : Diffie Hellman - chave estática

Setup Criação de pares de chaves de longa duração e iniciação do TA com a respectiva chave pública.

(1) $A: a \leftarrow \mathbb{Z}_r^* \quad ; \quad \rightarrow \beta = g^a \quad ; \quad B: b \leftarrow \mathbb{Z}_r^* \quad ; \quad \rightarrow \gamma = g^b$
(2) O TA reconhece e autentica os pares (A, β) e (B, γ)

Run

(1) $A: \mathsf{TA}(B, \gamma) =?= 1 \quad ; \quad \lambda_a \leftarrow \gamma^a$
(2) $B: \mathsf{TA}(A, \beta) =?= 1 \quad ; \quad \lambda_b \leftarrow \beta^b$

Neste, e nos restantes protocolos desta secção, amos assumir também que existem oráculos

- $\mathfrak{N}$: produz um "nounce" em cada activação.
- TA é uma decisão que sob "input" do nome A e de uma chave pública β decide se o par (A, β) é autêntico.

O TA determina também, na fase de "setup", o grupo Diffie-Hellman $\langle \mathbb{G}, g, r \rangle$ sob o qual o protocolo se desenvolve.

O protocolo estático acaba por não produzir qualquer mensagem para o canal público. Cada um dos agentes obtém, do TA, a chave pública do outro e calcula imediatamente o segredo com a sua chave privada.

A propriedade da correcção é trivialmente satisfeita já que $\lambda_a = \gamma^a = (g^b)^a = (g^a)^b = \beta^b = \lambda_b$. Como não existem mensagens, a única informação disponível ao intruso C é a que provém do TA; C pode obter também as chaves públicas; no entanto só conhecerá o segredo se conseguir resolver o problema CDHP $g, g^a, g^b \rightarrow g^{a\,b}$; portanto a condição de confidencialidade é garantida.

A autenticidade dos agentes é garantida porque as chaves públicas são autenticadas pelo TA. Não há garantias da autenticidade do segredo já que nenhum dos agentes pode ter garantia de que o outro o chega a calcular.

No entanto, a desvantagem essencial deste protocolo é o facto de a chave acordade ser estática: entre os mesmos dois agentes o segredo acordado é sempre o mesmo. Nomeadamente qualquer indiscrição na informação privada, compromete todas os usos anteriores da chave; a longo prazo, o protocolo é inseguro.

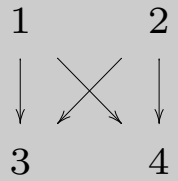
Para suprir esta falha de segurança, a sugestão imediata será o de substituir as chaves de longa duração a e b, por chaves locais a cada instância do protocolo. Como consequência imediata, estas chaves não podem ser autenticadas.

Protocolo 18 : Diffie Hellman - versão efémera

Setup O TA escolhe um grupo Diffie-Hellmam $\langle \mathbb{G}, g, r \rangle$ e torna pública esta informação.

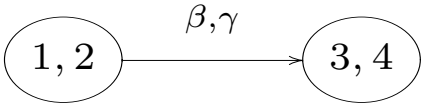
Run

$$
\begin{array}{llll}
(1) & A: & a \leftarrow \mathbb{Z}_r^* \;\; ; & a \rightarrow \beta = g^a \\
(2) & B: & b \leftarrow \mathbb{Z}_r^* \;\; ; & b \rightarrow \gamma = g^b \\
(3) & A: & \lambda_a \leftarrow \gamma^a & \\
(4) & B: & \lambda_b \leftarrow \beta^b &
\end{array}
$$



Em primeiro lugar note-se que, ao contrário do protocolo estático, existe uma sequência de passos e uma clara relação de precedência entre eles. De facto a precedência é definida $(1), (2) \rightarrow (3)$ e $(1), (2) \rightarrow (4)$.

Como não está estabelecida nenhuma precedência entre (1) e (2) nem entre (3) e (4), estes pares de passos podem-se considerar "simultâneos" em termos de uma execução legítima do protocolo. Assim a execução pode ser representada por



Este protocolo permite, porém, execuções ilegítimas que designaremos por **ataques**. Para analisar como se desenvolvem estes ataques vamos considerar uma notação genérica que nos permite uma representação mais detalhada da sucessão de passos do protocolo.

Cada agente X (A, B ou um atacante) gera ou calcula segredos s durante os vários passos do protocolo. Representamos por $X.s$ o valor do segredo s que X conhece. Caso o agente execute o mesmo passo vários vezes, os valores do mesmo segredo s gerado em sucessivos passos é denotado por $X.s_1, X.s_2, ,$ etc.

Vamos ter também uma definição genérica dos passos deste protocolo. Note-se que o protocolo tem dois tipos de passos que designaremos por T e S e que são instanciados com os agentes principais ao passo e com a informação que vão buscar ao canal público.

$$T(X) \doteq \left\{X:\ k \leftarrow \mathbb{Z}_r^*\ ;\ k \rightarrow g^k\right\} \quad , \quad S(X,u) \doteq \left\{X:\ \lambda \leftarrow u^{X.k}\right\}$$

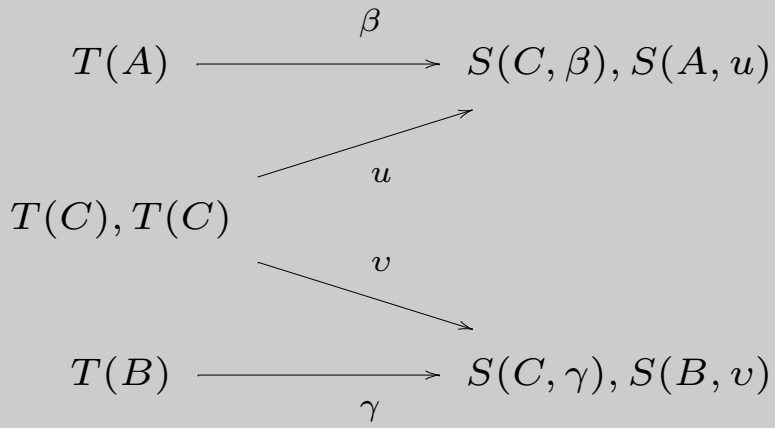
Com esta notação os passos (1) e (2) são, respectivamente, $T(A)$ e $T(B)$, identificando as chaves privadas efémeras a e b com k; isto é, $a = A.k$ e $b = B.k$. Do mesmo modo os passos (3) e (4) são, respectivamente, $S(A,\gamma)$ e $S(B,\beta)$, identificando $\lambda_a = A.\lambda$ e $\lambda_b = B.\lambda$.

Usaremos a notação $T(X) \xrightarrow{u} S(Y,u)$ para indicar que u foi publicidado por $T(X)$ e é usado em $S(Y,u)$. As relações de precedência são, então, para todo X, Y, u

$$T(X) \xrightarrow{u} S(Y,u) \quad , \quad T(X) \longrightarrow S(X,u)$$

Vamos considerar, agora, um atacante C que inicia dois passos T. É agora possível uma história como a que representa-mos no seguinte diagrama[53].

Ataque 3 Homem-No-Meio



Existe uma ocorrência do passo T e do passo S para A e para B e duas ocorrências, de ambos os passos, para C. Os passos T produzem

$$\beta = g^{A.k} \qquad u = g^{C.k_1} \qquad \upsilon = g^{C.k_2} \qquad \gamma = g^{B.k}$$

[53] Por simplicidade não se representa explicitamente as dependências entre um passo T e um passo S envolvendo o mesmo agente.

Os passos S produzem

$$A.\lambda = u^{A.k} = g^{C.k_1\ A.k} \qquad C.\lambda_1 = g^{A.k\ C.k_1} \qquad C.\lambda_2 = g^{B.k\ C.k_2} \qquad B.\lambda = g^{C.k_2\ B.k}$$

Por isso, o par de passos $S(A,u), S(C,\beta)$ acordou o mesmo segredo $g^{C.k_1\ A.k}$ enquanto que o par de passos $S(B,\upsilon), S(C,\gamma)$ acordou o segredo $g^{C.k_2\ B.k}$.

Desta forma o atacante C foi capaz de acordar segredos tanto com A como com B. Como não existe qualquer autenticação dos agentes, nenhum dos agentes proncipais tem capacidade para identificar o seu parceiro no acordo. Portanto, A e B podem pensar que estão a comunicar entre si mas realmente estão a comunicar com C.

A não-autenticidade dos agentes traduz-se aqui em falha na autenticidade do segredo; de facto, nem A nem B podem confirmar que o segredo usado pelo outro é o mesmo que eles próprios calcularam e, como se verifica, o segredo é diferente.

□

Comparando as duas versões do protocolo Diffie-Hellam vê-se que o protocolo 17 fornece autenticação dos agentes, mas não garante autenticação do segredo nem novidade nesse segredo; em contra-partida o protocolo 18 produz uma chave fresca em cada instanciação mas não garante a autenticidade dessa chave nem dos agentes.

Para tentar aliar as vantagens de ambas as versões (e eliminar as falhas de segurança) vamos considerar um protocolo

que contém, para além do grupo Diffie-Hellman, um esquema de assinaturas e uma cifra. Usando a notação usual da escrita de protocolos usaremos a seguinte convenção

$\nu\, x \cdot m(x)$	a mensagem que resulta de instanciar $m(x)$ com um nounce x
$\{m\}_k$	criptograma resultante da cifra de m com a chave k
$\{x\}_k^{-1}$	texto claro obtido do criptograma x com a chave k
$[m]$	"hash" da mensagem m
$[m]_k$	MAC ("message authentication code") da mensagem m com a chave k
$(m)_X$	assinatura da mensagem m pelo agente X

Vamos também assumir que existe um oráculo Ver que verifica assinaturas. Este oráculo é implementado como uma decisão $\mathrm{Ver}(X, m, s)$ que. sob "input" de um nome X, uma mensagem m e de uma assinatura s, decide se s é uma assinatura autentica de m por X.

O protocolo *Station-To-Station* (STS) é um protocolo baseado no protocolo Diffie-Hellman mas onde os agente principais executam "passos complementares". O protocolo é sequencial no sentido em que existe um agente específico que o inicia e os passos seguintes são sempre respostas ao passo que o precede; existe só uma história que define uma execução legítima.

Protocolo 19 : Station-To-Station - versão anónima

(1) A: $a \leftarrow \mathbb{Z}_r^*$; $a \rightarrow \beta = g^a$

(2) B: $b \leftarrow \mathbb{Z}_r^*$; $k \leftarrow \beta^b$; $\rightarrow \sigma = g^b \,\|\, \left\{ \left(g^b \| \beta \right)_B \right\}_k$

(3) A: $(\gamma, t) \leftarrow \sigma$; $k \leftarrow \gamma^a$; $\mathsf{Ver}(B, \gamma \| \beta, \{t\}_k^{-1}) \stackrel{?}{=} 1$; $\rightarrow \upsilon = \left\{ (g^a \| \gamma)_A \right\}_k$

(4) B: $\mathsf{Ver}(A, \beta \| g^b, \{\upsilon\}_k^{-1}) \stackrel{?}{=} 1$

Convencionalmente os protocolos de acordo de chave representam-se não nesta forma extensa mas numa forma mais abstracta onde são apresentadas apenas as mensagens que são publicadas no canal público.

Protocolo 20 : STS - versão anónima

(1) A: $\nu\, a \cdot \beta$ sendo $\beta = g^a$

(2) B: $\nu\, b \cdot \gamma \| \{(\gamma \| \beta)_B\}_{k_B}$ sendo $\gamma = g^b$ e $k_B = \beta^b$

(3) A: $\{(\beta \| \gamma)_A\}_{k_A}$ sendo $k_A = \gamma^a$

(4) B: ε

Uma diferença importante entre estas duas representações é o facto de toda a informação privada ser, agora, gerada sob a forma de "nounces" e não "simplesmente" aleatória (o que permite repetições). Nesta representação são

consideradas implícitas todas as decomposições de mensagens em componentes e todas as verificações de assinaturas. Por isso, no último passo, está implícita a verificação da assinatura produzida no passo anterior; isto é, o passo existe só que, mesmo que a verificação tenha sucesso, não produz qualquer mensagem.

Basicamente STS é o protocolo Diffi-Hellman aumentado com autenticação dos agentes e do segredo. A correcção está assegurada pela equação $(g^a)^b = (g^b)^a$. A novidade da chave é assegurada pela geração dos "nounces" nos passos (1) e (2). A autenticidade dos agentes é reconhecida porque ambos assinam uma mensagem previamente conhecida por ambos ($g^b \| g^a$ ou $g^a \| g^b$) e essa assinatura é verificada pelo outro agente no passo seguinte. A confirmação da chave concretiza-se porque estas assinaturas são cifradas com a chave acordada; para verificar a assinatura é preciso decifrá-la primeiro.

A presença da cifra nos passos (2) e (3) é essencial; sem ela, no passo (3), um atacante "homem-no-meio" poderia substituir a assinatura $(\gamma \| \beta)_A$ pela sua própria assinatura. Deste modo A e B completam o protocolo, mas A acredita que comunicou com B enquanto que B acredita que comunicou com C.

Este tipo de falha permanece mesmo no protocolo original. Considere-se, por exemplo, o ponto de vista do agente A; a menos que ele saiba previamente qual é o nome do agente com quem deve acordar o protocolo, não lhe é possível no passo (3) verificar a assinatura produzida no passo (2). O agente A não garante a autenticidade de "agentes imprevistos".

Isto ocorre porque a informação que cada agente tem sobre o outro interveniente tem de ser fornecida por um outro canal externo ao do protocolo. Idealmente a identificação dos intervenientes no protocolo deveria estar contida nas

próprias mensagens do protocolo.

Pode-se pensar numa modificação ao protocolo onde cada mensagem contenha explicitamente, no cabeçalho, a origem e o destino previstos. Uma primeira abordagem seria,

(1)	$A:$	$\nu\, a \cdot A\|\|B\|\|\beta$	sendo $\beta = g^a$
(2)	$B:$	$\nu\, b \cdot B\|\|A\|\|\gamma\|\| \{(\gamma\|\|\beta)_B\}_{k_B}$	sendo $k_B = \beta^b$ e $\gamma = g^b$
(3)	$A:$	$A\|\|B\|\| \{(\beta\|\|\gamma)_A\}_{k_A}$	sendo $k_A = \gamma^a$

Este protocolo é obviamente inseguro porque os nomes não estão autenticados; assim qualquer intruso pode re-enviar mensagems substituindo o nome do agente origem, do agente destino ou de ambos. O melhor que se pode dizer é que este protocolo não é mais inseguro que o STS original mas não resolve o problema da identificação dos agentes.

Obviamente que se podia incluir os cabeçalhos, $A||B$ ou $B||A$, na componente assinada e cifrada. De facto, a autenticação do nome dos agentes suprime também, e em parte, a necessidade de cifrar a assinatura para confirmar a autenticidade do segredo; isto porque que cada um dos agentes sabe que o outro conhece exactamente os mesmos valores de β e γ que ele próprio conhece.

Isto não é exactamente a mesma coisa do que a autenticidade da chave acordada. O que cada agente sabe é que o

outro tem os elementos necessários para calcular a chave certa; não sabe, porém, se o outro (agindo de má fé) não usa outra chave. Para obter a confirmação das chaves é necessário juntar um MAC da informação assinada.

Cada agente publicita duas mensagens; não é necessário que em ambas seja identificado o emissor e o destinatário. A versão simplificada de um protocolo STS com identificadores e autenticação MAC é

Protocolo 21 : STS - versão com identificadores

			sendo
(1)	A:	$\nu\, a \cdot A \parallel \beta_A$	$\beta_A = g^a$
(2)	B:	$\nu\, b \cdot B \parallel \beta_B \parallel (m_B)_B \parallel [m_B]_{k_B}$	$\beta_B = g^b$, $m_B = \beta_B \Vert \beta_A \Vert A$ e $k_B = (\beta_A)^b$
(3)	A:	$(m_A)_A \parallel [m_A]_{k_A}$	$m_A = \beta_A \Vert \beta_B \Vert B$ e $k_A = (\beta_B)^a$
(4)	B:	ε	

Este protocolo garante "foward secrecy"; de facto as únicas chaves privadas são as usadas para gerar assinaturas e a chave de sessão $g^{a\,b}$. A chave de sessão $g^{a\,b}$ depende apenas dos "nounces" gerados nessa instância do protocolo; não depende das chaves usadas para as assinaturas e, se o gerador de "nounces" for suficientemente aleatório, não depende de "nounces" noutras instâncias do protocolo.

□

O protocolo STS usa um esquema de assinaturas genérico. Se essa assinatura fosse, por exemplo, o DSA existe

uma óbvia redundância já que muitas das primitivas criptográficas do protocolo Diffie-Hellman básico se repetem na assinatura DSA. Assim, na perspectiva de melhorar a eficiência do STS, parece razoável procurar harmonizar as primitivas usadas no protocolo de acordo de chaves básico, com as que são usadas nas assinaturas digitais.

Uma primeira tentativa de harmonização é o protocolo Arazi, integrado no DSS (Digital Signature Standard).

Protocolo 22 : Arazi

Setup TA escolhe um grupo de Diffie-Hellman $\langle \mathbb{G}, g, r \rangle$ de ordem prima. A e B geram chaves privadas de longa duração a e b, publicitam as respectivas chaves públicas $\beta_A = g^a$ e $\beta_B = g^b$. Os pares (A, β_A) e (B, β_B) são autenticadas pelo TA. Assume-se os elementos comuns referidos na página 331.

Run

(1) $A: \ \nu\, x \cdot \lambda_A \| s_A$ sendo $\lambda_A = g^x$ e $x\, s_A = H(\lambda_A) + a\, f(\lambda_A)$

(2) $B: \ \nu\, y \cdot \lambda_B \| s_B$ sendo $\lambda_B = g^y$ e $y\, s_B = H(\lambda_B) + b\, f(\lambda_B)$

(3) $A: \ \varepsilon$

A verificação das assinaturas s_X para a mensagem λ_X (sendo X o agente A ou o agente B) é $\lambda_X^{s_X} =?= h(\lambda_X)\, \beta_X^{f(\lambda_X)}$.

A chave acordada é a do protocolo Diffie-Hellman: $k_A = \lambda_B^x$ e $k_B = \lambda_A^y$.

Este protocolo distingue-se do protocolo Diffie-Hellman básico apenas pelo facto de juntar às mensagens g^x e g^y as suas assinaturas. Se os agentes principais forem conhecidos *à priori*, então o protocolo garante a autenticidade dos agentes.

O protocolo falha em termos de "foward secrecy". Note-se que se uma das chapas privadas de longa duração (por exemplo, a) for comprometida no futuro então a chave acordada k_A está comprometida. Isso deriva da equação $x\, s_A = H(\lambda_A) + a\, f(\lambda_A)$ usada para calcular a assinatura s_A: se a for conhecido então x é conhecida já que todos os restantes elementos são públicos. Conhecido x, calcula-se $k_A = \lambda_B^x$.

As equações de geração de assinatura fornecem a redundância que permite este tipo de ataques. Pode-se provar, por exemplo, que se, numa instância particular, a chave acordada $k_A = k_B$ for comprometida, então estas equações podem fornecer informação que permita calcular esta mesma chave em instâncias futuras.

□

Se comparar-mos a assinatura aqui calculada com o DSA (protocolo 9, pag. 335) vemos que a assinatura do protocolo de Arazi não gera randomização; limita-se a usar o mesmo "nounce" (x ou y) que é usado na geração da chave acordada. Pode-se tentar resolver esta questão com dois "nounces" distintos em cada agente: um para a assinatura e outro para a chave.

Neste caso, a aleatorização da assinatura (com os "nounces" v e u) evita que os "nounces" da chave (x e y) sejam determinados mesmo quando ambas as chaves privadas a e b estão comprometidas. Portanto este protocolo resolve

as falhas na "foward secrecy" do protocolo de Arazi.

Protocolo 23 : Hirose-Yoshida

Setup coincide com a fase "setup" no protocolo de Arazi (protocolo 22)

Run

		sendo
(1)	$A:\ \nu\, x \cdot \lambda_A \| A$	$\lambda_A = g^x$
(2)	$B:\ \nu\, y, \upsilon \cdot \lambda_B \| \sigma_B \| s_B \| B$	$\lambda_B = g^y\ ,\ \sigma_B = H(g^\upsilon \| \lambda_B \| \lambda_A)$
		$s_B = \upsilon - \sigma_B\, y - \sigma_B^2\, b$
(3)	$A:\ \nu\, u \cdot \sigma_A \| s_A$	$\sigma_A = H(g^u \| \lambda_A \| \lambda_B)\ ,\ s_A = u - \sigma_A\, x - \sigma_A^2\, a$
(4)	$B:\ \varepsilon$	

A verificação das assinatura de B calcula g^υ como $g^{s_B}\,(\lambda_B\,\beta_B^{\sigma_B})^{\sigma_B}$ e, depois, compara σ_B com $H(g^\upsilon \| \lambda_B \| \lambda_A)$. Para a assinatura de A procede-se de forma análoga.

A chave acordada é a do protocolo Diffie-Hellman: $k_A = \lambda_B^x$ e $k_B = \lambda_A^y$.

□

Para finalizar esta breve introdução ao protocolos de acordo de chaves é importante referir que as técnicas aqui

propostas são apenas uma das componentes dos protocolos operacionais. Algumas das razões

(i) Dado que todos os protocolos de acordo de chaves são parametrizados por um conjunto de técnicas (cifras, funções de "hash", assinaturas) e envolvem escolhas sobre parâmetros das estruturas matemáticas, os protocolos operacionais têm de envolver trocas de mensagens que conduzam a um acordo sobre as técnicas criptográficas usadas e sobre os valores dos parâmetros.

Estes "acordos de parâmetros" são sujeitos a ataques análogos aos dos protocolo principal. Por isso é necessário incluir aqui, igualmente, componentes dirigidas à autenticidade.

(ii) Os protocolos operacionais têm de responder a um tipo de ataques que não está normalmente presente nos protocolos que aqui estudámos. Nomeadamente os protocolos operacionais, como o nome indica, têm de responder a questões ligadas à operacionalidade; nomeadamente, se é vulnerável a ataques de "denial of service" (DoS).

É razoável assumir que não existe nenhuma defesa eficaz contra os ataques DoS se assumir-mos que o atacante tem controlo completo do canal público. No entanto vários mecanismos têm sido propostos para, pelo menos, minimizar as probabilidade de um ataque com sucesso.

Na perspectiva de um ataque DoS, os agentes principais são vistos de forma assimétrica: um deles é um **"servidor"** S, que é objecto do ataque, e o outro é um **"cliente"** C que age de forma hostil tentando esgotar os recursos do servidor.

Algumas abordagens possíveis são:

1. *stateless connections*
   O cliente armazena toda a informação de estado (mensagens anteriores e "nounces") que o servidor necessita e envia-as para o servidor conforme é necessário. Isto requer que a informação esteja cifrada e que sejam necessários mecanismos de autenticação que garantam ao servidor que está a receber a informação certa. No entanto o servidor não necessita de armazenar localmente estado e poupa nos seus recursos. Em contrapartida há custos adicionais quer em esforço computacional como em tráfego que façam com que, de facto, os recursos do servidor sejam esgotados ainda mais depressa.
2. *cookies*
   O servidor atrasa a necessidade de estado local enviando um "cookie" ao cliente, sempre que ele tenta estabelecer uma ligação. Esse *cookie* contém informação identificadora da ligação e um "semi-nounce" cifrados com uma chave privada do servidor. Nesta fase S ainda não se comprometeu com o protocolo nem criou informação de estado. O cliente tem de devolver o "cookie" na próxima mensagem e dentro de um intervalo de tempo limitado. O "semi-nounce" não necessita de ser único em cada "cookie" mas é actualizado muito frequentemente.
3. *outras autenticações*
   Os *cookies* são uma forma básica de autenticação prévia. Pode-se estender outras formas de autenticação a todas as mensagens do protocolo. Isto é, obviamente, muito custoso mas tem a vantagem de o servidor poder parar imediatamente o protocolo logo que detecta uma mensagem não autenticada. Em alternativa o servidor pode, de forma incremental, ir aumentando as suas exigências de autenticação consoante a sua carga de ligações activas aumenta. Por exemplo, pode requerer autenticação via dos protocolos de identificação desafio-resposta.

5.9 Protocolos para Trocas Cifradas de Chaves

O objectivo geral dos protocolos *"Encrypted Key Exchange"* (EKE) é o de um acordo ou transporte de uma chave de sessão usando, como forma de autenticação das mensagens, uma "password" partilhada pelos agentes principais.

O argumento por detrás desta opção, por oposição aos protocolos de acordo de chaves descritos na secção anterior, vai no sentido de considerar que os mecanismos de autenticação aí usados, baseados em assinaturas digitais, requererem chaves privadas de longa duração com um número elevado de bits e envolvem mecanismos complexos de gestão e certificação dessas chaves.

A alternativa aqui proposta usa "passwords", com pouca incerteza[54], para autenticação. O facto de a "password" ser pequena facilita a sua gestão pelos agentes principais; nomeadamente, a "password" é fácil de recordar e, normalmente, não requer mecanismos de protecção muito sofisticados.

As assumpções básicas, que condicionam a segurança desta família de técnicas, são

1. A "password" (ou uma sua imagem) é conhecida só pelos agentes principais. Um adversário que, de qualquer forma, conheça "password" pode intervir no protocolo fazendo-se passar por um dos agentes principais.
2. A "password" é usada como chave de uma cifra simétrica; devido à reduzida dimensão do espaço de chaves, a cifra é vulnerável a um ataque por força-bruta com texto conhecido.

[54] Obviamente, se já existisse uma chave partilhada π de elevada incerteza, o protocolo seria irrelevante!

3. A "password" não é vulnerável a ataques de dicionário por adversários passivos. Isto é, um adversário que escute um número N de criptogramas gerados com a chave π, mantém uma incerteza sobre a chave $\vartheta(\pi)$ que é essencialmente constante, independentemente de N.

Quase sempre uma chave π, num espaço de chaves K, é usada para cifrar elementos de uma órbita $G = [g]$ de um grupo de Diffie-Hellman $\langle \mathbf{G}, g, r \rangle$. Assume-se que G está contido no domínio de todas as possíveis chaves $\pi \in K$, mas pode ser apenas uma pequena parte desse domínio.

Dado um criptograma $y = \pi\tilde{}(u)$, com $u \in G$, um atacante passivo, mesmo não conhecendo u, pode tentar encontrar possíveis chaves π para as quais $\pi^{-1}(y) \notin G$. Tais chaves, se existirem, serão imediatamente eliminadas do espaço de chaves K diminuindo a incerteza em π.

Por isso, a não vulnerabilidade a ataques de dicionário, implica a seguinte condição de segurança

$$\forall \pi, \pi\tilde{} \in K\,,\, \forall u \in G \quad . \quad \pi^{-1}(\pi\tilde{}(u)) \subseteq G \tag{124}$$

□

Um adversário activo pode sempre reduzir ligeiramente a incerteza em π, gerando uma tentativa $\pi\tilde{}$, entrando no protocolo com esta este valor fazendo-se passar por um legítimo titular de π e, consoante o protocolo corre com

sucesso ou não, decidir se a sua tentativa é a "password" correcta. Desta forma, fazendo $2^{\vartheta(\pi)-1}$ tentativas, pode subir a propbabilidade de escolha da chave correcta acima de $1/2$.

Atendendo à nossa primeira assunção, isto implica poder completar o protocolo com sucesso fazendo-se passar por um dos titulares legítimos de π. Desta forma faz sentido considerar a seguinte condição de segurança para esta classe de protocolos:

O protocolo EKE diz-se **seguro contra um ataque de dicionário activo** se, iniciando um número de instâncias do protocolo que seja substancialmente inferior a $2^{\vartheta(\pi)-1}$, nenhum adversário consegue obter probabilidade superior a $1/2$ de terminar com sucesso.

Nos protocolos desta secção assume-se

Elementos Comuns

(1) Um grupo Diffie-Hellman $\langle \mathbb{G}, g, r \rangle$ de ordem r prima; $G = [g]$ denota a órbita do gerador g.
(2) Uma cifra $\{\cdot\}$, com um espaço de chaves K.
(3) Funções de *hash* $H : \mathbb{N} \longrightarrow G$ e $h : G \longrightarrow K$.
(4) Funções de *hash* $H_i : \mathbb{N} \longrightarrow \mathbb{B}^t$, $i = 1, 2, 3$, independentes.

As funções de *hash* $H_i : \mathbb{N} \longrightarrow G$ são independentes quando, para todo $i \neq j$, é computacionalmente intratável encontrar strings x, y tais que $H_i(x) = H_j(y)$.

A partir de uma única função de *hash*, $\theta : \mathbb{N} \longrightarrow \mathbb{B}^t$, pode-se sempre gerar uma sequência de funções de *hash* independentes. Por exemplo, pode-se definir $H_i(x) = \theta(i \| x)$ ou $H_i(x) = \theta(\{x\}_{k_i})$, sendo $k_i = h(H(i))$.

□

O paradigma de protocolo EKE é o chamado **protocolo de Bellovin & Merrit** e que é, como muitos outros, baseado no protocolo de Diffie-Hellman. Usa uma "password" π e uma cifra simétrica como única forma de autenticação.

Como o protocolo não usa esquemas de assinaturas, não requer chaves privadas e públicas de longa duração. Assim, todas as chaves privadas a, b são "nounces" e todas as chaves públicas (as mensagens g^a e g^b) são efémeras.

Essencialmente o protocolo cifra com π todas estas chaves públicas efémeras antes de as enviar pelo canal público. A sua autenticidade depende crucialmente da assunção de que π é só conhecido por A e B.

A autenticidade da chave Diffie-Hellman acordada $g^{a\,b}$ é verificada usando dois novos "nounces" $n, n^{\sim}$ que são trocados cifrados com uma chave derivada da chave Diffie-Hellman.

Protocolo 24 : EKE ("encrypted key exchange") Bellovin & Merrit

Seup TA escolhe uma *password* $\pi \in K$ e distribui-a, de forma confidencial, pelos agentes principais.

Run

(1) $A:\ \nu\, a \cdot A \parallel \{g^a\}_\pi$

(2) $B:\ \nu\, b, n \cdot \left\{g^b\right\}_\pi \parallel \{n\}_k$ sendo $k = h(g^{a\,b})$

(3) $A:\ \nu\, n^{\sim} \cdot \{n^{\sim} \| n\}_k$

(4) $B:\ \{n^{\sim}\}_k$

Uma variante deste protocolo força uma escolha particular de cifra $\{\cdot\}$ e da sua função de derivação de chaves h.

Protocolo 25 : PAK ("passowrd authenticated key") Boyco *et al.*

Setup TA escolhe π e distribui esta chave pelos agentes principais

Run

(1) $A:\ \nu\, a \cdot g^a\, \mu$ sendo $\mu = H(A\|B\|\pi)$

(2) $B:\ \nu\, b \cdot g^b \| H_1(\lambda)$ sendo $d = g^{a\,b}$ e $\lambda = A\|B\|g^a\,\mu\|g^b\|d\|\mu$

(3) $A:\ H_2(\lambda)$

A chave acordada é escolhida como $k = H_3(\lambda)$.

6.Criptografia Baseada na Identidade

Em 1984, Shamir[55] propôs o conceito de IBC ("Identity Based Cryptography") como um conjunto de técnicas assimétricas onde a identidade de um agente, descrita sob a forma de textos, endereços, mensagens, etc., pudesse ser usada como chave de cifras ou como meio de verificar uma assinatura.

Se compararmos esta abordagem com a "clássica" PKI vemos que, nesta última, cifrar mensagens ou verificar assinaturas é feito pelas chaves públicas. São estes items de informação que, perante os outros agentes, determinam a identidade do agente destinatário da mensagem cifrada ou autor de uma mensagem assinada.

Este tipo de identificação tem, porém, algumas desvantagens:

1. Nas técnicas PKI as chaves públicas são números gerados a partir de chaves privadas que, por seu turno, são valores escolhidos aleatoriamente. Por isso não têm qualquer significado fora do contexto onde são usadas. Por exemplo, vendo os bits de uma chave pública não se tem nenhuma indicação sobre quem é o legitimo titular dessa chave.
2. Exigem um item adicional (um certificado) para estabelecer a necessária relação entre chave e contexto de utilização. Esses certificados exigem, por seu turno, um enquadramento complexo para os gerar, distribuir e

[55] O "S" do RSA.

manter. Exige Autoridades de Certificação, exige um enquadramento jurídico para regular a sua actividade, exige agências reguladoras, etc.

A forma como as chaves privadas e públicas são geradas nas técnicas clássicas PKI faz com que a identificação contextual do agente (nome, endereços, fotografia, atributos de acesso, etc.) seja completamente independente da sua identificação criptográfica. A primeira é ditada pelo contexto informativo onde o agente interage; a segunda é ditada pela conveniência computacional da técnica criptográfica usada.

As técnicas IBC introduzem uma mudança radical: *a identificação contextual coincide com a identificação criptográfica*.

A identidade de um agente A, num contexto informativo, é essencialmente uma representação da legitimidade, perante terceiros, da titularidade de direitos. Por exemplo quando A assina digitalmente uma mensagem, a identidade descreve a autoria da mensagem; quando A se identifica perante uma qualquer instituição, a identidade descreve os direitos do agente na sua relação com essa instituição; quando uma mensagem destinada a A é cifrada, a identidade representa o direito de A de conhecer o seu conteúdo; etc.

Com esta visão da identidade, as consequências da abordagem IBC são diversas:

1. A identidade de um agente assume a forma de uma descrição textual de atributos[56] mas também dos direitos

[56] É costume classificar os atributos usados em identificação em três categorias: *atributos institucionais* (p.ex. nome, endereço,

a que esta identidade está associada; por exemplo, pode conter um período de validade ou uma ACL ("access control list"). Uma identidade, usada num sistema de controlo de acessos, poderia ser

```
"Antonio Silva <asilva@qualquer.sitio> # 08/12/31 23:59 # universal: read, write"
```

que contém atributos institucionais, um limite temporal de validade e o tipo do acesso que permite.

2. A titularidade de uma identidade tem de ser estabelecida por uma gente de confiança (TA ou "trusted agent"). Essa titularidade, que inclui nomeadamente a capacidade para assinar as mensagens ou decifrar um criptograma, é verificada pelas técnicas criptográficas através do conhecimento que o agente titular tem de um determinado segredo; i.e uma *chave privada*.

   Desta forma, em qualquer esquema ou protocolo IBC, um passo essencial é o cálculo e distribuição, pelo TA, da chave privada associada à identidade do agente titular.

3. Adicionalmente o TA tem de fornecer ao contexto que usa a identidade de A (os agentes que verificam as assinaturas emitidas por A ou cifram as mensagens destinadas a A), garantias que foi realmente o TA o emissor da chave privada.

   Isso exige uma identificação publica do TA que, para tal, recorre a um par *chave pública+chave privada* no sentido clássico. Assim, a chave pública do TA vai ter de intervir em todos os esquemas públicos (verificação de assinaturas ou cifra de mensagens) onde intervenha a identidade.

organização) que são determinados pelo contexto social onde o identificado está inserido, *atributos naturais* (p.ex. data e local de nascimento, identificação dos pais) que são determinados pelo seu nascimento e que correspondem ao acto de aquisição de personalidade jurídica, e *atributos físicos* tais como dados biométricos, fotografia, etc.

4. Ao contrário das técnicas clássicas, no IBC a identidade (chave pública) precede a chave privada. Por exemplo, uma mensagem pode ser cifrada sem que o destinatário disponha da capacidade para a decifrar; só quando tem necessidade de provar a titularidade da identidade é que o destinatário a apresenta perante o TA que, usando os mecanismos de validação que entender adequados, emite posteriormente a chave privada respectiva.

Dentro das técnicas IBC é conveniente identificar duas classes: a classe que agrupa as técnicas que têm como objectivo a autenticação (assinaturas, protocolos de identificação, acordos de chaves, etc.) e a classe das técnicas que estão orientadas à confidencialidade (cifras, principalmente).

As primeiras técnicas caem dentro do que se designa por **IBA ("Identity Based Autentication")**; as segundas caem dentro do que designa por **IBE ("Identity Based Encryption")**.

6.1 Técnicas IBC de Shamir

Como exemplos paradigmático dos princípios que acabámos de citar, vamos descrever duas técnicas (um esquema de asinaturas e um protocolo de acordo de chaves) que adaptam os princípios com que, em 1984, Shamir introduziu as técnicas IBC. Como seria de esperar do autor, estes protocolos são uma evolução do esquema de assinaturas RSA e usa os elementos comuns típicos de uma técnica RSA.

No esquema de assinaturas intervêm um "prover" $\mathcal{P}$ e um "verifier" $\mathcal{V}$. A identidade de $\mathcal{P}$ é representada pelo próprio símbolo.

Protocolo 26 : IBC de Shamir - Elementos comuns

Elementos comuns: É gerado um módulo RSA m e determinados
- um subgrupo primo $\Gamma \subseteq \mathbb{Z}_n^*$ de ordem r onde são feitas as multiplicações;
- duas funções de *hash*: $H : \Gamma \times \mathbb{B}^* \to \Gamma$ e *hash* $\mathsf{ID} : \mathbb{B}^* \to \Gamma \setminus 1$.

"Setup e Extract" T gera um par $\langle s, k \rangle$ de chaves RSA; torna m e k público. Distribui, por canal privado, a chave privada do "prover".

(1) $T : s \leftarrow \mathbb{Z}_\phi^* \quad ; \quad T : s \to k = s^{-1} \pmod{\phi}$
(2) $T : p \leftarrow \mathsf{ID}(\mathcal{P}) \quad ; \quad \lambda \leftarrow p^s$
(3) $\mathcal{P} : \lambda \leftarrow T.\lambda$

Protocolo 27 : Esquema de Assinaturas IBC de Shamir

Assinatura O "prover" constrói uma assinatura σ de um texto M

(1) $\mathcal{P}: \upsilon \leftarrow \mathbb{Z}_\phi^* \quad ; \quad \gamma \leftarrow \upsilon^k \quad ; \quad h \leftarrow H(\gamma, M)$

(2) $\mathcal{P}: \lambda \rightarrow \sigma = \gamma \,\|\, \lambda\, \upsilon^h$ *assinatura*

Verificação o "verifier" $\mathcal{V}$ valida o triplo $\langle M, \sigma, \mathcal{P} \rangle$.

(1) $\mathcal{V}: \gamma, r \leftarrow \sigma \quad ; \quad h \leftarrow H(\gamma, M) \quad ; \quad p \leftarrow \mathsf{ID}(\mathcal{P})$

(2) $\mathcal{V}: r^k\, \gamma^{-h} \stackrel{?}{=} p$ *verificação*

JUSTIFICAÇÃO: Seja $p = \mathsf{ID}(\mathcal{P})$ e $h = H(\gamma, M)$. Como $\lambda = p^s$, pelo teorema RSA, tem-se $\lambda^k = p$; tem-se $r = \lambda\, \upsilon^h$, logo $r^k\, \gamma^{-h} = \lambda^k \cdot (\upsilon^k)^h\, \gamma^{-h} = p\, \gamma^h\, \gamma^{-h} = \mathsf{ID}(\mathcal{P})$.

Para o protocolo de acordo de chaves assume-se que existem dois agentes de identidades A e B que pretendem acordar num segredo comum usando canais abertos. O TA calcula as chaves privadas λ_A e λ_B associadas às identidades A e B e distribui-as aos agentes apropriados.

Cada instância do protocolo assenta ainda numa mensagem M pública que pode ser uma identidade de referência (por exemplo a identidade da técnica criptográfica usada, descrevendo textualmente a sua informação pública) ou uma mensagem específica sobre a qual o protocolo define um mecanismo de autenticação. No final cada agente não

só tem confiança no segredo acordado mas também tem uma prova de que o outro agente conhece a mensagem M (ou, pelo menos, o seu *hash*).

Protocolo 28 : Acordo de Chaves IBC de Shamir

Extact T calcula e distribui as chaves privadas de cada agente

$$\begin{array}{ll} (1) & T\colon p_A, p_B \leftarrow \mathsf{ID}(A)\,,\,\mathsf{ID}(B) \quad ; \quad \lambda_A, \lambda_B \leftarrow p_A^{-s}\,,\,p_B^{-s} \quad ; \quad M \rightarrow r = \mathsf{ID}(M) \\ (2) & A\colon \lambda_A \leftarrow T.\lambda_A \quad ; \quad B\colon \lambda_B \leftarrow T.\lambda_B \end{array}$$

Inicialização Cada agente gera um segredo e publicita a informação pública correspondente

$$\begin{array}{ll} (1) & A\colon a \leftarrow \mathbb{Z}_\phi^* \quad ; \quad B\colon b \leftarrow \mathbb{Z}_\phi^* \\ (2) & A\colon a, \lambda_A \rightarrow \sigma_A = \lambda_A\, r^a \quad ; \quad B\colon b, \lambda_B \rightarrow \sigma_B = \lambda_B\, r^b \end{array}$$

Geração do segredo:

$$(1) \quad A\colon \kappa_A \leftarrow \left(\sigma_B^k\, p_B\right)^a \quad ; \quad B\colon \kappa_B \leftarrow \left(\sigma_A^k\, p_A\right)^b$$

JUSTIFICAÇÃO: A definição das chaves privadas $lambda_A, \lambda_B$ e o teorema RSA asseguram que, em Γ,

$$\sigma_A^k \, p_A \;=\; \lambda_A^k \, r^{a\,k} \, p_A \;=\; p_A^{-k\,s} \, r^{a\,k} \, p_A \;=\; r^{a\,k}$$

e, de forma análoga, $\lambda_B^k \, p_B = r^{b\,k}$. Portanto,

$$\kappa_A \;=\; \left(\sigma_B^k \, p_B\right)^a \;=\; r^{a\,b\,k} \;=\; \left(\sigma_A^t \, p_A\right)^b \;=\; \kappa_B$$

Todos os protocolos de acordo de chaves requerem uma segunda fase em que ambos os agentes verificam que a chave acordada é a mesma. Para isso usam uma qualquer cifra simétrica para cifrar, com a chave que conhecem, informação que supostamente é comum.

Protocolo 29 : Acordo de Chaves de Shamir - Verificação

Elementos Comuns Um cifra simétrica $\langle E, D\rangle$

Informação pública $A\colon \kappa_A \to y_A = E(\kappa_A, \kappa_A) \quad ; \quad B\colon \kappa_B \to y_B = E(\kappa_B, \kappa_B)\,.$

Verificação $A\colon \kappa_A \stackrel{?}{=} D(\kappa_A, y_B) \quad ; \quad B\colon \kappa_B \stackrel{?}{=} D(\kappa_B, y_A)\,.$

6.2 Grupos "Gap" Diffie-Hellman (GDHG)

Seja $\langle \mathbb{G}, g, r \rangle$ um grupo Diffie-Hellman (abreviadamente **DHG** [57]); isto é um grupo cíclico onde o CDHP é considerado intratável.

Recordemos que, no CDHP é dado g^a e g^b e pretende-se calcular $g^{a\,b}$. Pode-se enfraquecer as exigências deste problema assumindo que também é dado um terceiro argumento g^c e pretende-se apenas decidir se c (que é desconhecido) coincide com $a\,b$ (que também é desconhecido).

Note-se que, devido às propriedades algébricas do grupo cíclico, verifica-se $c = a\,b$ se e só se $g^c = g^{a\,b}$. O problema em questão designa-se por "Decisão Diffie-Hellman" e pode, por isso, ser descrito como

Problema da Decisão Diffie-Hellman (DDHP)
Dados g^a, g^b *e* g^c *em* $\mathbb{G}$, *decidir se* $g^c = g^{a\,b}$.

Obviamente que DDHP é redutível a CDHP; de facto, se fornecermos g^a e g^b como *input* de um oráculo CDHP, ele force-nos $g^{a\,b}$ que pode ser comparado com g^c.

O inverso não é necessariamente verdade: existem grupos onde CDHP é comutacionalmente intratável e o DDHP não é intratável. De facto uma nova família de técnicas criptográficas surgiu nos últimos anos aproveitando este "gap" entre os dois problemas.

[57] Diffie-Hellman Group.

A definição do DDHP que apresentámos está demasiadamente informal: afinal o que significa "decidir"? Uma formalização probabilística do DDHP tem de tomar como referência uma determinada família de testes $\mathcal{T}$ e usar essa família para distinguir as linguagens que representam as duas situações que pretendemos diferenciar.

Concretamente, as linguagens em questão codificam triplos $\langle x, y, z\rangle \in G^3$ e definem-se do modo seguinte.

184 Definição
Dado um grupo DH $\langle \mathbb{G}, g, r\rangle$ *; então:*

- *A linguagem dos* **triplos Diffie-Hellman** *da órbita de* g *– representada por* $\mathsf{DHT}(g)$ *– é o conjunto de todas as bit-strings que codificam triplos* $\langle g^a, g^b, g^{a\,b}\rangle$ *quando* $a, b \in \mathbb{N}$ *percorrem todo o seu domínio.*
- *A linguagem dos* **pseudo triplos Diffie-Hellman** *da órbita de* g *– representada por* $\mathsf{PDHT}(g)$ *– é o conjunto de todas as bit-strings que codificam triplos* $\langle g^a, g^b, g^c\rangle$ *quando* $a, b, c \in \mathbb{N}$ *percorrem todo o seu domínio.*

A diferença entre estas duas linguagens está na terceira componente do triplo: em $\mathsf{DHT}(g)$ a componente $g^{a\,b}$ é determinada pelas duas primeiras componentes; na linguagem $\mathsf{PDHT}(g)$ a componente g^c é "livre". É óbvio que a segunda linguagem contém a primeira.

Agora pode-se dar uma forma mais precisa à definição do DDHP.

Problema da Decisão Diffie-Hellman Probabilística (DDHP)
Dados g e um sistema de testes $\mathcal{T}$, encontrar um $\mathcal{T}$-diferenciador – ver definição 164, pag. 224 – das linguagens $\mathsf{DHT}(g)$ *e* $\mathsf{PDHT}(g)$.

Esta definição pode ser generalizada considerando, para além do grupo DH $\langle \mathbb{G}, g, r \rangle$, um outro grupo de torsão Γ, não necessariamente Diffie-Hellman.

185 Definição
Dado um grupo DH $\langle \mathbb{G}, g, r \rangle$ e um grupo de torsão Γ; então:

- *A linguagem dos* ***co-triplos Diffie-Hellman*** *da órbita de g e Γ– representada por* $\mathsf{co\text{-}DHT}(g, \Gamma)$ *– é o conjunto de todas as bit-strings que codificam triplos $\langle g^a, \gamma, \gamma^a \rangle$ quando $a \in \mathbb{N}$ e $\gamma \in \Gamma$ percorrem os respectivos domínios.*
- *A linguagem dos* ***pseudo co-triplos Diffie-Hellman*** *da órbita de g – representada por* $\mathsf{co\text{-}PDHT}(g, \Gamma)$ *– é o conjunto de todas as bit-strings que codificam triplos $\langle g^a, \gamma, \gamma^c \rangle$ quando $a, c \in \mathbb{N}$ e $\gamma \in \Gamma$ percorrem os respectivos domínios.*

Problema da Decisão Diffie-Hellman Probabilística Generalizado (co-DDHP)
Dados g, Γ e um sistema de testes $\mathcal{T}$, encontrar um $\mathcal{T}$-diferenciador das linguagens $\mathsf{co\text{-}DHT}(g, \Gamma)$ *e* $\mathsf{co\text{-}PDHT}(g, \Gamma)$.

É óbvio que esta segunda reformulação generaliza a primeira: basta considerar o caso em que Γ coincide com o grupo $\mathbb{G}$; neste caso, fazer γ percorrer Γ é equivalente a fazer b percorrer $\mathbb{N}$ e considerar $\gamma = g^b$.

□

Como já referimos os problemas da computação Diffie-Hellmam e da decisão Diffie-Hellman não são equivalentes. Em vários grupos DH (onde CDHP é intratável) existem algoritmos PPT para resolver o DDHP.

Em tais grupos existe uma "falha" (**gap**) entre os graus de complexidade computacional associados aos dois problemas; este "gap" pode ser explorado em técnicas criptográficas que baseiem o uso legítimo da técnica numa soluçaõ do DDHP e o uso ilegítimo (ataques) na solução do CDHP.

186 Definição
Um grupo DH $\langle \mathbb{G}, g, r \rangle$ *diz-se* ***gap-Diffie-Hellman*** *– abreviadamente* ***GDHG*** [58] *– para uma família de testes* $\mathcal{T}$ *quando DDHP tem solução para esse conjunto de testes.*

Sabendo que, neste grupos, CDHP não é computacionalmente viável mas DDHP é tratável, coloca-se a questão de como implementar esta decisão. Dois tipos de estratégia são conhecidos: as baseadas nos oráculos co-CDHP e as baseadas em emparelhamentos ("pairings").

[58] Gap Diffie-Hellman Group

co-CDHP Vamos assumir que se conhece um grupo cíclico Γ, com a mesma ordem r do que $\mathbb{G}$, para o qual co-CDHP é tratável. Formalmente assume-se

Existe um oráculo que, dados $\gamma \in \Gamma$ e $g^a \in G$ produz γ^a.

Decide-se se $\langle g^a, g^b, g^c \rangle$ é ou não um triplo DH em $\mathbb{G}$, da seguinte forma:

$\mathrm{DDHP}(g^a, g^b, g^c)$
(1) Gerar um $\gamma \in \Gamma$ aleatorio de ordem r,
(2) Usando o oráculo co-CDHP com γ e g^a, calcular γ^a,
(3) Usando o oráculo co-CDHP com γ^a e g^b, calcular $\gamma^{a\,b}$,
(4) Usando o oráculo co-CDHP com γ e g^c, calcular γ^c
(5) Aceitar $\langle g^a, g^b, g^c \rangle$ como triplo DH se e só se $\gamma^{a\,b} = \gamma^c$.

Algoritmo 1: Diferenciador DDHP derivado de um oráculo co-CDHP.

Note-se que, por γ ter a mesma ordem que o grupo, é um gerador de Γ e, portanto, verifica-se $\gamma^{a\,b} = \gamma^c$ sse $a\,b = c$ e, portanto, sse $g^{a\,b} = g^c$.

Emparelhamentos ("pairings") Vamos assumir que se conhece um grupo cíclico Γ, com a mesma ordem r de $\mathbb{G}$ e uma função $\mathbf{e}\colon G \times G \to \Gamma$ que satisfaz as seguintes propriedades:

(1) é *PT implementável*; i.e. existe um algoritmo PPT determinístico que implementa $\mathbf{e}$
(2) é *bilinear*; i.e., $\mathbf{e}(g^a, g^b) = \mathbf{e}(g, g)^{a\,b}$ para todos $a, b \in \mathbb{Z}_r$
(3) é *não-degenerado* no 1º argumento; i.e., $\mathbf{e}(g^a, g^c) = \mathbf{e}(g^b, g^c)$ sse $a = b$.

Uma tal função $\mathbf{e}$ designa-se por **emparelhamento**[59].

Dado um emparelhamento, decide-se se $\langle g^a, g^b, g^c\rangle$ é ou não um triplo DH em $\mathbb{G}$, da seguinte forma:

$\mathrm{DDHP}(g^a, g^b, g^c)$
(1) Calcular $\mathbf{e}(g^a, g^b)$
(2) Calcular $\mathbf{e}(g^c, g)$
(3) Aceitar $\langle g^a, g^b, g^c\rangle$ como triplo DH se e só se os dois valores anteriores coincidirem.

Algoritmo 2: Diferenciador DDHP derivado de um emparelhamento.

[59] Uma definição mais geral de emparelhamento considera 3 grupos G, G', Γ e uma função $\mathbf{e}\colon G \times G' \to \Gamma$ que seja PT implementável, bilinear e não-degenerada no 1º argumento.

Note-se que, pela bilinearidade de $\mathbf{e}$, $\mathbf{e}(g^a, g^b) = \mathbf{e}(g, g)^{a\,b} = \mathbf{e}(g^{a\,b}, g)$. Então $\mathbf{e}(g^a, g^b) = \mathbf{e}(g^c, g)$ verifica-se sse $\mathbf{e}(g^{a\,b}, g) = \mathbf{e}(g^c, g)$ e, usando a não-degenerência do emparelhamento, essa igualdade verifica-se sse $a\,b = c$.

A técnica criptográfica mais simples, baseada em emparelhamentos, é um acordo tripartido de chaves: três agentes A, B e C usam um grupo cíclico e um emparelhamento para acordarem num segredo comum.

O protocolo que se segue é uma generalização óbvia do clássico acordo de chaves de Diffie-Hellman e, como este, está sujeito a ataques "homem-no-meio" se os agentes não estiverem autenticados mútuamente. Por isso é normalmente acompanhado com algum sistema de assinaturas que garanta essa autenticidade[60].

Os **elementos comuns** são um grupo DH $\langle \mathbb{G}, g, r \rangle$ e um emparelhamento $\mathbf{e}\colon G \times G \longrightarrow \Gamma$. Seja $\boldsymbol{\kappa}$ a função definida por $\boldsymbol{\kappa}(x, y, a) = \mathbf{e}(x, y)^a$.

[60] Por questões de eficiência implementacional, o esquema de assinaturas usa, geralmente, o mesmo grupo DH $\mathbb{G}$. Porém a versão que vamos aqui apresentar abstrai em relação à componente de autenticação mútua já que isso não é relevante para o uso dos emparelhamentos.

Protocolo 30 : Acordo Tripartido de Chaves (Joux)

Geração de chaves Cada uma dos agentes gera um segredo aleatório e publicita a chave pública correspondente.

(1) $A\colon a \leftarrow \mathbb{Z}_r^* \quad ; \quad B\colon b \leftarrow \mathbb{Z}_r^* \quad ; \quad C\colon c \leftarrow \mathbb{Z}_r^*$

(2) $A\colon a \rightarrow \sigma_A = g^a \quad ; \quad B\colon b \rightarrow \sigma_B = g^b \quad ; \quad C\colon c \rightarrow \sigma_C = g^c$

Geração do segredo comum Cada agente usa o emparelhamento e o seu segredo próprio

(1) $A\colon \kappa_A \leftarrow \boldsymbol{\kappa}(\sigma_B, \sigma_C, a) \quad ; \quad B\colon \kappa_B \leftarrow \boldsymbol{\kappa}(\sigma_A, \sigma_C, b) \quad ; \quad C\colon \kappa_C \leftarrow \boldsymbol{\kappa}(\sigma_B, \sigma_A, c)$

Justificação: Facilmente se verifica, usando a bilinearidade do emparelhamento, $\kappa_A = \kappa_B = \kappa_C = \mathbf{e}(g,g)^{a\,b\,c}$.

6.3 IBC em Grupos Gap Diffie-Hellman

A família de técnicas IBA caem dentro das técnicas implementáveis em grupos gap-DH e, de certa forma, o protocolo de Shamir é uma excepção. Normalmente as técnicas de IBA são definidas sobre grupos DH e, dentro destes, os grupos Gap-Diffie-Hellman têm uma importância particular.

Vamos assumir, no que se segue, que $\langle \mathbb{G}, g, r \rangle$ é um grupo gap-Diffie-Hellman equipado com um oráculo DDHP.

DDHP(g^a, g^b, g^c)
– *retorna o valor* 1 *se e só se os argumentos formam um triplo DH*

Não interessa a forma específica como é construído um tal oráculo: pode vir de um oráculo co-CDHP, de um emparelhamento ou de qualquer outra metodologia. Qualquer que seja o oráculo, convém atender ao resultado

187 Facto
Seja $q \neq 1$ *um elemento arbitrário de* G*;* $\langle g^a, q^b, q^c \rangle$ *é um triplo de Diffie-Hellman sse* $a\,b = c$.

Prova: Como g é gerador do grupo, existe sempre um $n \neq 0 \in \mathbb{Z}_r$ tal que $q = g^n$. Portanto o triplo $\langle g^a, q^b, q^c \rangle = \langle g^a, g^{n\,b}, q^{n\,c} \rangle$ é um DHT sse $n\,a\,b = n\,c$. Como $\mathbb{Z}_r$ não tem divisores de zero e $n \neq 0$, temos de concluir que $a\,b = c$.

Elementos Comuns

(1) Um grupo Gap-Diffie-Hellman $\langle \mathbb{G}, g, r \rangle$ munido de um oráculo DDHP
(2) Um gerador aleatório $\mathbb{Z}_r$ de elementos de $\mathbb{Z}_r$ uniformemente distribuídos
(3) Duas funções de *hash* $\mathrm{ID}: \mathbb{B}^* \rightarrow G \setminus 1$ e $H: G \times \mathbb{B}^* \rightarrow \mathbb{Z}_r$

Protocolo 31 : Assinatura IBA baseada em GDHG (Chon Cha & Hee Cheon)

Setup O TA gera o seu par de chaves: a privada s e a pública $\beta = g^s$; β é externamente autenticada.
(1) $T: s \leftarrow \mathbb{Z}_r \quad ; \quad s \rightarrow \beta = g^s$

Extract O TA calcula, e fornece ao "prover" $\mathcal{P}$, a chave privada λ correspondente à identidade $p = \mathrm{ID}(\mathcal{P})$. O "prover" $\mathcal{P}$ só aceita essa chave se decide, com sucesso, que $\langle \beta, p, \lambda \rangle$ é um DHT.
(1) $T: p \leftarrow \mathrm{ID}(\mathcal{P}) \quad ; \quad \lambda \leftarrow p^s$
(2) $\mathcal{P}: p \leftarrow \mathrm{ID}(\mathcal{P}) \quad ; \quad \mathrm{DDHP}(\beta \ , \ p \ , \ \lambda) \ =? \ 1$

Sign O "prover" $\mathcal{P}$ produz a assinatura σ a partir do texto M com a chave privada λ
(1) $\mathcal{P}: p \leftarrow \mathrm{ID}(\mathcal{P}) \quad ; \quad \upsilon, \gamma \leftarrow \mathbb{Z}_r \ , \ p^{\upsilon} \quad ; \quad h \leftarrow H(\gamma, M)$
(2) $\mathcal{P}: \lambda \rightarrow \sigma = \gamma \parallel \lambda^{\upsilon + h}$

Verify o "verifier" valida o triplo formado pelo texto M, a assinatura σ e a identidade $\mathcal{P}$
(1) $\mathcal{V}: p \leftarrow \mathrm{ID}(\mathcal{P}) \quad ; \quad \gamma, r \leftarrow \sigma \quad ; \quad h \leftarrow H(\gamma, M) \quad ; \quad t \leftarrow \gamma \cdot p^h$
(2) $\mathcal{V}: \mathrm{DDHP}(\beta \ , \ t \ , \ r) \ =? \ 1$

JUSTIFICAÇÃO: Seja $p = \mathrm{ID}(\mathcal{P})$. Para autenticação de λ atenda-se a que $\beta = g^s$ e $\lambda = p^s$; o *input* do oráculo DDHP é $\langle \beta, p, \lambda \rangle = \langle g^s, p, p^s \rangle$ que é um DHT (facto 187). Na verificação da assinatura, como $\gamma = p^v$, tem-se $t = p^{v+h}$; como $\lambda = p^s$, tem-se $r = \lambda^{v+h} = p^{s\,(v+h)}$. Os elementos fornecidos ao oráculo são $\langle g^s, p^{v+h}, p^{s\,(v+h)} \rangle$ e eles formam um DHT.

Este esquema tem associado um protocolo de identificação de conhecimento zero do qual deriva por uma transformação de Fiat-Shamir.

Protocolo 32 : Identificaçãp IBA baseada em GDHG

Setup & Extract Como no protocolo 31: o TA publica a sua chave pública β e computa e distribui a chave privada λ associada à identidade $\mathcal{P}$.

O "prover" $\mathcal{P}$ autentica previamente a sua chave privada resolvendo uma instância de DDHP.

Instância

(1) $\mathcal{P}\colon p \leftarrow \mathrm{ID}(\mathcal{P}) \;\; ; \;\; v \leftarrow \mathbb{Z}_r \;\; ; \;\; v \rightarrow \gamma = p^v$ *intensão*

(2) $\mathcal{V}\colon d \leftarrow \mathbb{Z}_r \;\; ; \;\; d \rightarrow d$ *desafio*

(3) $\mathcal{P}\colon \lambda \rightarrow r = \lambda^{v+d}$ *resposta*

(4) $\mathcal{V}\colon p \leftarrow \mathrm{ID}(\mathcal{P}) \;\; ; \;\; t \leftarrow \gamma \cdot p^d \;\; ; \;\; \mathrm{DDHP}(\beta\ ,\ t\ ,\ r) \stackrel{?}{=} 1$ *verificação*

JUSTIFICAÇÃO: A mesma que no protocolo 31 e derivada do facto que, no final do protocolo, se verifica $\beta = g^s$, $t = p^{v+d}$ e $r = p^{s\,(v+d)}$.

A segurança de qualquer um destes protocolos assenta, crucialmente, na crença de que o CDHP não pode ser resolvido neste grupo G. De facto um oráculo CDHP, ao receber os *inputs* $\langle \beta, p \rangle$ (sendo $p = \mathrm{ID}(\mathcal{P})$), calcula a chave privada $\lambda = p^s$.

Com o mesmo oráculo, após o passo (2) do protocolo de identificação, um intruso, mesmo sem conhecer λ, pode calcular a resposta correcta r. Dado que tem acesso ao β e consegue calcular o $t = \gamma \cdot p^d$, usa o oráculo CDHP sobre o par $\langle \beta, t \rangle$ e determina r; pode, deste modo, assumir a identidade do "prover" e substituí-lo no passo (3).

Um esquema de assinaturas bastante mais simples é designado por BLS e não está orientado à identidade. Este mecanismo BLS já foi usado, por nós, nos dois anteriores protocolos para autenticar a chave privada do "prover".

Protocolo 33 : BLS "short signatures" (Boneh et al.)

Geração de chaves O "prover" $\mathcal{P}$ gera uma chave privada s e a espectiva chave pública β que deve ser autenticada por um mecanismo externo (certificado).

(1) $\mathcal{P}$: $s \leftarrow \mathbb{Z}_r^*$; $s \rightarrow \beta = g^s$

Sign Assinatura σ da mensagem M

(1) $\mathcal{P}$: $h \leftarrow H(\beta, M)$; $s \rightarrow \sigma = h^s$

Verify verificar a validade da assinatura σ sobre o texto M com a chave pública β

(1) $\mathcal{V}$: $h \leftarrow H(\beta, M)$; $\mathrm{DDHP}(\beta\ ,\ h\ ,\ \sigma) =?= 1$

JUSTIFICAÇÃO: $\beta = g^s$, h e $\sigma = h^s$ formam um DHT.

Este esquema pode ser facilmente adaptado facilmente a assinaturas cegas.

Protocolo 34 : BLS em assinatura cega

Geração de chaves Como no protocolo 33

Sign O "verifier" cifra a mensagem previamente e recupera a assinatura original num 3º passo.

(1) $\mathcal{V}: \mu \leftarrow \mathbb{Z}_r \quad ; \quad h \leftarrow H(\beta, M) \quad ; \quad \mu \rightarrow y = g^{-\mu} \cdot h$ – *cifra*

(2) $\mathcal{P}: s \rightarrow x = y^s$ – *assinatura*

(3) $\mathcal{V}: \sigma \leftarrow x \cdot \beta^{\mu}$ – *recuperação*

Recuperação Como no protocolo 33

Justificação:

Temos $x = \left(g^{-\mu} \cdot h\right)^s = g^{-\mu\, s} \cdot h^s$. Uma vez que $\sigma = h^s$ e $\beta = g^s$, será $x = \beta^{-\mu} \cdot \sigma$. Desta forma $\beta^{\mu} \cdot x$ recupera a assinatura σ.

Adicionalmente $\mathcal{P}$ conhece $g^{-\mu} \cdot h$ e é capaz de calcular $\beta^{-\mu} \cdot \sigma$; nenhum destes termos lhe permite determinar μ, h ou σ.

6.4 IBE sobre Emparelhamentos

A definição de emparelhamento que se segue é uma pequena extensão da definição na págin 397

188 Definição

Seja $\langle \mathbb{G}, g, r \rangle$ *um grupo Diffie-Hellman e* G' *e* Γ *dois grupos cíclicos com a mesma ordem prima* r. *Seja* $\psi : \mathbb{G} \to G'$ *um isomorfismo de grupos com a propriedade* $\psi(g) \neq 1$.

Uma função $\mathbf{e} : G \times G_1 \to \Gamma$ *é um* **emparelhamento ("pairing")** *se for PPT implementável e*

- *não degenerada no* 1^o *argumento: isto é,* $\mathbf{e}(g^a, q) = \mathbf{e}(g^b, q)$ *implica* $a = b$, *e*
- *bilinear: isto é,* $\mathbf{e}(g^a, q^b) = \mathbf{e}(g, q)^{a\,b}$,

para todos $q \neq 1 \in G'$ *e* $a, b \in \mathbb{Z}_r$.

Note-se que num grupo cíclico de ordem prima r, qualquer elemento $p \neq 1$ é gerador do grupo[61]. Desta forma, para garantir que $\mathbf{e}(\cdot, \cdot)$ é não-degenerado basta garantir que $\gamma = \mathbf{e}(g, \psi(g)) \neq 1$ [62].

[61] Qualquer subgrupo de um grupo cíclico G tem uma ordem que divide a ordem do grupo. Se a ordem de G é um número primo, os únicos subgrupos de G são o próprio G e o subgrupo trivial $\{1\}$ formado só pelo elemento 1. A órbita de p só pode, portanto, ser esse subgrupo trivial (quando $p = 1$) ou, quando $p \neq 1$, ser todo o grupo G.

[62] Se $q \neq 1$ existe $n \neq 0$ tal que $q = \psi(g)^n$; então $\mathbf{e}(g^a, q) = \gamma^{a\,n}$ e $\mathbf{e}(g^b, q) = \gamma^{b\,n}$; sendo $\gamma^{a\,n} = \gamma^{b\,n}$, dado que $\gamma \neq 1$ é gerador do grupo Γ e $n \neq 0$ não é divisor de zero em $\mathbb{Z}_r$, será $a = b$.

A existência de um grupo DH onde está definido uma função emparelhamente permite estabelecer imediatamente um oráculo DDHP da forma que vimos anteriormente

– Dados $p, g^a, p^b, p^c \in \mathbb{G}$ *, se for* $p \neq 1$*, o oráculo decide se* $a\,b = c$ *.*

$\mathrm{DDHP}(g^a, p^b, p^c)$

decide $\mathbf{e}(g^a, \psi(p^b)) \stackrel{?}{=} \mathbf{e}(g, \psi(p^c))$

Algoritmo 3: Oráculo DDHP baseado em emparelhamentos.

Nota: $\mathbf{e}(g^a, \psi(p^b)) = \mathbf{e}(g^{a\,b}, \psi(p))$; também $\mathbf{e}(g, \psi(p^c)) = \mathbf{e}(g^c, \psi(p))$; como tem de ser $\psi(p) \neq 1$ e $\mathbf{e}$ é não-degenerado, terá de ser $a\,b = c$.

O poder dos emparelhamentos vai mais além desde que algumas condições adicionais de computabilidade e segurança sejam colocadas no grupo cíclico Γ e na função ψ. Note-se que a definição de um grupo DH exige que as operação de grupo seja PPT implementável. Isto faz com que a exponenciação seja PPT implementável mas não, obrigatoriamente, o logaritmo discreto.

Como não se impõe nos grupos auxiliares G' e Γ outras restrições para além de serem cíclicos de ordem r, a definição de emparelhamento não dá quaisquer garantias quanto à dificuldade do logaritmo discreto nestes grupos

como não existem garantias quanto à facilidade com que se implementam as operações de grupo e a exponenciação. Essas garantias têm de ser colocadas explicitamente em condições adicionais de computabilidade e segurança.

Primeiro as **condições de computabilidade**.

189 DEFINIÇÃO
Seja $\langle \mathbb{G}, g, r \rangle$ *um grupo DH onde está definido, como expresso na definição 188, um emparelhamento. Se for PPT implementável a função* $\boldsymbol{\kappa} \colon \mathbb{G} \times \mathbb{G} \times \mathbb{Z}_r \longrightarrow \Gamma$

$$\boldsymbol{\kappa} \ : \ \langle q, p, x \rangle \quad \mapsto \quad \mathbf{e}(q, \psi(p))^x \qquad com \quad q, p \in \mathbb{G} \ , \ x \in \mathbb{Z}_r \tag{125}$$

o grupo diz-se ***bilinear DH***.

O problema que está na base das várias aplicações criptográficas dos emparelhamentos é o já referido protocolo de Joux para o acordo tripartido de chaves. Recordemos, que o protocolo baseava-se em três agentes A, B e C, que são titulares de chaves privadas a, b e c, e que procuram acordar numa chave comum κ

A chave acordada, dita P3K ("pairings triple key"), é determinado por um $p \neq 1 \in \mathbb{G}$ e pelos três segredos $a, b, c \in \mathbb{Z}_r$. É um item da forma $\kappa = \gamma^{a\,b\,c}$, em que $\gamma = \mathbf{e}(g, \psi(p)) = \boldsymbol{\kappa}(g, p, 1)$, que é calculada de forma diferente consoante o conhecimento de cada agente.

- O agente C conhece c e conhece também $\langle g^a, p^b \rangle$ ou $\langle g^b, p^a \rangle$

$C: \kappa \leftarrow \boldsymbol{\kappa}(g^a, p^b, c)$ ou $C: \kappa \leftarrow \boldsymbol{\kappa}(g^b, p^a, c)$

- O agente B conhece b e também $\langle g^a, p^c \rangle$ ou $\langle g^c, p^a \rangle$

$B: \kappa \leftarrow \boldsymbol{\kappa}(g^a, p^c, b)$ ou $B: \kappa \leftarrow \boldsymbol{\kappa}(g^c, p^a, b)$

- O agente A conhece a e conhece $\langle g^b, p^c \rangle$ ou $\langle g^c, p^b \rangle$

$A: \kappa \leftarrow \boldsymbol{\kappa}(g^b, p^c, a)$ ou $A: \kappa \leftarrow \boldsymbol{\kappa}(g^c, p^b, a)$

As diferentes técnicas criptográficas assentes no cálculo de chaves P3K baseiam a sua correcção na capacidade de calcular a mesma chave de forma diferente por vários agentes consoante os segredos que possuem. Por exemplo numa cifra, a chave será calculada na cifragem com a identidade do destinatário e com um segredo de sessão aleatório e, na de-cifragem, com a chave privada e com a exponenciação do segredo de sessão.

A propriedade algébrica que permite assegurar essa correcção, e que deriva imediatamente da definição (125), é

$$\boldsymbol{\kappa}(g^a, p^b, c) \;=\; \boldsymbol{\kappa}(g^x, p^y, z) \qquad \text{sse} \qquad a\,b\,c \;=\; x\,y\,z \tag{126}$$

Uma consequência desta propriedade, que por si só, é suficiente para assegurar a correcção é

190 Facto
O valor de $\boldsymbol{\kappa}(g^a, p^b, c)$ *é invariante em relação a qualquer permutação que for feita no tuplo* $\langle a, b, c\rangle$.

□

O cálculo da chave P3K é possível quando o agente que a calcula conhece um dos segredos (a, b ou c). Note-se que a função $\boldsymbol{\kappa}$ toma dois argumentos no grupo $\mathbb{G}$ mas um terceiro argumento no "domínio dos segredos" $\mathbb{Z}_r$. As **condições de segurança** que iremos estabelecer impõe que esta condição, para além de suficiente, tem de ser necessária; isto é, não deve ser possível calcular a chave κ sem conhecer um dos segredos.

Assim, seja $\langle \mathbb{G}, g, r\rangle$ um grupo bilinear DH (segundo a definição 188 e a definição 189); neste contexto existem três problemas que permitem exprimir a segurança.

Problema da Computação DH Bilinear (BCDHP)
Dados $p \neq 1$ *e pares* $\langle g^a, p^a\rangle, \langle g^b, p^b\rangle, \langle g^c, p^c\rangle$, *determinar* $\boldsymbol{\kappa}(g^a, p^b, c)$.

Este problema tem uma versão sob forma de decisão

Problema da Decisão DH Bilinear (BDDHP)
Dados $p \neq 1$ *e pares* $\langle g^a, p^a\rangle, \langle g^b, p^b\rangle, \langle g^c, p^c\rangle$, *distinguir* $\boldsymbol{\kappa}(g^a, p^b, c)$ *de* $\boldsymbol{\kappa}(g^a, p^b, x)$, *com* $x \in \mathbb{Z}_r$ *aleatório.*

Uma variante de BCDHP considera um número finito de elementos de $\mathbb{G}$ gerados pelo expoente desconhecido a a partir de bases g e p. Para cada função $f : \mathbb{Z}_r \longrightarrow \mathbb{Z}_r$ e limite n define-se

Problema da Computação DH Bilinear f, n (f, n-BCDHP)

Dados n *pares* $\langle g^{a^i}, p^{a^i} \rangle \in \mathbb{G}^2$, *com* $i = 0 \ldots n-1$, *determinar* $\boldsymbol{\kappa}(g, p, f(a))$.

□

Como exemplo de aplicação imediata destas noções vamos considerar o esquema de cifra de Boneh & Franklin (2001) que veio reactivar o moderno interesse em IBC.

Os seguintes elementos ocorrem em todos os protocolos derivados do protocolo básico de Boneh& Franklin.

Protocolo 35 : Elementos Comuns em IBC do tipo Boneh & Franklin

(1) O espaço dos textos $\mathbb{B}^t$.
(2) Um grupo de DH bilinear $\langle \mathbb{G}, g, r \rangle$ onde é BCDHP é intratável.
(3) Uma funçao de *hash* para representação de identidade: $\mathrm{ID} : \mathbb{B}^* \longrightarrow \mathbb{G} \setminus 1$.
(4) Funções de *hash*: $H : \mathbb{B}^* \longrightarrow \mathbb{B}^t$ e $H_r : \mathbb{B}^* \longrightarrow \mathbb{Z}_r$.
(5) Função *hash* de conversão $f : \Gamma \longrightarrow \mathbb{B}^t$.

Os **agentes** em causa são o "agente de confiança" T, o "encryptor" $\mathcal{E}$ e "decryptor" $\mathcal{D}$. O **objectivo** de $\mathcal{E}$ é cifrar um texto m (limitado a t bits) usando a identidade $\mathcal{D}$ como chave pública.

Protocolo 36 : Esquema IBE de Boneh & Franklin

"Setup" TA gera a sua chave privada s e respectiva chave pública $\beta = g^s$ que é certificada externamente

(1) $T\colon s \leftarrow \mathbb{Z}_r^* \quad ; \quad s \rightarrow \beta = g^s$

"Extract" TA determina a representação da identidade do "decryptor" e fornece-lhe a respectiva chave privada por canal fechado; $\mathcal{D}$ autentica a chave que recebe usando o oráculo DDHP gerado pelo emparelhamento.

(1) $T\colon d \leftarrow \mathsf{ID}(\mathcal{D}) \quad ; \quad \lambda \leftarrow d^s$

(2) $\mathcal{D}\colon \lambda \leftarrow T.\lambda \quad ; \quad d \leftarrow \mathsf{ID}(\mathcal{D}) \quad ; \quad \mathrm{DDHP}(\beta, d, \lambda) \stackrel{?}{=} 1$

Cifra/"Encryption" O texto $m \in \mathbb{B}^t$ é cifrado com a identidade $\mathcal{D}$ e produz o criptograma σ

(1) $\mathcal{E}\colon \upsilon \leftarrow \mathbb{B}^t \quad ; \quad a \leftarrow H_r(\upsilon \| m)$

(2) $\mathcal{E}\colon d \leftarrow \mathsf{ID}(\mathcal{D}) \quad ; \quad \mu \leftarrow \boldsymbol{\kappa}(\beta, d, a)$

(3) $\mathcal{E}\colon a, \upsilon, \mu, m \rightarrow \sigma = g^a \,\|\, \upsilon \oplus f(\mu) \,\|\, m \oplus H(\upsilon)$

Decifra/"Decryption" Com a chave privada λ e o criptograma $\sigma = \gamma \| \upsilon' \| m'$ recupera-se o texto m

(1) $\mathcal{D}\colon \gamma, \upsilon', m' \leftarrow \sigma \quad ; \quad \mu \leftarrow \boldsymbol{\kappa}(\gamma, \lambda, 1)$

(2) $\mathcal{D}\colon \upsilon \leftarrow \upsilon' \oplus f(\mu) \quad ; \quad m \leftarrow m' \oplus H(\upsilon)$

(3) $\mathcal{D}\colon a \leftarrow H_r(\upsilon \| m) \quad ; \quad \gamma \stackrel{?}{=} g^a$

A correcção deste protocolo assenta na constação de que o valor μ usado no esquema de cifra coincide com o valor

μ usado no esquema de decifra. Na cifra usa-se o triplo de valores $\langle g^s, d^1, a\rangle$; no esquema de decifra usa-se os valores $\langle g^a, d^s, 1\rangle$. Ambos derivam de permutações do mesmo triplo de "segredos" $\langle s, a, 1\rangle$; portanto o valor de μ é o mesmo em ambos os casos.

Note-se que o último passo no esquema de decifra existe apenas como confirmação de que o protocolo foi correctamente executado. O esquema de decifra permite recuperar todos os segredos (excepto a chave privada) gerados pelo esquema de cifra: o valor aleatório υ e a sua projecção a no domínio $\mathbb{Z}_r$ e o texto m; o teste $\gamma =? g^a$ permite verificar se o valor recuperado a coincide com o valor que foi usado para gerar γ no criptograma.

□

A segurança deste protocolo assenta na intratabilidade do BCDHP (na incapacidade de recuperar μ do conhecimento de $\beta = g^s$, $\gamma = g^a$ e d^1 apenas) mas, crucialmente, da forma como as quatro funções de *hash* usadas (ID, H, H_r e f) preservam, no *output*, a aleatoriedade do *input*.

Um modelo de segurança, onde se assume que todas as funções de *hash*, quando sujeitas a *inputs* aleatórios, se comportam como geradores aleatórios, designa-se por **modelo de oráculo aleatório ("random oracle model" ou ROM)**.

No ROM é relativamente fácil demonstrar a segurança do protocolo acima. No entanto a adopção de um tal modelo é sempre questionável; a alternativa seria concretizar as funções de *hash* em esquemas computacionais específicos e estudar o comportamento do esquema IBC com essas concretizações.

Um modelo de segurança que assume apenas a intratabilidade computacional dos diferentes problemas (CDHP, BCDHP, DLP, etc.) e não impõe quaisquer assumpções adicionais, designa-se por **modelo standard**.

A presença de muitas funções de *hash* abstractas levanta a impossibilidade de aplicar o modelo standard nas provas de segurança deste protocolo. Idealmente um esquema deveria reduzir ao mínimo o número dessas funções para tentar aproximar, tanto quanto possível, o seu modelo de segurança do modelo standard.

Por este motivo o protocolo 36 não é facilmente analisável num modelo standard de segurança.

□

Um outro aspecto a ter em conta, na avaliação de esquemas e protocolos em IBC, é a forma como as chaves privadas são geradas e distribuídas.

Em todos os protocolos IBC vistos até ao momento, as chaves privadas são gerados por um "trusted agent" que, posteriormente, as distribui aos seus titulares usando canais privados. Isto significa que, numa comunidade de agentes dependente de um único TA, existe um agente privilegiado que detém o conhecimento de todas as chaves privadas: ele possui um **depósito em confiança** (**"escrow"**) das chaves privadas de toda a comunidade.

Com esse depósito, o TA pode definir as políticas que entender para a distribuição das chaves; nomeadamente ele pode (por força ou por ser vítima de ataque) revelar uma chave privada a alguém que não é titular da mesma. Esta

situação é, obviamente, indesejável; a comunidade de agentes pode achar que um tal poder não pode ser depositado num único agente.

Por isso as técnicas IBC são também avaliadas pelo grau de **"escrow-freeness"** que apresentam. Por exemplo, para evitar a dependência em relação ao depósito de confiança, um protocolo pode distribuir a geração de chaves por várias fontes de confiança de forma a que nenhuma delas, individualmente, seja capaz de reconstruir qualquer chave.

Claramente o protocolo 36 avalia mal, não só em termos de modelo standard de segurança, mas também em "escrow-freeness". Veremos em seguida algumas alternativas.

□

Um terceiro aspecto diz respeito à forma como a identidade de um agente, expressa genericamente numa string de bits, é convertida em um objecto de um dos domínios que fazem parte dos elementos comuns da técnica criptográfica. Esse objecto designamos por **representante** da identidade.

Esta questão está associada, normalmente, à forma como a chave privada de um agente é gerada a partir da informação privada das fontes de confiança e do representante da identidade.

No contexto de técnicas baseadas em grupos DH, todos os protocolos apresentados até ao momento (nomeadamente o IBE de Boneh & Franklin) encaram estes dois aspectos sempre da mesma forma:

- Cada identidade $I \in \mathbb{B}^*$ é representada por um elemento p do grupo cíclico $\mathbb{G} \setminus 1$. Esse representante constrói-se como $p = \mathsf{ID}(I)$ sendo $\mathsf{ID}: \mathbb{B}^* \to \mathbb{G} \setminus 1$ uma função de *hash*.
- Existe uma única fonte de confiança TA que tem uma chave privada no "domínio dos segredos" $s \in \mathbb{Z}_r$. A chave pública correspondente é da forma $\beta = g^s$.
- A chave privada λ_I do agente titular da idendidade I é dada por $\lambda_I = p^s$.

Este mecanismo concretiza-se nas fases "setup" e "extract" desta família de técnicas

Protocolo 37 : Geração de Chaves do Tipo I (Boneh & Franklin)

"Setup" O TA gera a sua chave privada s e publicita a respectiva chave pública $\beta = g^s$

(1) $T: s \leftarrow \mathbb{Z}_r \quad ; \quad s \to \beta = g^s$

"Extract" O TA calcula a chave privada de I e envia-a por canal privado para o seu titular.

(1) $T: p \leftarrow \mathsf{ID}(I) \quad ; \quad \lambda \leftarrow p^s$

(2) $I: \lambda \leftarrow T.\lambda$ *– o agente I recolhe o conhecimento λ de T por canal privado*

Esta construção permite uma versão "escrow-free". Para isso considera-se que o TA está distribuído por várias fontes de confiança $T_1, T_2, \ldots, T_d$ cada uma com a sua chave privada $s_1, s_2, \ldots, s_n$.

Cada T_k vai proceder exactamente como se fosse um único TA no esquema anterior. No entanto, individualmente, cada um desses agentes produz apenas parte da chave pública e parte da chave privada. Um utilizador (da chave

pública ou da chave privada) tem de as reconstruir a chave que necessita a partir das várias componentes fornecidas pelas fontes.

Protocolo 38 : Geração de Chaves do Tipo I (versão "escrow-free")

"Setup" Cada T_k gera a sua chave privada s_k e publicita a respectiva componente da chave pública $\beta_k = g^{s_k}$

(1) T_k: $s_k \leftarrow \mathbb{Z}_r$; $s_k \rightarrow \beta_k = g^{s_k}$ – *para todo* $k = 1 \ldots d$

"Extract" Cada T_k calcula um factor λ_k da chave privada λ

(1) T_k: $p \leftarrow \mathsf{ID}(I)$; $\lambda_k \leftarrow p^{s_k}$ – *para todo* $k = 1 \ldots d$

(2) I: $\lambda_k \leftarrow T_k.\lambda_k$ – *para todo* $k = 1 \ldots d$

Construção de chaves Usando uma "máscara" $\overline{b} = b_1 \| b_2 \| \ldots \| b_d \in \mathbb{Z}_r^d$, o público $*$ reconstrói a chave pública β e I reconstrói a chave privada λ.

(1) $*$: $\beta \leftarrow (\beta_1)^{b_1} \ (\beta_2)^{b_2} \ \ldots \ (\beta_d)^{b_d}$

(2) I: $\lambda \leftarrow (\lambda_1)^{b_1} \ (\lambda_2)^{b_2} \ \ldots \ (\lambda_d)^{b_d}$

Se definirmos $s = \sum_{k=1}^{d} b_k \, s_k$, vê-se facilmente que $\beta = g^s$ e que $\lambda = p^s$. Portanto este esquema comporta-se exactamente como o esquema original de Boneh & Franklin com a diferença de que a "chave privada" s nunca chega a existir; é, de certa forma, "virtual".

Cada uma das fontes de confiança T_k conhece uma parte de s mas não conhece a chave toda. Desta forma, a

menos que exista um conluio completo entre as várias fontes de confiança, nenhum dos T_k individualmente conhece a chave privada λ e, por isso, não a pode divulgar (mesmo que queira).

Adicionalmente, se existir evidencia que T_k foi atacado ou divulgou informação privilegiada, as respectivas componentes β_k e λ_k podem ser retiradas da construção das chaves β e λ preservando a relação entre essas chaves. Esta é a função da "máscara" $b_1, \ldots, b_d$: escolher as fontes de confiança que fornecem a informação usada na reconstrução de β e λ e definir um "peso" para cada uma dessas componentes.

Um protocolo que esconda a máscara $\overline{b}$ das fontes de confiança, é "escrow-free" e "anti-conluio"; isto porque, mesmo que as fontes estejam em conluio completo, não podem prever a máscara e, assim, não podem reconstruir λ. Obviamente que um tal protocolo exige que a máscara $\overline{b}$ seja um segredo comum apenas dos intervenientes do protocolo e tal implica, normalmente, o uso prévio de um protocolo de acordo de chaves.

□

Uma abordagem alternativa mapeia a identidade I, não no grupo $\mathbb{G}$, mas no mesmo domínio $\mathbb{Z}_r$ que contém as chaves privadas das eventuais fontes de confiança. A função de *hash* usada tem agora a forma $\mathsf{ID}: \mathbb{B}^* \longrightarrow \mathbb{Z}_r$ e o representante é também construído como $p = \mathsf{ID}(I)$.

Este solução presta-se à construção de funções polinomiais $q(p, s)$ que "misturam" a identidade p com a chave privada s de um TA. Com tais funções, a exponenciais $g^{q(p,s)}$ podem ser reconstruídas a partir do conhecimento de p e das potências $\beta_0 = g, \beta_1 = g^s, \beta_2 = g^{s^2}$, etc.

Por exemplo, um polinómio

$$q(p,s) \;=\; p^2 + s\,p + s^2$$

permite construir um valor $\mu = g^{f(p,s)}$ como $\mu \;=\; g^{p^2}\;(g^s)^p\;g^{s^2}$. Se os valores $\beta_0 = g, \beta_1 = g^s$ e $\beta_2 = g^{s^2}$ constituírem a chave pública, o conhecimento de p permite o cálculo público de

$$\mu \;=\; g^{q(p,s)} \;=\; \beta_0^{p^2} \cdot \beta_1^p \cdot \beta_2$$

Esta abordagem está inerente ao segundo tipo de IBC baseado em grupos Diffie-Helmman bilineares. Os elementos comuns são, essencialmente, os mesmos que estão descritos no protocolo 35. A única diferença é a função de *hash* ID que, aqui, tem $\mathbb{Z}_r$ como contradomínio.

Protocolo 39 : Elementos Comuns em IBC do tipo Sakai & Kasahara

(1) O espaço dos textos $\mathbb{B}^t$.
(2) Um grupo de DH bilinear $\langle \mathbb{G}, g, r \rangle$ onde é BCDHP é intratável.
(3) Uma funçao de *hash* para representação de identidade: $\mathrm{ID} : \mathbb{B}^* \longrightarrow \mathbb{Z}_r$
(4) Funções de *hash*: $H : \mathbb{B}^* \longrightarrow \mathbb{B}^t$ e $H_r : \mathbb{B}^* \longrightarrow \mathbb{Z}_r$.
(5) Função *hash* de conversão $f : \Gamma \longrightarrow \mathbb{B}^t$.

A geração de chaves que vamos apresentar mistura o segredo s e a identidade p num monómio $(p + s)$.

Protocolo 40 : Geração de Chaves do Tipo II (Sakai & Kasahara)

"Setup" Como no tipo I

(1) $T\colon s \leftarrow \mathbb{Z}_r^* \quad ; \quad s \rightarrow \beta = g^s$

"Extract" A chave privada λ é gerada e comunicada por canal privado.

(1) $T\colon p \leftarrow \mathsf{ID}(I) \quad ; \quad z \leftarrow (p+s)^{-1} \quad ; \quad \lambda \leftarrow g^z$

(2) $I\colon \lambda \leftarrow T.\lambda$

Protocolo 41 : Esquema IBE (Sakai & Kasahara)

"Setup&Extract" Geração de chaves do tipo II

Cifra/ "Encryption" O texto $m \in \mathbb{B}^t$ é cifrado com a identidade $\mathcal{D}$ e produz o criptograma σ

(1) $\mathcal{E}\colon \upsilon \leftarrow \mathbb{B}^t \quad ; \quad a \leftarrow H_r(\upsilon \| m) \quad ; \quad \mu \leftarrow \boldsymbol{\kappa}(g,g,a)$

(2) $\mathcal{E}\colon d \leftarrow \mathsf{ID}(\mathcal{D}) \quad ; \quad \gamma \leftarrow (\beta\, g^d)^a$

(3) $\mathcal{E}\colon \upsilon, \mu, m \rightarrow \sigma = \gamma \,\|\, \upsilon \oplus f(\mu) \,\|\, m \oplus H(\upsilon)$

Decifra/ "Decryption" Com a chave privada λ e o criptograma $\sigma = \gamma \| \upsilon' \| m'$ recupera-se o texto m

(1) $\mathcal{D}\colon \gamma, \upsilon', m' \leftarrow \sigma \quad ; \quad \mu \leftarrow \boldsymbol{\kappa}(\gamma, \lambda, 1)$

(2) $\mathcal{D}\colon \upsilon \leftarrow \upsilon' \oplus f(\mu) \quad ; \quad m \leftarrow m' \oplus H(\upsilon)$ – *recuperação*

(3) $\mathcal{D}\colon a \leftarrow H_r(\upsilon \| m) \quad ; \quad d \leftarrow \mathsf{ID}(\mathcal{D}) \quad ; \quad \gamma \stackrel{?}{=} (\beta\, g^d)^a$ – *verificação*

O protocolo de IBE originário de Sakai & Kasahara (2003) que é essencialmente o protocolo de Boneh & Franklin adaptado a esta configuração de chaves. Os agentes e o objectivo são os mesmos do protocolo 36 de Boneh & Franklin.

A correcção deriva do facto de ser $\gamma = g^{(s+d)\,a}$, de ser $\lambda = g^{(s+d)^{-1}}$ e ser $\mu = \boldsymbol{\kappa}(g, g, a) = \boldsymbol{\kappa}(g^{(s+d)\,a}, g^{(s+d)^{-1}}, 1)$. A segurança deriva do facto de ser intratável determinar $g^{(s+d)^{-1}}$ mesmo que se conheça g^s e g^d.

Este esquema pode ser generalizado, substituindo o monómio $(s+d)$ por outro qualquer polinómio $q(s,d)$. Para isso a chave pública é agora formada pelos vários $\beta_i = g^{s^i}$, com i desde 0 até ao grau do polinómio. A chave privada seria $\lambda = g^{q(s,d)^{-1}}$ e o cálculo de γ reconstrói $g^{a\,p(s,d)}$ a partir dos vários β_i e das potências g^{d^i}.

□

Outra generalização substitui $\boldsymbol{\kappa}$, apresentada em (126), por uma construção algébrica chamada **co-emparelhamento**.

191 DEFINIÇÃO
Sejam $\langle \mathbb{G}, g, r\rangle$ *e* $\langle \Gamma, h, r\rangle$ *dois grupos DH com a mesma ordem prima* r. *Um* **co-emparelhamento** *é uma função PPT implementável* $\mathbf{c}\colon \mathbb{G} \times \Gamma \to \Gamma$ *que é não-degenerada (* $\mathbf{c}(g,h) \neq 1$ *) e bilinear (* $\mathbf{c}(g^a, \gamma) = \gamma^a$, *para todo* $a \in \mathbb{Z}_r$ *e* $\gamma \in \Gamma$ *).*

Um co-emparelhamento produz, imediatamente, um oráculo DDHP.

– Dados $g^a, p^b, p^c \in \mathbb{G}$, se for $p \neq 1$, o oráculo decide se $a\,b = c$.

$\mathrm{DDHP}(g^a, p^b, p^c)$

$\gamma \leftarrow \mathbf{c}(g^a, h)$
decide $\mathbf{c}(p^c, h) \;=?=\; \mathbf{c}(p^b, \gamma)$

Algoritmo 4: Oráculo DDHP baseado em co-emparelhamentos.

No esquema de geração de chaves tipo II (protocolo 40), λ passa a ser um elemento do grupo Γ.

Protocolo 42 : Geração de Chaves tipo II - variante co-emparelhamento

"Setup" Como no tipo II, protocolo 40

"Extract" A chave privada λ é gerada e comunicada por canal privado.

(1) $T: d \leftarrow \mathrm{ID}(I) \quad ; \quad z \leftarrow (s+d)^{-1} \quad ; \quad \lambda \leftarrow h^z$

(2) $I: \lambda \leftarrow T.\lambda$

No esquema IBE de decifra, μ passa a ser gerado pela função co-emparelhamento $\mathbf{c}$.

Protocolo 43 : Esquema IBE (Sakai & Kasahara) - variante co-emparelhamento

"Setup&Extract" protocolo 42

Cifra/ "Encryption" O texto $m \in \mathbb{B}^t$ é cifrado com a identidade $\mathcal{D}$ e produz o criptograma σ

(1) $\mathcal{E}: \upsilon \leftarrow \mathbb{B}^t \quad ; \quad a \leftarrow H_r(\upsilon \| m) \quad ; \quad \mu \leftarrow h^a$

(2) $\mathcal{E}: d \leftarrow \mathsf{ID}(\mathcal{D}) \quad ; \quad \gamma \leftarrow (\beta\, g^d)^a$

(3) $\mathcal{E}: \upsilon, \mu \;\rightarrow\; \sigma = \gamma \,\|\, \upsilon \oplus f(\mu) \,\|\, m \oplus H(\upsilon)$

Decifra/"Decryption" Com a chave privada λ e o criptograma $\sigma = \gamma \| \upsilon' \| m'$ recupera-se o texto m

(1) $\mathcal{D}: \gamma, \upsilon', m' \leftarrow \sigma \quad ; \quad \mu \leftarrow \mathbf{c}(\gamma, \lambda)$

(2) $\mathcal{D}: \upsilon \leftarrow \upsilon' \oplus f(\mu) \quad ; \quad m \leftarrow m' \oplus H(\upsilon)$ – *recuperação*

(3) $\mathcal{D}: a \leftarrow H_r(\upsilon \| m) \quad ; \quad \gamma \stackrel{?}{=} (\beta\, g^d)^a$ – *verificação*

Justificação A correcção desta versão do esquema deriva da bilinearidade da função de co-emparelhamento:

$$\mu \;=\; \mathbf{c}(\gamma, \lambda) \;=\; \mathbf{c}(g^{a\,(s+d)}, h^{(s+d)^{-1}}) \;=\; h^a$$

□

As cifras assimétricas têm uma complexidade computacional que torna inviável usá-las com mensagens grandes. Por

isso são usadas quase só em mensagens muito curtas; tipicamente, em chaves.

Essas chaves, assim cifradas, são enviadas a um destinatário e constituiem, desta forma, segredos que o gerador e o destinatário partilham. Finalmente as chaves partilhadas podem ser usadas numa cifra simétrica que, por ser computacionalmente muito eficiente, tem capacidade para processar mensagens de tamanho arbitrário.

Este mecanismo é designado por **cifragem híbrida**. Tradicionalmente o uso de uma cifra híbrida por um agente E, que quer enviar a um outro agente D uma mensagem m cifrada, assume a existência de:

- Uma cifra assimétrica (**Cifra** e **Decifra**) e uma cifra simétrica (**SimCifra** e **SimDecifra**).
- Uma par de chaves (ek, dk), pública e privada, para a cifra assimétrica.

Protocolo 44 : Cifra Híbrida

Cifra Cifrar um texto m de tamanho arbitrário

(1) E gera uma chave aleatória k, determina $\sigma = \textbf{Cifra}(k, \text{ek})$ e torna público este valor

(2) E usa k para cifrar o texto m com a cifra simétrica: faz $y = \textbf{SimCifra}(m.k)$ e torna-o público.

Decifra D recupera a chave k e o texto m

(1) D recupera k com a chave privada: $k \leftarrow \textbf{Decifra}(\sigma, \text{dk})$.

(2) D recupera, com k, o texto m: $m \leftarrow \textbf{SimDecifra}(y, k)$.

A análise de segurança deste protocolo básico mostra-se mais simples que separar-mos as primitivas (1) da cifragem e decifragem num único esquema e juntarmos as primitivas (2) destes mesmo passos num segundo o esquema.

Esta abordagem designa-se por **KEM-DEM**.

A primeira componente, o **Key Encapsulation Mechanism (KEM)**, corresponde *grosso modo*, ao passo de geração do par de chaves $(\mathsf{ek}, \mathsf{dk})$, ao passo de geração do segredo k e a sua cifra ("encapsulamento") com a chave pública ek, e ao passo de extracção ("desencapsulamento") de k usando a chave privada dk.

Na segunda componente intervém o texto m (os "dados"); designa-se por **Data Encapsulation Mechanism (DEM)** e é descrito pela cifragem do texto m (e sua recuperação) usando, num esquema de cifra/decifra simétrica, a chave k gerada e recuperada pelo KEM.

Uma característica importante dos esquemas **KEM** actuais é que a **geração** da chave k e o seu **encapsulamento**, é feita globalmente num único passo lógico. Isto contrasta com o esquema clássico das cifras híbridas onde a geração de k e a sua cifragem são algoritmos separados.

Outra característica importante reside no facto de a componente **DEM** poder ser completamente independente da componente KEM. Nomeadamente a segurança da DEM é completamente independente da segurança da KEM; o único ponto que têm em comum é a chave k cujo tamanho força alguma dependência entre as duas componentes. Por isso, e dado que a segurança do DEM é essencialmente a de uma cifra simétrica, pode-se abstrair o estudo desta técnica só ao estudo do KEM.

Protocolo 45 : Modelo KEM-DEM

- O **Key Encapsulation Mechanism (KEM)** é formado por 3 algoritmos:

Geração

Um algoritmo probabilístico $\mathcal{K}$ que gera um par de chaves pública e privada $(\mathsf{ek}\|\mathsf{dk})$.

Encapsulamento

Um algoritmo probabilístico $\mathcal{E}$ que recebe como input a chave pública ek e gera $(k\|\sigma)$ formado por um segredo k e o seu "encapsulamento" σ.

De-encapsulamento

Um algoritmo determinístico $\mathcal{D}$ que recebe $\sigma\|\mathsf{dk}$ como input e gera k ou então falha.

- O **Data Encapsulation Mechanism (DEM)** é formado por 2 algoritmos:

Cifra

Um algoritmo probabilístico $\mathcal{E}_{\mathsf{sym}}$ que recebe como input $k\|m$ (uma chave k e um texto m de comprimento arbitrário) e gera um criptograma y.

Decifra

Um algoritmo determinístico $\mathcal{D}_{\mathsf{sym}}$ que recebe como input $k\|y$ (uma chave k e um criptograma de comprimento arbitrário y) e gera o texto m ou, então, falha.

A correcção do KEM exprime a propriedade de a chave k, gerada e encapsulada com a chave pública ek, ser a

mesma que se recupera do encapsulamento σ e da chave privada dk. Isto é

Para todo par de chaves $(\mathsf{ek}, \mathsf{dk})$ *, são indistinguíveis as variáveis aleatórias* k *e* k' *geradas pelos algoritmos*

$$k \| \sigma \leftarrow \mathcal{E}(\mathsf{ek}) \qquad , \qquad k' \leftarrow \mathcal{D}(\sigma \| \mathsf{dk})$$

De forma análoga a correcção do DEM exprime a capacidade de recuperar o texto m do criptograma y

Para toda a chave k *, são indistinguíveis as variáveis aleatórias* m *e* m' *determinada pelo algoritmo*

$$y \leftarrow \mathcal{E}_{\mathsf{sym}}(k \| m) \ ; \ m' \leftarrow \mathcal{D}_{\mathsf{sym}}(k \| y)$$

□

Obviamente é possível criar KEM´s usando identidades como chaves públicas. Por exemplo, o esquema IBE de Sakai-Kasahara (protocolo 41) pode ser adaptado a um esquema KEM.

Protocolo 46 : Esquema KEM orientado à identidade (inspirado no IBE de Sakai-Kasahara)

Setup & Extract os elementos comuns descritos no protocolo 39 e a geração de chaves do tipo II descrita no protocolo 40

Encapsulamento com a identidade $\mathcal{D}$, gera o segredo k e o seu encapsulamento σ.

(1) $\mathcal{E}: \upsilon \leftarrow \mathbb{B}^t \quad ; \quad k \leftarrow H(\upsilon)$ – *geração da chave*

(2) $\mathcal{E}: d \leftarrow \mathrm{ID}(\mathcal{D}) \quad ; \quad a \leftarrow H_r(\upsilon) \quad ; \quad \gamma \leftarrow (\beta\, g^d)^a$ – *encapsulamento*

(3) $\mathcal{E}: \mu \leftarrow \boldsymbol{\kappa}(g, g, a) \quad ; \quad \upsilon, \mu \rightarrow \sigma = \gamma \,\|\, (\upsilon \oplus f(\mu))$ – *emissão*

De-encapsulamento Com a chave privada λ e o encapsulamento $\sigma = \gamma \| \upsilon'$ recupera-se a chave k

(1) $\mathcal{D}: \gamma, \upsilon' \leftarrow \sigma \quad ; \quad \mu \leftarrow \boldsymbol{\kappa}(\gamma, \lambda, 1) \quad ; \quad \upsilon \leftarrow \upsilon' \oplus f(\mu)$ – *de-encapsulamento*

(2) $\mathcal{D}: a \leftarrow H_r(\upsilon) \quad ; \quad \gamma \stackrel{?}{=} (\beta\, g^d)^a$ – *confirmação*

(3) $\mathcal{D}: k \leftarrow H(\upsilon)$ – *recuperação da chave*

É possível construir variantes "escrow-free" destes esquemas IBE ou KEM-DEM usando duas formas de abordagem: com ou sem certificados.

Se tomar-mos como base o esquema de geração de chaves do tipo I, recorde-se que $\mathcal{D}$ tinha uma chave privada $\lambda = p^s$, sendo $p = \mathrm{ID}(\mathcal{D})$ e s a chave privada do TA. Aqui o TA conhece a chave privada de $\mathcal{D}$

Uma solução alternativa consiste em dar a $\mathcal{D}$ um segredo próprio x (não conhecido do TA) e usar, como informação privada λ, algo que dependa de x, para além da componente p^s conhecida pelo TA.

Isto sugere os dois seguintes esquemas KEM, "escrow-free", com e sem certificados.

No primeiro a função ID tem a aridade $\mathbb{G} \times \mathbb{B}^* \longrightarrow \mathbb{G}$; no segundo essa função tem a aridade usual $\mathbb{B}^* \longrightarrow \mathbb{G}$. No esquema sem certificados, estes são substituídos por um mecanismo de auto-certificação.

Protocolo 47 : KEM baseado em certificados (Gentry)

"Setup" TA gera a sua chave privada e publica a chave pública correspondente.

(1) $T: s \leftarrow \mathbb{Z}_r^* \quad ; \quad s \rightarrow \beta = g^s$

"Extract& Certify" $\mathcal{D}$ escolhe o segredo próprio e publicita a chave pública correspondente. TA certifica a chave pública ζ e emite a 2ª componente da chave privada de $\mathcal{D}$.

(1) $\mathcal{D}: x \leftarrow \mathbb{Z}_r^* \quad ; \quad x \rightarrow \zeta = g^x$

(2) $T: p \leftarrow \mathsf{ID}(\zeta, \mathcal{D}) \quad ; \quad z \leftarrow p^s$

(3) $\mathcal{D}: p \leftarrow \mathsf{ID}(\zeta, \mathcal{D}) \quad ; \quad \lambda \leftarrow (T.z) \cdot p^x$

Encapsulamento $\mathcal{E}$ gera a chave k e o seu encapsulamento σ.

(1) $\mathcal{E}: \upsilon \leftarrow \mathbb{B}^t \quad ; \quad k \leftarrow H(\upsilon)$ – *"keygen"*

(2) $\mathcal{E}: p \leftarrow \mathsf{ID}(\zeta, \mathcal{D}) \quad ; \quad a \leftarrow H_r(\upsilon) \quad ; \quad \gamma \leftarrow g^a$ – *redundância*

(3) $\mathcal{E}: \mu \leftarrow \boldsymbol{\kappa}(\beta, p, a) \cdot \boldsymbol{\kappa}(\zeta, p, a) \quad ; \quad \mu, \upsilon \rightarrow \sigma = \gamma \,\|\, \upsilon \oplus f(\mu)$ – *encapsulamento*

De-emcapsulamento $\mathcal{D}$ recupera k e verifica a sua autenticidade

(1) $\mathcal{D}: \gamma, y \leftarrow \sigma \quad ; \quad \mu \leftarrow \boldsymbol{\kappa}(\gamma, \lambda, 1) \quad ; \quad \upsilon \leftarrow y \oplus f(\mu) \quad ; \quad k \leftarrow H(\upsilon)$ – *extracção*

(2) $\mathcal{D}: a \leftarrow H_r(\upsilon) \quad ; \quad \gamma \stackrel{?}{=} g^a$ – *verificação*

Protocolo 48 : KEM sem certificados (Al-Riyami & Paterson)

"Setup" Como no protocolo 47

"Extract" $\mathcal{D}$ gera um segredo x, publica a respectiva informação pública ζ (que é auto-certificável) e extrai do TA a informação necessário ao cálculo da chave privada λ.

(1) $\mathcal{D}\colon x \leftarrow \mathbb{Z}_r^* \;\; ; \;\; x \rightarrow \zeta = g^x \,\|\, \beta^x$

(2) $T\colon p \leftarrow \mathrm{ID}(\mathcal{D}) \;\; ; \;\; z \leftarrow p^s$

(3) $\mathcal{D}\colon \lambda \leftarrow (T.z)^x$

Encapsulamento o emissor $\mathcal{E}$ gera a chave k e o seu encapsulamento σ

(1) $\mathcal{E}\colon q, t \leftarrow \zeta \;\; ; \;\; \mathrm{DDHP}(\beta, q, t) \stackrel{?}{=} 1$ – *certificação*

(2) $\mathcal{E}\colon \upsilon \leftarrow \mathbb{B}^t \;\; ; \;\; a \leftarrow H_r(\upsilon) \;\; ; \;\; \gamma \leftarrow g^a \;\; ; \;\; k \leftarrow H(\upsilon)$ – *redundância & "keygen"*

(3) $\mathcal{E}\colon p \leftarrow \mathrm{ID}(\mathcal{D}) \;\; ; \;\; \mu \leftarrow \boldsymbol{\kappa}(t, p, a) \;\; ; \;\; \upsilon, \mu \rightarrow \sigma = \gamma \,\|\, \upsilon \oplus f(\mu)$ – *encapsulamento*

De-encapsulamento $\mathcal{D}$ recupera k e verifica a sua autenticidade

(1) $\mathcal{D}\colon \gamma, y \leftarrow \sigma \;\; ; \;\; \mu \leftarrow \boldsymbol{\kappa}(\gamma, \lambda, 1) \;\; ; \;\; \upsilon \leftarrow y \oplus f(\mu) \;\; ; \;\; k \leftarrow H(\upsilon)$ – *extracção*

(2) $\mathcal{D}\colon a \leftarrow H_r(\upsilon) \;\; ; \;\; \gamma \stackrel{?}{=} g^a$ – *verificação*

No protocolo com certificados, a chave privada do agente $\mathcal{D}$ é $\lambda = p^{(s+x)}$. No de-emcapsulamento, $\mu = \boldsymbol{\kappa}(\gamma, \lambda, 1) = \boldsymbol{\kappa}(g^a, p^{(s+x)}, 1) = \boldsymbol{\kappa}(g, p, a\,(s+x))$. No encapsulamento tem-se $\mu = \boldsymbol{\kappa}(g^s, p, a) \cdot \boldsymbol{\kappa}(g^x, p, a) = \boldsymbol{\kappa}(g, p, (s+x)\,a)$.

No protocolo sem certificados, a chave privada é $\lambda = p^{s\,x}$. No encapsulamento $\mu = \boldsymbol{\kappa}(t, p, a) = \boldsymbol{\kappa}(g, p, s\,x\,a)$; no de-emcapsulamento $\mu = \boldsymbol{\kappa}(\gamma, \lambda, 1) = \boldsymbol{\kappa}(g^a, p^{s\,x}, 1) = \boldsymbol{\kappa}(g, p, s\,x\,a)$.

6.5 IBA sobre Emparelhamentos

Emparelhamentos e co-emparelhamentos tornam viáveis vários protocolos associados à autenticação: protocolos de identificação, esquemas de assinaturas, protocolos de acordo de chaves, etc. Tornam ainda viáveis protocolos mais complexos para assinatura e cifra simultânea de mensagens.

O mais simples destes protocolos será um protocolo de identificação que usa o esquema de geração de chaves do tipo I (ver pag. 415).

Protocolo 49 : Identificação (emparelhamentos)

"Setup&Extract" Protocolo de geração do tipo I (protocolo 37) em que um "prover" $\mathcal{P}$ recolhe a sua chave privada $\lambda = p^s$ com $p = \mathrm{ID}(\mathcal{P})$. A chave pública do TA é $\beta = g^s$.

Desafio O "verifier" $\mathcal{V}$ envia um desafio aleatório d

(1) $\mathcal{V}: \upsilon \leftarrow \mathbb{Z}_r^* \quad ; \quad \upsilon \rightarrow d = g^{\upsilon}$.

Resposta O "prover" $\mathcal{P}$ determina a resposta μ

(1) $\mathcal{P}: \lambda \rightarrow \mu = \boldsymbol{\kappa}(d, \lambda, 1)$

Verificação O "verifier" verifica a correcção da resposta usando o mecanismo $\boldsymbol{\kappa}$ (definição 189, pag. 407)

(1) $\mathcal{V}: p \leftarrow \mathrm{ID}(\mathcal{P}) \quad ; \quad \boldsymbol{\kappa}(\beta, p, \upsilon) \ =?= \ \mu$

A correcção do protocolo deriva das propriedades da função $\boldsymbol{\kappa}$

$$\mu \;=\; \boldsymbol{\kappa}(d,\lambda,1) \;=\; \boldsymbol{\kappa}(g^{\upsilon},p^{s},1) \;=\; \boldsymbol{\kappa}(g^{s},p^{1},\upsilon) \;=\; \boldsymbol{\kappa}(\beta,p,\upsilon)$$

O mesmo protocolo pode ser realizado com co-emparelhamentos. Para isso é necessário alterar ligeiramente os elementos comuns e a geração de chaves.

Assim assume-se que existe uma função co-emparelhamento $\mathbf{c}\colon \mathbb{G}\times\mathbb{G}'\to\mathbb{G}'$, sendo $\mathbb{G},\mathbb{G}'$ dois grupos de Diffie-Hellman com a mesma ordem prima r. A função de *hash* par determinação do representante da identidade é da forma $\mathsf{ID}\colon \mathbb{B}^*\to\mathbb{G}'\setminus 1$. Tal como na geração de chaves do tipo I, o TA tem uma chave privada $s\in\mathbb{Z}_r^*$ e uma chave pública $\beta=g^s$ e m agente I, com representante de identidade $p=\mathsf{ID}(I)$, tem chave privada $\lambda=p^s$.

Protocolo 50 : Identificação (co-emparelhamentos)

Desafio O "verifier" $\mathcal{V}$ envia um desafio aleatório d

(1) $\mathcal{V}\colon \upsilon\leftarrow\mathbb{Z}_r^* \quad ; \quad \upsilon\to d=g^{\upsilon}.$

Resposta O "prover" $\mathcal{P}$ determina a resposta μ usando o co-emparelhamento

(1) $\mathcal{P}\colon \lambda\to\mu=\mathbf{c}(d,\lambda)$

Verificação O "verifier" verifica a correcção da resposta usando também o co-emparelhamento

(1) $\mathcal{V}\colon p\leftarrow\mathsf{ID}(\mathcal{P}) \quad ; \quad \mathbf{c}(\beta,p^{\upsilon}) \;=?=\; \mu$

A correcção do protocolo deriva da propriedade bilenear da função de co-emparelhamento: $\mathbf{c}(g^{\upsilon}, p^{s}) = p^{\upsilon\, s} = \mathbf{c}(g^{s}, p^{\upsilon})$.

□

Usando emparelhamentos (algoritmo 3) e co-emparelhamentos (algoritmo 4) é possível construir oráculos DDHP e, dessa forma, implementar assinaturas e protocolos de identificação que requeiram apenas a estrutura de um grupo Gap Diffie-Hellman. Estão nesta categoria as assinatura de Cha & Cheon (protocolo 31) e BLS (protocolo 33) e o esquema de identificação de Cha & Cheon (protocolo 32).

Outras tecnicas de autenticação requerem as proptiedades algébricas dos emparelhamentos. Um exemplo paradigmático é o esquema de assinatura de Hess baseado no "setup" e geração de chaves do tipo Boneh & Franklin.

Protocolo 51 : Esquema de assinaturas IBA de Hess

"Setup&Extract" Usa os elementos comuns e a geração de chaves do tipo I (protocolo 37). Usa também uma função de *hash* $H' \colon \Gamma \times \mathbb{B}^t \to \mathbb{Z}_r$.

Assinatura O "prover" $\mathcal{P}$ constrói uma assinatura σ para a mensagem $m \in \mathbb{B}^t$

(1) $\mathcal{P} \colon \upsilon \leftarrow \mathbb{Z}_r^* \quad ; \quad \mu \leftarrow \boldsymbol{\kappa}(\lambda, g, \upsilon) \quad ; \quad h \leftarrow H'(\mu, m) \quad ; \quad \upsilon, \lambda, h \to \sigma = h \| \lambda^{\upsilon - h}$

Verificação com a chave pública β do TA e a identidade do "prover", verifica a assinatura σ no texto m.

(1) $\mathcal{V} \colon h, \alpha \leftarrow \sigma \quad ; \quad p \leftarrow \mathsf{ID}(\mathcal{P}) \quad ; \quad \mu' \leftarrow \boldsymbol{\kappa}(\alpha, g, 1) \cdot \boldsymbol{\kappa}(p, \beta, h)$

(2) $\mathcal{V} \colon h \ =?= \ H'(\mu', m)$

Notas

1. A função H' pode ser gerada a partir dos elementos já existentes; por exemplo pode-se definir $H'(\mu, m) = H_r(f(\mu) \oplus m)$.
2. A correcção deriva das propriedades algébricas do emparelhamento.
   Temos $\lambda = p^s$ e $\beta = g^s$; na verificação tem-se $\alpha = \lambda^{\upsilon - h} = p^{s\,(\upsilon - h)}$; se representarmos por γ o valor $\boldsymbol{\kappa}(p, g, 1)$, tem-se
   $$\mu = \boldsymbol{\kappa}(\lambda, g, \upsilon) = \gamma^{s\,\upsilon} \qquad , \qquad \mu' = \boldsymbol{\kappa}(\alpha, g, 1) \cdot \boldsymbol{\kappa}(p, \beta, h) = \gamma^{s\,(\upsilon - h)} \cdot \gamma^{s\,h}$$
   Consequentemente o valor de μ calculado na assinatura coincide com o valor de μ' calculado na verificação.
3. Este esquema pode ser imediatamente adaptado a co-emparelhamentos $\mathbf{c} \colon \mathbb{G} \times \Gamma \to \Gamma$. Para isso a função de *hash* ID deve dar resultados em Γ e μ tem também um valor neste segundo grupo.
   Na assinatura, μ calcula-se como $\mu \leftarrow \mathbf{c}(g^{\upsilon}, \lambda)$; na verificação μ calcula-se como $\mu \leftarrow \mathbf{c}(g, \alpha) \cdot \mathbf{c}(\beta, p^{h})$.
   Em tudo o resto o esquema mantém-se inalterado e é fácil ver que está correcto.

Uma variante do esquema de assinaturas de Cha & Cheon (protocolo 31) pode ser construída usando um "setup" e geração de chaves do tipo II (protocolo 40) e designa-se por **assinaturas BLQM**.

O esquema BLQM aqui apresentado usa emparelhamentos e a função $\boldsymbol{\kappa}$. Este esquema pode ser transformado num esquema análogo que use co-emparelhamentos $\mathbf{c} \colon \mathbb{G} \times \Gamma \to \Gamma$. Recomenda-se ao aluno que, como exercício, construa uma versão do BLQM usando co-emparelhamentos.[63].

[63] Sugestão: ver a adaptação do esquema IBE de Sakay & Kasahara aos co-emparelhamentos

Protocolo 52 : Esquema de Assinaturas BLMQ (Barreto, Libert, McCullagh & Quisquater)

"Setup&Extract" Usa os elementos comuns e a geração de chaves do tipo II (protocolo 40,pag. 419). A função de *hash* H_r tem a aridade $H_r : \Gamma \times \mathbb{B}^* \to \mathbb{Z}_r^*$.

Assinatura O "prover" $\mathcal{P}$ assina o texto m.

(1) $\mathcal{P} : \upsilon \leftarrow \mathbb{Z}_r^* \quad ; \quad \mu \leftarrow \boldsymbol{\kappa}(g, g, \upsilon) \quad ; \quad h \leftarrow H_r(\mu, m)$

(2) $\mathcal{P} : \lambda, \upsilon \to \sigma = h \| \lambda^{\upsilon + h}$

Verificação O "verifier" $\mathcal{V}$ verifica a assinatura σ de $\mathcal{P}$ no texto m usando chave pública β do TA.

(1) $\mathcal{V} : h, \alpha \leftarrow \sigma \quad ; \quad p \leftarrow \mathrm{ID}(\mathcal{P}) \quad ; \quad \mu' \leftarrow \boldsymbol{\kappa}(g, g, -h) \cdot \boldsymbol{\kappa}(\alpha, \beta\, g^p, 1)$

(2) $\mathcal{V} : h \ =?= \ H_r(\mu', m)$

A correcção deriva do facto de ser $\lambda = g^{(s+p)^{-1}}$ e $\beta = g^s$. Donde, o termo $\beta\, g^p$ tem o valor $g^{(s+p)}$ e $\alpha = g^{(s+p)^{-1} \cdot (\upsilon + h)}$. Se representar-mos por γ o valor $\boldsymbol{\kappa}(g, g, 1)$, tem-se

$$\mu = \gamma^{\upsilon} \quad , \quad \mu' = \gamma^{-h} \cdot \gamma^{(s+p)^{-1} \cdot (\upsilon + h) \cdot (s+p)} = \gamma^{-h} \cdot \gamma^{(\upsilon + h)} = \gamma^{\upsilon}$$

Como $\mu = \mu'$, tem-se $h = H_r(\mu', m)$.

6.6 Autenticação mútua em IBC

Nas secções anteriores vimos esquemas e protocolos onde, essencialmente, só um dos agentes (para além do TA) tinha necessidade de chave privada porque só esse agente necessitava de ser autenticado.

Nesta secção vamos ver técnicas criptográficas que envolvem a autenticação mútua de dois agentes distintos A e B e, portanto, ambos os agentes necessitam de chaves privadas. Estão nesta categoria os protocolos de acordo de chaves e os protocolos "sygn-encrypt" que assinam e cifram, simultaneamente, uma mensagem.

O exemplo mais simples de protocolo que requer duas chaves privadas é o protocolo de Diffie-Hellman (protocolo 6, pág. 325). Resumidamente

> *Num grupo DH de gerador* g*, o agente* A*, com chave privada* a*, torna público* g^a *e o agente* B*, com chave privada* b*, torna público* g^b*.*

É bem conhecido que este protocolo simples de está sujeito ao ataque de "homem-no-meio" porque não tem autenticação mútua dos agentes intervenientes.

O protocolo tem, no entanto, a vantagem de trocar mensagens muito simples: os valores g^a e g^b. Será interressante manter estas mensagens (ou semelhantes) mas acrescentar a autenticação dos agentes. Isso torna-se possível se recorrer-mos a emparelhamentos ou co-emparelhamentos.

Protocolo 53 : Acordo de Chaves (Smart, Chen & Kudla)

"Setup" & "Extract´´" Tipo I, sendo s a chave privada do TA, $\beta = g^s$ a sua chave pública. As chaves privadas dos agentes intervenientes são $\lambda_A = q^s$ e $\lambda_B = p^s$ sendo $q = \mathsf{ID}(A)$ e $p = \mathsf{ID}(B)$.

"Run" Troca de mensagens

(1) $A\colon a \leftarrow \mathbb{Z}_r^* \quad ; \quad a \rightarrow \sigma_A = g^a$

(2) $B\colon b \leftarrow \mathbb{Z}_r^* \quad ; \quad b \rightarrow \sigma_B = g^b$

Construção Os agentes constroem a mesma chave e autenticam-se mutuamente

(1) $A\colon p \leftarrow \mathsf{ID}(B) \quad ; \quad \mu_A \leftarrow \boldsymbol{\kappa}(\lambda_A, \sigma_B, 1) \cdot \boldsymbol{\kappa}(p, \beta, a) \quad ; \quad \kappa_A \leftarrow (\sigma_B)^a \,\|\mu_A$

(2) $B\colon q \leftarrow \mathsf{ID}(A) \quad ; \quad \mu_B \leftarrow \boldsymbol{\kappa}(\lambda_B, \sigma_A, 1) \cdot \boldsymbol{\kappa}(q, \beta, b) \quad ; \quad \kappa_B \leftarrow (\sigma_A)^b \,\|\mu_B$

Notas

1. Porque $\lambda_A = q^s$ e $\lambda_B = p^s$, tem-se

$$\mu_A = \boldsymbol{\kappa}(q, g, s\,b) \cdot \boldsymbol{\kappa}(p, g, s\,a) \qquad \text{e} \qquad \mu_B = \boldsymbol{\kappa}(p, g, s\,a) \cdot \boldsymbol{\kappa}(q, g, s\,b)$$

   Portanto $\mu_A = \mu_B$; como $(\sigma_B)^a = g^{a\,b} = (\sigma_A)^b$, tem-se $\kappa_A = \kappa_B$; i.e. ambos os agentes reconstroem a mesma chave.

2. Existe autenticação mútua dos agentes porque a chave κ construída por cada um desses agentes depende da identidade pública do outro agente e da chave privada do agente que a constrói. Um eventual intruso C não consegue passar informação a A que o faça calcular uma chave pré-definida por C pensando que está a interagir com B.

A informação de autenticação está na componente $\mu_A = \mu_B$ da chave. Note-se que essa componente é calculada como o produto de dois termos em que cada um deles contém uma parte de informação pública (a chave pública β ou uma das mensagens) e uma parte de informação privada (a chave privada ou o segredo gerado localmente). Esta observação faz com que, escolhendo outros termos com estas caracteristicas, seja possível gerar variantes deste esquema básico:

Variante com co-emparelhamentos

Existe uma simples adaptação deste protocolo a co-emparelhamentos $\mathbf{c}\colon \mathbb{G} \times \Gamma \to \Gamma$. Para isso a função de *hash* tem de dar valores em Γ e constói-se

$$\mu_A \leftarrow \mathbf{c}(\sigma_B, \lambda_A) \cdot \mathbf{c}(\beta, p^a) \quad , \quad \mu_B \leftarrow \mathbf{c}(\sigma_A, \lambda_B) \cdot \mathbf{c}(\beta, q^b)$$

Tem-se então $\mu_A = \mu_B = \mathbf{c}(g, q)^{s\,b} \cdot \mathbf{c}(g, p)^{s\,a}$.

Variante Chen&Kudla

Neste caso as mensagens trocadas são

$$A\colon a \to \sigma_A = q^a \quad , \quad B\colon b \to \sigma_B = p^b$$

As chaves são calculadas como

$$\mu_A \leftarrow \boldsymbol{\kappa}(\lambda_A, \sigma_B \cdot p^a, 1) \quad , \quad \mu_B \leftarrow \boldsymbol{\kappa}(\sigma_A \cdot q^b, \lambda_B, 1)$$

É simples verificar que $\mu_A = \mu_B = \boldsymbol{\kappa}(q, p, s\,(a+b))$.

Variante Chen&Kudla com emparelhamentos

Exercício a cargo do aluno.

Um esquema similar pode ser construído com geração de chaves do tipo II.

Protocolo 54 : Acordo de Chaves (McCullagh & Barreto)

"Setup" & "Extract" Tipo II, sendo s a chave privada do TA, $\beta = g^s$ a sua chave pública. As chaves privadas dos agentes intervenientes são $\lambda_A = g^{f(s,q)}$ e $\lambda_B = g^{f(s,p)}$ sendo $q = \mathrm{ID}(A)$ e $p = \mathrm{ID}(B)$ e $f(s,x) = (s+x)^{-1}$.

"Run" Troca de mensagens

(1) $A\colon a \leftarrow \mathbb{Z}_r^* \;\;;\;\; p \leftarrow \mathrm{ID}(B) \;\;;\;\; a \rightarrow \sigma_A = (g^p \cdot \beta)^a$

(2) $B\colon b \leftarrow \mathbb{Z}_r^* \;\;;\;\; q \leftarrow \mathrm{ID}(A) \;\;;\;\; b \rightarrow \sigma_B = (g^q \cdot \beta)^b$

Construção Os agentes constroem a mesma chave e autenticam-se mutuamente

(1) $A\colon \kappa_A \leftarrow \boldsymbol{\kappa}(\lambda_A, \sigma_B, 1) \cdot \boldsymbol{\kappa}(g, g, a)$

(2) $B\colon \kappa_B \leftarrow \boldsymbol{\kappa}(\lambda_B, \sigma_A, 1) \cdot \boldsymbol{\kappa}(g, g, b)$

Note-se que $\sigma_B = g^{b\,(s+q)}$. Atendendo à forma de λ_A tem-se $\kappa_A = \boldsymbol{\kappa}(g, g, f(s,q) \cdot b \cdot (s+q)) \cdot \boldsymbol{\kappa}(g, g, a) = \boldsymbol{\kappa}(g, g, a+b)$. Dualmente se mostra que $\kappa_B = \boldsymbol{\kappa}(g, g, a+b) = \kappa_A$.

Este protocolo também se adapta facilmente ao uso de co-emparelhamentos. Como exercício o aluno deve construir tal adaptação.

□

As técnicas IBC mais interessantes, envolvendo autenticação mútua de agentes, são os **protocolos "sign-encryption"** (abreviadamente **"signcryption"**) que combinam esquemas de assinaturas com esquemas de cifra.

Nestes protocolos, o emissor $\mathcal{E}$ cifra uma mensagem m com a identidade do agente $\mathcal{D}$ como chave pública e, simultaneamente, assina essa mensagem com a sua chave privada. Por seu lado, $\mathcal{D}$ verifica a autenticidade da mensagem usando a identidade de $\mathcal{E}$ como chave pública e recupera m do criptograma, com a sua chave privada.

Isto significa que ambos os agentes têm de ter chaves privadas: $\mathcal{E}$ precisa de uma chave privada para assinar a mensagem e $\mathcal{D}$ precisa de uma chave privada para decifrar a mensagem.

Vamos apresentar dois protocolos de "signcryption" baseados em emparelhamentos: um deles usa geração de chaves do tipo I e o outro usa geração de chaves do tipo II. Ambos podem ser facilmente adpatados a co-emparelhamentos ficando a cargo do aluno realizar as modificações apropriadas.

Protocolo 55 : Signcryption (Chen, Malone & Lee)

"Setup" & "Extract´´" Tipo I, sendo s a chave privada do TA, $\beta = g^s$ a sua chave pública. As chaves privadas dos agentes intervenientes são $\lambda_e = e^s$ e $\lambda_d = d^s$ sendo $e = \mathrm{ID}(\mathcal{E})$ e $d = \mathrm{ID}(\mathcal{D})$.
São usadas funções de*hash* $f : \Gamma \to \mathbb{B}^t$ e $H_r : \mathbb{G} \times \mathbb{B}^t \to \mathbb{Z}_r^*$.

"Sign-Encrypt" $\mathcal{E}$ assina a mensagem $m \in \mathbb{B}^t$ destinada a $\mathcal{D}$ e cifra-a; produz um "signed-cryptogram" σ.

(1) $\mathcal{E} : \upsilon \leftarrow \mathbb{Z}_r^* \;;\; \gamma \leftarrow e^{\upsilon} \;;\; h \leftarrow H_r(\gamma, m) \;;\; \kappa \leftarrow \lambda_e^{h+\upsilon} \,\|\, \mathcal{E} \,\|\, m$ – *sign*

(2) $\mathcal{E} : \mu \leftarrow \boldsymbol{\kappa}(\lambda_e, d, \upsilon) \;;\; \mu, \kappa, \gamma \to \sigma = \gamma \,\|\, (f(\mu) \oplus \kappa)$ – *encrypt*

"Decrypt-Verify" $\mathcal{D}$ recupera a mensagem e verifica a sua autenticidade.

(1) $\mathcal{D} : \gamma, y \leftarrow \sigma \;;\; \mu \leftarrow \boldsymbol{\kappa}(\gamma, \lambda_d, 1) \;;\; \alpha, \mathcal{E}, m \leftarrow f(\mu) \oplus y$ – *decrypt*

(2) $\mathcal{D} : e \leftarrow \mathrm{ID}(\mathcal{E}) \;;\; h \leftarrow H_r(\gamma, m) \;;\; \mathrm{DDHP}(\beta, \gamma \cdot e^h, \alpha) \;=?=\; 1$ – *verify*

1. O emissor $\mathcal{E}$ começa por substituir a mensagem m por uma versão κ autenticada da mesma, juntando-lhe a sua própria identidade $\mathcal{E}$ e uma assinatura $\alpha = \lambda_e^{(\upsilon+h)}$; κ é depois cifrada a partir de uma chave μ convertida para o espaço apropriado pela função $f(\cdot)$. A redundância γ liga estes dois passos e vai permitir recuperar a chave μ e verificar a assinatura.
2. Na operação de cifra, a "chave" μ é calculada como $\boldsymbol{\kappa}(\lambda_e, d, \upsilon) = \boldsymbol{\kappa}(e, d, s\,\upsilon)$. Na operação de decifra é calculada como $\boldsymbol{\kappa}(\gamma, \lambda_d, 1) = \boldsymbol{\kappa}(e^{\upsilon}, d^s, 1) = \boldsymbol{\kappa}(e, d, \upsilon\, s)$. A chave é a mesma e $\mathcal{D}$ consegue recuperar, para além da mensagem m, a assinatura α e a identidade do emissor $\mathcal{E}$.
3. A verificação da assinatura recorre a um oráculo DDHP (que pode ser implementado com a mesma função $\boldsymbol{\kappa}$), testando se o triplo formadado por $\beta = g^s$, pelo termo $\gamma \cdot e^h = e^{(\upsilon+h)}$ e $\alpha = \lambda_e^{(\upsilon+h)} = e^{s\,(\upsilon+h)}$ formam um triplo DH.

Em seguida mostra-se uma versão modificada do protocolo BLMQ em que, tal como no esquema anterior, a assinatura de m precede a cifra.

Protocolo 56 : BLMQ Signcryption Scheme (versão "first sign")

"Setup" & "Extract" Tipo II, sendo s a chave privada do TA, $\beta = g^s$ a sua chave pública. As chaves privadas do emissor e destinatário são $\lambda_e = g^{t(s,e)}$ e $\lambda_d = g^{t(s,d)}$ sendo $e = \mathsf{ID}(\mathcal{E})$ e $d = \mathsf{ID}(\mathcal{D})$ e $t(s,x) = (s+x)^{-1}$.

São usadas funções de *hash* $f : \Gamma \to \mathbb{B}^t$ e $H_r : \Gamma \times \mathbb{B}^t \to \mathbb{Z}_r^*$.

"Sign-Encrypt" $\mathcal{E}$ cifra a mensagem $m \in \mathbb{B}^t$ destinada a $\mathcal{D}$ e assina-a; produz um "signed-cryptogram" σ.

(1) $\mathcal{E} : \upsilon \leftarrow \mathbb{Z}_r^* \;\; ; \;\; \mu \leftarrow \boldsymbol{\kappa}(g, g, \upsilon) \;\; ; \;\; h \leftarrow H_r(\mu, m) \;\; ; \;\; \kappa \leftarrow \lambda_e^{(\upsilon+h)} \,\|\, \mathcal{E} \,\|\, m$ *– sign*

(2) $\mathcal{E} : d \leftarrow \mathsf{ID}(\mathcal{D}) \;\; ; \;\; \gamma \leftarrow g^{d\,\upsilon} \cdot \beta^{\upsilon} \;\; ; \;\; \gamma, \mu, \kappa \to \sigma = \gamma \,\|\, (f(\mu) \oplus \kappa)$ *– encrypt*

"Decrypt-Verify" $\mathcal{D}$ recupera a mensagem e verifica a sua autenticidade.

(1) $\mathcal{D} : \gamma, y \leftarrow \sigma \;\; ; \;\; \mu \leftarrow \boldsymbol{\kappa}(\gamma, \lambda_d, 1) \;\; ; \;\; \alpha, \mathcal{E}, m \leftarrow f(\mu) \oplus y$ *– decrypt*

(2) $\mathcal{D} : h \leftarrow H_r(\mu, m) \;\; ; \;\; e \leftarrow \mathsf{ID}(\mathcal{E}) \;\; ; \;\; \mu \cdot \boldsymbol{\kappa}(g, g, h) \stackrel{?}{=} \boldsymbol{\kappa}(\alpha, g^e \cdot \beta, 1)$ *– verify*

1. Tal como no protocolo 55, o emissor substitui a mensagem m por uma versão autenticada κ que contém, para além de m, a

assinatura $\alpha = \lambda_e^{(\upsilon+h)}$ e a sua identidade $\mathcal{E}$. A ligação entre a assinatura e a cifra é feita por uma chave comum μ usada tanto para construir o *hash* $h = H_r(\mu, m)$ como a chave de cifra $f(\mu)$. Na redundância γ vai estar incluída a identidade $\mathcal{D}$ do destinatário de forma a este poder, com a sua chave privada, recuperar μ.

2. Note-se que $\gamma = g^{\upsilon\,(d+s)}$ e $\lambda_d = g^{t(s,d)}$ com $t(s,d) = (d+s)^{-1}$; portanto

$$\boldsymbol{\kappa}(\gamma, \lambda_d, 1) \;=\; \boldsymbol{\kappa}(g, g, \upsilon \cdot (d+s) \cdot (d+s)^{-1}) \;=\; \boldsymbol{\kappa}(g, g, \upsilon) \;=\; \mu$$

Para verificar a assinatura note-se que

$$\mu \cdot \boldsymbol{\kappa}(g, g, h) \;=\; \boldsymbol{\kappa}(g, g, \upsilon) \cdot \boldsymbol{\kappa}(g, g, h) \;=\; \boldsymbol{\kappa}(g, g, \upsilon + h)$$

Como $\alpha = \lambda_e^{(\upsilon+h)}$, sendo $\lambda_e = g^{t(s,e)}$ e $t(s,e) = (s+e)^{-1}$, tem-se também

$$\boldsymbol{\kappa}(\alpha, g^e \cdot \beta, 1) \;=\; \boldsymbol{\kappa}(\alpha, g^{(s+e)}, 1) \;=\; \boldsymbol{\kappa}(g, g, (s+e)^{-1}\,(\upsilon+h)\,(s+e)) \;=\; \boldsymbol{\kappa}(g, g, \upsilon + h)$$

Os dois esquemas "signcryption" são construídos de forma à identidade do emissor estar protegida no criptograma: o destinatário precisa de decifrar a mensagem para saber qual foi o emissor. Desta forma estes esquemas não só garantem confidencialidade da mensagem como também a privacidade dos intervenientes.

7.Criptografia com Agentes Múltiplos

Frequentemente um determinado passo de uma técnica criptográfica pode requerer a participação de mais do que um agente. Suponhamos, por exemplo, que duas instituições pretendem estabelecer um contracto entre si, contracto esse que tem de ser assinado digitalmente.

Nos esquemas de assinaturas que conhecemos, o acto de assinar é realizado por indivíduos (e não instituições) que são titulares de chaves privadas próprias; não faz qualquer sentido propor-se, por exemplo, uma chave privada institucional porque, por definição, não é privada. Como as chaves privadas são individuais temos de ter um mecanismo de assinaturas que permita a vários indivíduos assinar em nome da instituição.

Pode ser exigida, adicionalmente, mais do que uma assinatura individual para comprometer a instituição; por exemplo, se a instituição tiver 3 directores, pode-se exigir a assinatura individual de quaisquer dois deles para realizar a assinatura institucional. Quando são necessárias várias assinaturas individuais para gerar uma assinatura institucional, também se pode exigir que elas sejam realizadas por uma determinada ordem.

Esboçamos uma instância daquilo que designaremos por **multi-assinatura**. Basicamente uma multi-assinatura é um acto (a assinatura institucional) que assume duas formas principais:

Multi-assinaturas "treshold" (t, n) A assinatura é uma assinatura individual que recorre a informação privada que está distribuída por n agentes; o concluio entre um número de agentes $\leq t$ não é suficiente para determinar essa informação privada; é necessário combinar as chaves privadas de, pelo menos, $t + 1$ desses agentes para construir a chave privada.

Multi-assinaturas estruturadas O esquema de assinaturas é um algoritmo que usa, como oráculo, a realização de assinaturas individuais de vários agentes segundo uma ordem pré-estabelica. Normalmente o acto institucional pode ser realizado por mais do que uma dessas *cadeias de assinaturas*.

Uma outra abordagem, designada por **assinatura de grupo**, define grupos de assinantes individuais que podem contribuir para determinar a assinatura institucional; qualquer assinatura individual dentro do grupo realiza a assinatura institucional; isto corresponde a uma multi-assinatura onde todas as cadeias têm comprimento 1. No entanto exige-se algo mais: o *anonimato*; exige-se que a verificação da assinatura institucional reconheça a autenticidade do documento e a sua autoria (em termos da instituição emissora) mas não seja capaz de identificar a o titular da assinatura individual que lhe deu origem.

□

Nos esquemas de cifra podem-se definir procesos duais. Por simplicidade vamos apenas referir a mecanismos KEM-DEM já que cifras assimétricas são directamente implementáveis com (ou deriváveis de) este tipo de mecanismos.

Um esquema de **multi-KEM** envolve o agente *emissor* $\mathcal{E}$ e vários *destinatários* $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_n$. O emissor gera

uma chave k e um encapsulamento σ recorrendo às chaves públicas $p_1, p_2, \cdots, p_n$ dos vários destinatários de tal forma que, com qualquer uma das chaves privadas correspondentes $s_1, s_2, \cdots, s_n$, é possível recuperar k a partir do encapsulamento.

Resumidamente, o encapsulamento é um algoritmo probabilístico que produz

$$k \,\|\, \sigma \quad \leftarrow \quad \mathsf{mKEM}(p_1, p_2, \cdots, p_n)$$

e o desencapsulamento é um algoritmo determinístico que produz

$$k \quad \leftarrow \quad \mathsf{mKEM}(s_i, \sigma)^{-1} \qquad \forall\, i \in 1..n$$

No esquema multi-KEM qualquer titular de uma das chaves privadas s_i pode reconstruir a chave k e, assim, decifrar qualquer mensagem que tenha sido cifrada com ela. O objectivo é ransmitir a mesma mensagem a múltiplos destinatários reutilizando o mesmo encapsulamento; em relação a um esquema KEM singular, usado uma vez para cada destinatário, poupa-se $(n-1)$ encapsulamentos.

Outro tipo de objectivo envolve apenas um emissor $\mathcal{E}$ e um destinatário $\mathcal{D}$ mas exige que a chave privada usada recuperar k, seja construída a partir de várias fracções. Como caso paricular do de-emcapsulamento multi-chave é usado no chamado **"workflow decryption"**.

Neste processo decifrar é uma operação privilegiada que exige, ao agente destinatário do criptograma, o acesso a um

conjunto de *credenciais* fornecidas por ou mais *autoridades de credenciamento*. A interacção entre os vários agentes rege-se pelos seguintes princípios:

1. O **emissor** especifica o conjunto de credenciais necessárias para decifrar a mensagem numa *"policy"* determinada antes de cifrar a mensagem. A cifra é realizada independentemente das credenciais que estão ou não estão emitidas.
2. As **autoridades de credenciamento** emitem as credenciais no pressuposto que ocorreram determinados eventos. Desta forma as autoridades podem controlar a sequência de eventos (o "workflow") que conduz a um objectivo final de decifrar o criptograma.
3. Cumprindo os requisitos das autoridades, o **destinatário** vai recolhendo as credenciais e, com elas, construindo a chave que permite decifrar o criptograma. Cada credencial age como uma parte da chave privada.
4. Emissor e destinatário podem querer que o protocolo seja "escrow-free"; isto é, por si só, as autoridades de credenciamento não conseguem reunir informação que lhes permite, em conjunto, reconstruir a chave privada.

7.1 Esquemas de Partição de Chaves

O esquema de assinaturas BLS (protocolo 33 na página 403) pode ser facilmente generalizável a vários tipos de objectivos. Vimos que podia ser adaptada a um esquema de assinaturas cegas (protocolo 34, pag. 404) e iremos ver agora como se pode adaptar aos objectivos das multi-assinaturas.

Recordemos que no protocolo BLS básico, "prover" $\mathcal{P}$ detém uma chave privada s e publica a chave pública $\beta = g^s$. Se for h o *hash* da mensagem, $\mathcal{P}$ constrói a assinatura como $\sigma = h^s$.

Uma primeira abordagem consiste em usar um esquema de distribuição do segredo s por vários agentes de uma forma análoga à usada no protocolo 38 (pag. 416) para distribuir por vários agentes a responsabilidade de geração de uma chave privada.

Assume-se que existem vários "provers" $\mathcal{P}_1, \cdots, \mathcal{P}_n$ que são titulares de chaves privadas $s_1, \cdots, s_n \in \mathbb{Z}_r^*$ a que correspondem chaves públicas $\beta_1, \cdots, \beta_n \in \mathbb{G}$ sendo $\beta_i = g^{s_i}$.

Para cada "prover" $\mathcal{P}_i$, tal como no esquema BLS, sobre um determinado *hash* h o segredo s_i determina a assinatura $\sigma_i = h^{s_i}$; o triplo $\langle \beta_i, h, \sigma_i \rangle$ é um triplo DH.

192 Definição

*Uma n-**máscara** em $\mathbb{Z}_r$ é um vector $\mathbf{b} = (b_1, b_2, \ldots, b_n) \in (\mathbb{Z}_r)^n$ de n componentes em $\mathbb{Z}_r$. A máscara é **booleana** se todas as componentes forem 0 ou 1. O **suporte** da máscara é o sub-conjunto de índices i para os quais $b_i \neq 0$; a cardinalidade deste conjunto chama-se **tamanho** da máscara e representa-se por $|\mathbf{b}|$.*

Cada máscara $\mathbf{b}$ permite construir um segredo global s, uma chave pública global β e uma assinatura global σ do modo usual:

$$s = \sum_{i=1}^{n} s_i \cdot b_i \quad , \quad \beta = \prod_{i=1}^{n} (\beta_i)^{b_i} \quad , \quad \sigma = \prod_{i=1}^{n} (\sigma_i)^{b_i} \tag{127}$$

É fácil verificar que g^s coincide com β e que h^s coincide com σ; assim $\langle \beta, h, \sigma \rangle$ é também um triplo DH.

Conforme a estratégia para determinação da máscara $\mathbf{b}$ e, por conseguinte, construção de σ e β, definem-se variantes do esquema básico BLS envolvendo múltiplos assinantes. Essencialmente existe uma escolha entre:

Não coordenação da chave pública

Os "provers" $\mathcal{P}_i$ não coordenam a sua acção: cada assinante gera a publica, independentemente dos restantes, a chave pública β_i e, para cada *hash* h, a assinatura σ_i.

Cabe ao "verifier" escolher os assinantes em quem confia; faz isso gerando uma máscara boolena $\mathbf{b}$ booleana, cujo suporte coincide com esse conjunto de assinantes, e com ela calcular tanto β como σ.

Coordenação na chave pública

Existe um agente coordenador que distribui as chaves privadas s_i pelos "provers" e é o único que conhece o segredo global s e, por isso, é o único capaz de gerar a chave pública $\beta = g^s$. A verificação da assinatura recorre sempre a esta chave pública única.

Na hipótese de não-coordenação da chave pública o esquema BLS com múltiplos "provers" pode ser:

Protocolo 57 : Esquemas de Assinaturas BLS com n-máscaras

"Setup" Os "provers" $\mathcal{P}_1, \cdots, \mathcal{P}_n$ geram as chaves privadas e publicam as chaves públicas correspondentes.

(1) $\mathcal{P}_i$: $s_i \leftarrow \mathbb{Z}_r^*$; $s_i \rightarrow \beta_i = g^{s_i}$ para $i = 1..n$

Assinatura Sobre o mesma mensagem M cada um dos "provers" gera uma assinatura

(1) $\mathcal{P}_i$: $h \leftarrow H(M)$; $s_i \rightarrow \sigma_i = h^{s_i}$ para $i = 1..n$

Construção Usando uma máscara $\mathbf{b}$, o "verifier" constrói a assinatura global σ e a chave pública global β.

(1) $\mathcal{V}$: $\sigma \leftarrow \prod_i (\sigma_i)^{b_i}$; $\beta \leftarrow \prod_i (\beta_i)^{b_i}$

Verificação $\mathcal{V}$ verifica a assinatura σ do texto M com a chave pública β

(1) $\mathcal{V}$: $h \leftarrow H(M)$; $\mathrm{DDHP}(\beta, h, \sigma)$

Na hipótese de existir uma chave pública única β, a geração da assinatura σ recorre a duas estratégias possíveis:

Geração pelo verificador / assinatura não-anónima

A geração da assinatura não é coordenada; um sub-conjunto dos "provers" gera e publica assinaturas individuais σ_i. Cabe ao verificador recolher estas assinaturas, recolher a identidade dos "provers" assinantes e, se tiver informação suficiente, reconstruir a assinatura global σ.

Geração por um "prover" privilegiado / assinatura anónima

Existe um "prover" coordenador que recolhe um número suficiente de assinaturas individuais σ_i e a identidade dos votantes respectivos. Com esta informação constrói a assinatura global que publica.

A diferença crucial entre estas duas estratégias reside no facto de, na primeira construção, a identidade dos assinantes é conhecida pelo verificador enquanto que na segunda opção essa identidade está escondida do verificador. Assim a primeira assinatura não é anónima enquanto a segunda é completamente anónima.

□

Em qualquer dos casos, o facto de o segredo s ser "distribuído" pelos vários "provers" $\mathcal{P}_i$ leva-nos a um dos problemas clássicos em Criptografia, o **problema da partição de segredos**. Essencialmente este problema pode-se definir do seguinte modo:

Problema da Partição de Segredos

Dados inteiros positivos $t \leq n$ *e um domínio recursivo enumerável* D, *gerar um segredo* $\mathbf{s} \in D$ *e um conjunto* $\mathcal{S} = \{z_1, z_2, \ldots, z_n\}$ *de items chamados* **sombras** *ou* **quotas**, *de tal forma que:*

1. *O conhecimento de qualquer sub-conjunto* $R \subseteq \mathcal{S}$, *de cardinalidade inferior a um limite* t, *não permite conhecer* $\mathbf{s}$,
2. *Existe um algoritmo PPT, designado* **algoritmo de recuperação**, *que, sob input de um qualquer* $R \subseteq \mathcal{S}$ *falha se* $|R| < t$ *e dá o resultado* $\mathbf{s}$, *se* $|R| \geq t$.

Na terminologia inglesa uma solução para este problema designa-se por (t, n) **threshold scheme**. Este problema tem sido amplamente estudado desde que foi introduzido por Shamir & Blakley em 1979.

Neste curso interressa-nos particularizar estes esquemas impondo as seguintes restrições:

(t, n)-partição linear de segredos

1. O segredo $\mathbf{s}$ e todas as quotas z_i são elementos de um corpo $\mathbb{K}$.
2. Cada quota z tem um *titular* identificado por uma marca ("label") $l \in \mathcal{L}$ com $|\mathcal{L}| = n$.
3. Existe uma função PPT implementável $\zeta : \mathcal{L} \longrightarrow \mathbb{K}$ que mapeia marcas em quotas.
4. Existe um algoritmo PPT, M que recebe como input um conjunto de t marcas $\{l_1, \cdots, l_t\}$ e produz uma máscara $\mathbf{b} \in \mathbb{K}^t$ que verifica

$$\mathbf{s} = \sum_{k=1}^{t} b_k \cdot \zeta(l_k) \tag{128}$$

   O algoritmo de recuperação obtém $\mathbf{s}$ usando ζ, M e (128).
5. $\mathbb{K}$, $\mathcal{L}$ e M são informação pública, enquanto que ζ é informação privada.

O exemplo paradigmático é o esquema de Shamir que usa a interpolação polinomial para determinar o segredo $\mathbf{s}$

Partição de segredos (t, n) de Shamir

Seja $\mathbb{K}$ *um corpo e* $f \in \mathbb{K}[x]$ *um polinómio com* $(t-1)$ *raízes distintas tal que* $f(0) = \mathbf{s}$.

$$f(x) \;=\; c_1 + c_2 \cdot x + \cdots + c_t \cdot x^{t-1} \qquad \textit{com} \quad c_1 = \mathbf{s}$$

Seja $\mathcal{L}$ *um conjunto de marcas de cardinalidade* n *e* $H : \mathcal{L} \to \mathbb{K}^*$ *uma função injectiva. As quotas do segredo* $\mathbf{s}$ *são, para cada* $l \in \mathcal{L}$ *, dadas por* $z_l = f(H(l))$.

Dado um conjunto de t marcas $\{l_1, l_2, \ldots, l_t\}$, a recuperação de $\mathbf{s}$ a partir das quotas z, faz-se por simples interpolação polinomial no conjunto de t pontos $\{(x_i, z_i)\}_{i=1}^{t}$ em que $x_i = H(l_i)$ e $z_i = f(x_i)$.

De facto seja $\mathbf{X} \in \mathbb{K}^{t \times t}$ a matriz quadrada definida por $\mathbf{X}_{ij} = x_i^j$. Pela definição tem-se $z_i = \sum_j \mathbf{X}_{ij} \cdot c_j$. Em notação matricial será $\mathbf{X}\,\mathbf{c} = \mathbf{z}$ sendo $\mathbf{c} = (c_1, \ldots, c_t)$ o vector dos coeficientes e $\mathbf{z} = (z_1, \ldots, z_t)$ o vector das quotas.

Portanto tem-se $\mathbf{c} = \mathbf{X}^{-1}\,\mathbf{z}$. Isto significa que

$$\mathbf{s} \;=\; c_1 \;=\; \sum_{i=1}^{t} b_i \cdot z_i$$

em que os coeficientes b_i formam a primeira linha da matriz $\mathbf{X}^{-1}$.

Estas observações sugerem a seguinte implementação do esquema de partilha de segredos (t, n) de Shamir

Protocolo 58 : (t, n) Partilha de segredos de Shamir

Elementos Comuns Um corpo $\mathbb{K}$ e o conjunto de marcas vistas como bit-strings

(1) Um corpo $\mathbb{K}$ e uma função de *hash* $H : \mathbb{B}^* \rightarrow \mathbb{K}^*$

(2) Um conjunto finito $\mathcal{L}$ de n marcas $l \in \mathbb{B}^*$

Geração e partilha Um TA gera o segredo $\mathbf{s}$ e gera o segredo e as quotas.

(1) $T : c_i \leftarrow \mathbb{K}^* \quad i = 1 \cdots t \quad ; \quad \mathbf{s} \leftarrow c_1$

(2) $T : x_l \leftarrow H(l) \quad ; \quad z_l \leftarrow \sum_j x_l^j \cdot c_j \quad$ para todo $l \in \mathcal{L}$

Recuperação Dado um conjunto de t marcas $\mathcal{L} = \{l_1, \ldots, l_t\}$ construir a máscara $\mathbf{b}$ que verifica (128). Para isso constrói-se a matriz $\mathbf{X}$ que, depois, é invertida.

(1) $* : x_i \leftarrow H(l_i) \quad i = 1 \ldots t \quad ; \quad \mathbf{X_{ij}} \leftarrow x_i^j \quad i, j = 1 \ldots t$

(2) $* : \mathbf{b} \leftarrow 1^{\text{a}}$ linha de $(\mathbf{X})^{-1}$

Desta forma pode-se estabelecer um protocolo BLS multi-assinatura. São apresentadas duas variantes do passo "assinatura": uma "variante não-anónima" onde é público quem são os assinantes individuais e quais são as assinaturas que eles produzem, e uma variante "assinatura anónima" onde as assinaturas σ_l não são públicas nem é público o conjunto de assinantes.

Assume-se que o esquema de partilha de chaves usa o protocolo 58 com o corpo $\mathbb{K} \equiv \mathbb{Z}_r$. Assume-se que as

quotas são distribuídas por um conjunto de n "provers" identificados por marcas que coincidem com as respectivas identidades. A função de *hash* ID mapeia identidades em elementos de $\mathbb{Z}_r^*$.

Protocolo 59 : BLS multi-assinatura

"Setup" Um TA gera um segredo, distribui as respectivas quotas por um conjunto de "provers" identificados pelas suas identidades e publica a chave pública correspondente.

(1) $T: c_j \leftarrow \mathbb{Z}_r^* \quad , \quad j = 1 \cdots t \quad ; \quad c_1 \rightarrow \beta = g^{c1}$

(2) $T: x_i \leftarrow \mathsf{ID}(\mathcal{P}_i) \quad ; \quad s_i \leftarrow \sum_j x_i^j \cdot c_j$ para todo $i = 1..n$

(3) $\mathcal{P}_i: s_i \leftarrow T.s_i$ para todo $i = 1..n$

Assinatura / variante não-anónima É público o conjunto $\mathcal{A}$ dos t assinantes que tornam públicas as assinaturas individuais. O "verifier" $\mathcal{V}$ constrói a assinatura global.

(1) $\mathcal{P}_l: h \leftarrow H(M) \quad ; \quad s_l \rightarrow \sigma_l = h^{s_l}$ para todo $l \in \mathcal{A}$

(2) $\mathcal{V}$ constrói a máscara $\mathbf{b}$ usando o algoritmo de recuperação no protocolo 58 com o conjunto $\mathcal{A}$

(3) $\mathcal{V}: \sigma \leftarrow \prod_{l \in \mathcal{A}} (\sigma_l)^{b_l}$

Assinatura/ variante anónima Existe um "prover" privilegiado $\mathcal{P}$ que conhece $\mathcal{A}$ e as assinaturas respectivas.

(1) $\mathcal{P}_l: h \leftarrow H(M) \quad ; \quad \sigma_l \leftarrow h^{s_l}$ para todo $l \in \mathcal{A}$

(2) $\mathcal{P}$ constrói a máscara $\mathbf{b}$ usando sobre $\mathcal{A}$ o algoritmo de recuperação no protocolo 58

(3) $\mathcal{P}: \sigma_l \leftarrow \mathcal{P}_l.\sigma_l \quad \forall\, l \in \mathcal{A} \quad ; \quad \cdot \rightarrow \sigma = \prod_{l \in \mathcal{A}} (\sigma_l)^{b_l}$

Verificação Como no protocolo BLS usual.

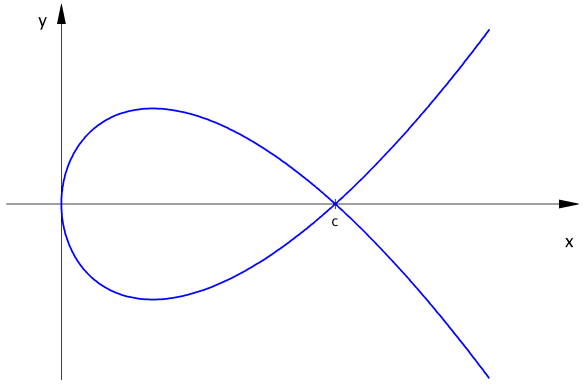
8.Curvas Elípticas

Na procura de grupos cíclicos com as melhores propriedades criptográficas, capazes de aliar garantias de segurança (na perspectiva de dificuldade computacional em resolver os problemas clássicos: DLP, CDHP, BCDHP, etc.) com eficiência de implementação (eficiência na representação e na manipulação computacional), uma área tradicional da Matemática foi "redescoberta": a Geometria Algébrica.

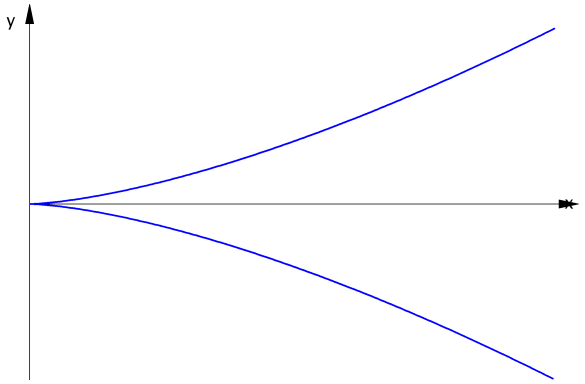
Esta área da Matemática personifica, por um lado, a visão que no século XIX se tinha da Álgebra: o estudos dos polinómios e das sua raízes. Por outro lado dá-lhe a dimensão geométrica e, por isso, estudava essencialmente as curvas definidas em espaços de dimensão real ou racional por equações polinomiais.

Como exemplo considere-se as seguintes curvas definidas no plano $\mathbb{Q}^2$ pelas raízes $\phi(x, y)$ dos polinómios indicados. Note-se que são polinómios a duas variáveis e todos são do 2^o grau em y e do 3^o grau em x.

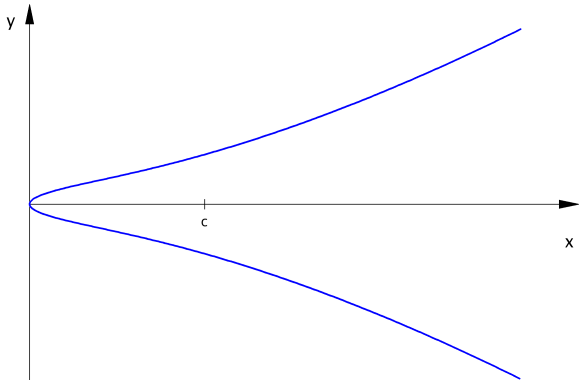
Informalmente, entende-se por *curva* plana racional o conjunto dos pontos $(x, y) \in \mathbb{Q}^2$ para os quais $\phi(x, y) = 0$, i.e, os pontos (x, y) que são *raízes* deste polinómio. Note-se que, ao contrário do que ocorre num polinómio a uma só variável em que as raízes são em número limitado pelo grau do polinómio, para polinómios com mais do que uma variáveis as raízes não estão limitadas pelo grau.



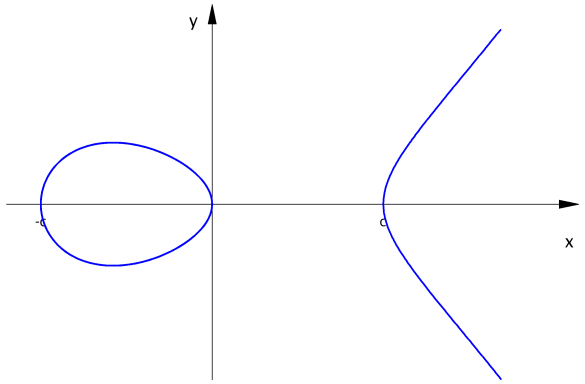
$y^2 - x\,(x-c)^2$



$y^2 - x^3$



$y^2 - x\,(x^2 + c^2)$



$y^2 - x\,(x^2 - c^2)$

Figura 4: Quatro exemplos de curvas planas cúbicas em x e quadráticas em y.

Alguns aspectos importantes que devemos ter em conta:

Estamos habituados a ver este tipo de curvas no plano real $\mathbb{R}^2$; desta forma existem pontos que pertencem à curva no plano real que não estão no plano $\mathbb{Q}^2$. Por exemplo, um ponto de coordenada $x = 1/2$ na curva $y^2 - x(x^2+1)$ tem ordenada $y = \pm\sqrt{5/8}$ que não pertence a $\mathbb{Q}$ (apesar de pertencer a uma extensão algébrica desse corpo). Portanto procurar os pontos em $\mathbb{Q}^2$ da curva $y^2 - x(x^2+1)$, não é uma tarefa trivial.

Na definição de curva, o corpo $\mathbb{Q}$ não tem nada de particular e pode ser substituído por um qualquer outro corpo $\mathbb{K}$. Agora ϕ pertence ao anel dos polinómios a duas variáveis com coeficientes numa extensão de $\mathbb{K}$.

Assim, genericamente, e como primeira definição, pode-se considerar que uma **curva plana** $C/\mathbb{K}$ é o conjunto das raízes em $\mathbb{K}^2$ de um polinómio $\phi \in \mathbb{K}[x, y]$.

Para evidenciar a relação entre a curva, o corpo de suporte e o polinómio, representamos a curva por $C/\mathbb{K}: \phi$.

Note-se que os coeficientes do polinómio estão numa extensão do corpo $\mathbb{K}$ mas não necessariamente em $\mathbb{K}$. Isto faz com que não seja suficiente escolher uma coordenada $x \in \mathbb{K}$ para existir um $y \in \mathbb{K}$ tal que $\phi(x, y) = 0$. De facto pode até acontecer que o polinómio $\phi(x, y)$ não tenha qualquer raiz em $\mathbb{K}^2$

Por exemplo, considere-se um polinómio com coeficientes em $\mathbb{C}$, $\phi(x, y) = i\,y - x - i$ (sendo i a unidade imaginária, $i^2 + 1 = 0$). A curva em $\mathbb{Q}$ definida por este polinómio é formada por um só ponto $\{(0, 1)\}$.

A procura dos pontos de uma curva é, portanto, um processo essencial e, para isso. pode-se recorrer a algumas "heurísticas". Por exemplo, quando o corpo $\mathbb{K}$ é finito, a curva é também um conjunto finito uma vez que o número de raízes de $\phi(x,y)$ em $\mathbb{K}$ está limitado pelo número de pontos disponíveis no plano $\mathbb{K}^2$. De facto, se $\mathbb{K} \equiv \mathbb{F}_q$ for o corpo finito de q elementos, o plano $\mathbb{K}^2$ tem exactamente q^2 possíveis pontos.

Assim é possível, em princípio, encontrar a curva C percorrendo sistematicamente todos $(x,y) \in \mathbb{K}$ e testando, para cada ponto, se verifica $\phi(x,y) = 0$. Obviamente, este procedimento só será computacionalmente viável se q^2 for razoavelmente pequeno.

Algumas formas particulares de polinómio facilitam a construção da curva. Por exemplo, uma classe de curvas importante é a formada pelas *rectas*. No plano $\mathbb{K}^2$ uma **recta** é definida por um polinómio de primeiro grau $l \in \overline{\mathbb{K}}[x,y]$, com $l(x,y) = a\,y + b\,x + c$ sendo $a,b,c \in \overline{\mathbb{K}}$ e $(a \neq 0) \vee (b \neq 0)$.

A curva $C\colon l(x,y)$ goza de uma propriedade muito importante: se tiver dois pontos distintos $P,Q \in \mathbb{K}^2$, com $P = (x_1,y_1)$ e $Q = (x_2,y_2)$, então qualquer $x \in \mathbb{K}$, distinto de x_1 e x_2, ou qualquer $y \in \mathbb{K}$ distinto de y_1 ou y_2, determinam um terceiro ponto $(x,y) \in \mathbb{K}^2$ na mesma curva [64].

[64] Passando a recta pelos pontos $(x,y),(x_1,y_1)$ e (x_2,y_2), tem de se verificar $a\,(y_2 - y_1) + b\,(x_2 - x_1) = 0$ e $a\,(y - y_1) + b\,(x - x_1) = 0$. Se for $a = 0$ tem-se $x = x_1 = x_2 \in \mathbb{K}$; todo $y \in \mathbb{K}$ determina um ponto em $\mathbb{K}^2$. Se for $a \neq 0$, verifica-se $(y - y_1) = (y_2 - y_1)\,(x - x_1)/(x_2 - x_1)$; todo $x \in \mathbb{K}$ determina um ponto em $\mathbb{K}^2$.

Quando as rectas se sobrepõem com outras curvas, definidas por polinómios de grau mais elevado, esta propriedade permite dizer
Considere-se, por exemplo, a curva $C\colon y^2 - x^3 - 1$ e a recta $L\colon y - x - 3/4$ representadas na figura 5. Queremos ver que curvas definem em $\mathbb{Q}$; nomeadamente queremos determinar as raízes racionais de $y^2 - x^3 - 1$.

A recta intersecta a curva C em 3 pontos; pela propriedade das rectas, se dois deles tiverem coordenadas racionais o terceiro também tem coordenadas racionais.

Isto sugere um mecanismo para construção de C. Se forem já conhecidos dois pontos de coordenadas racionais em C, traça-se a recta que eles determinam e calcula-se o terceiro ponto de intersecção com a curva. Esse ponto, porque está na recta, também tem coordenadas racionais desde que uma das coordenadas seja racional.

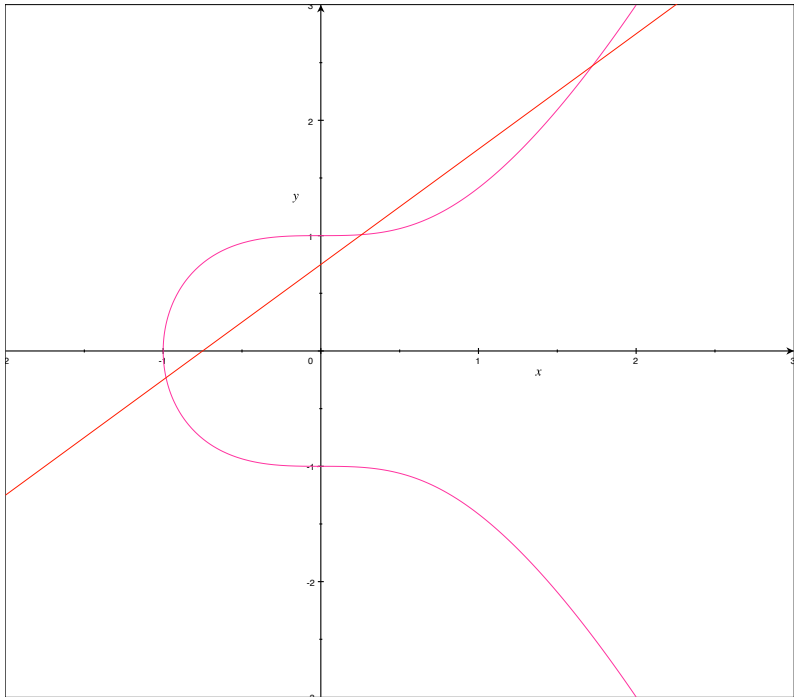


Figura 5: Curvas $y^2 - x^3 - 1$ e $y - x - 3/4$.

Na viabilidade deste mecanismo reside a razão porque se usam este tipo de curvas em Criptografia.

Intersecção de rectas com curvas cúbicas em $\mathbb{Q}$

Considere-se, por exemplo, uma curva cúbica definida pelos pontos $(x, y) \in \mathbb{C}^2$ que verificam a equação

$$y^2 + (a_1\, x + a_3)\, y \;=\; x^3 + a_2\, x^2 + a_4\, x + a_6 \qquad \text{com} \quad a_1, a_2, a_3, a_4, a_6 \in \mathbb{Q} \tag{129}$$

e procuremos determinar os pares (x, y) que pertencem a $\mathbb{Q}^2$.

Para iniciar este procedimento é necessário ter, pelo menos, dois pontos de coordenadas racionais (que podem não ser distintos). Agora a construção de um terceiro ponto a partir de dois outros pontos, $P = (x_1, y_1)$ e $Q = (x_2, y_2)$, passa pela determinação da recta que eles definem e, depois, pelo cálculo da intersecção dessa recta com a curva. Vamos descrever o mecanismo que permite calcular as coordenadas (x_3, y_3), em função das cordenadas de P e Q, do terceiro ponto R de intersecção da recta com a curva.

Se a recta é vertical, que se traduz por ser $x_1 = x_2 \wedge y_1 = -y_2$, o terceiro ponto de intersecção é um ponto especial, designado por *ponto no infinito* e representado por P_∞, que estudaremos na próxima secção.

Quando a recta não é vertical existem parâmetros a determinar $\lambda, \mu \in \mathbb{C}$ tais que todo o ponto (x, y), sobre recta, verifica $y = \mu + \lambda\, x$. Como os três pontos P, Q, R estão sobre a recta, tem-se

$$y_i \;=\; \mu + \lambda\, x_i \qquad \text{para} \quad i = 1, 2, 3 \tag{130}$$

Efectuando a substituição $y \longrightarrow \mu + \lambda\, x$ em (129) obtém-se

$$(\mu + \lambda\, x)^2 + (a_1\, x + a_3)\,(\mu + \lambda\, x) = x^3 + a_2\, x^2 + a_4\, x + a_6$$

Expandindo e agrupando os termos obtém-se

$$x^3 - (\lambda^2 + a_1\, \lambda - a_2)\, x^2 + \ldots \textit{monómios de ordem inferior} \;=\; 0$$

As três soluções desta equação são três ordenadas x_1, x_2, x_3 dos três pontos de intersecção da recta com a curva. Portanto esta mesma equação pode-se também escrever como $(x - x_1)\,(x - x_2)\,(x - x_3) \;=\; 0$. Dado que

$$(x - x_1)\,(x - x_2)\,(x - x_3) \;=\; x^3 - (x_1 + x_2 + x_3)\, x^2 + \ldots \textit{monómios de ordem inferior}$$

conclui-se

$$\lambda^2 + a_1\, \lambda - a_2 \;=\; x_1 + x_2 + x_3 \tag{131}$$

Uma vez que x_1 e x_2 são conhecidos, se λ for conhecido a equação (131) determina x_3. Além disso, sendo λ e x_3 conhecidos, as equações (130) determinam $y_3 = y_1 + \lambda\,(x_3 - x_1)$.

Para determinar λ temos duas situações possíveis:

$P \neq Q$

Sendo $(x_1, y_1) \neq (x_2, y_2)$, e sendo a recta não vertical (o que implica $x_1 \neq x_2$), então as equações (130) conduzem a

$$\lambda = (y_2 - y_1)/(x_2 - x_1) \tag{132}$$

$P = Q$

Neste caso a recta é tangente à curva no ponto (x_1, y_1); portanto λ é o declive da tangente nesse ponto; isto é, $\lambda = [\partial y/\partial x]\,(x_1, y_1)$. Derivando em ordem a x a equação (129), tem-se

$$(2\,y + a_1\,x + a_3)\;(\partial y/\partial x) + a_1\,y = 3\,x^2 + 2\,a_2\,x + a_4$$

Calculando esta derivada no ponto P, conclui-se

$$\lambda \;=\; (3\,x_1^2 + 2\,a_2\,x_1 - a_1\,y_1 + a_4)/(2\,y_1 + a_1\,x_1 + a_3) \tag{133}$$

Através de (132) (quando $P \neq Q$) ou através de (133) (quando $P = Q$) determinamos o parâmetros λ de uma recta não vertical que seja definida pelos dois pontos. Com (131) determinamos

$$x_3 \;=\; \lambda^2 + a_1\,\lambda - a_2 - x_1 - x_2 \qquad , \qquad y_3 \;=\; y_1 + \lambda\,(x_3 - x_1) \tag{134}$$

Estas relações definem o mecanismo computacional que, dados dois pontos racionais P e Q da curva em (129) determina um terceiro ponto racional R que é *colinear* com os dois pontos anteriores.

Note-se que, apesar de um ponto genérico $X = (x, y)$ que verifique a equação (131) ter coordenadas complexas, o mecanismo que acabámos de apresentar assegura que, sendo $P = (x_1, y_1)$ e $Q = (x_2, y_2)$ pontos de coordenadas racionais, o ponto $R = (x_3, y_3)$ também tem coordenadas racionais. De facto os parâmetros λ e μ calculados por (132) ou (133) são racionais e, desta forma, x_3 e y_3, calculados por (134), são necessariamante racionais.

O mecanismo de **colinearidade** determina uma relação ternária entre o três pontos de tal forma que, dados dois deles, é sempre possível calcular o terceiro. Por motivos que serão claros em seguida, vamos escrever essa relação da forma seguinte

$$P \oplus Q \oplus R = P_\infty$$

Para já não vamos dar significado especial ao símbolo "$\oplus$" (que será visto, apenas, como um separador de argumentos) e vamos interpretar "$\cdot = P_\infty$" apenas como um símbolo de predicado ternário. A notação apenas significa que os três pontos são colineares.

Se este fosse o único mecanismo para gerar pontos estaríamos bastante limitados já que, com os três pontos iniciais, o mecanismo permitiria gerar apenas dois pontos adicionais: o ponto $(2, 3)$, que é colinear com os pontos $P = (-1, 0)$ e $S = (0, 1)$, e o ponto $(2, -3)$ colinear com os pontos $(-1, 0)$ e $(0, -1)$.

Por isso são necessários outros mecanismos com esta função. O primeiro deles é óbvio: uma curva que seja definida por um polinómio onde o único termo em y tem grau 2 (um polinómio da forma $y^2 + f(x)$) então se $X = (x, y)$ é raiz do polinómio, também o ponto $(x, -y)$ é raiz do mesmo polinómio. Representamos este ponto por $-X$.

Temos agora uma nova transformação que mapeia pontos racionais da curva noutros pontos racionais da mesma curva: a aplicação $X \mapsto -X$ mapeia o ponto racional (x, y) no ponto racional $(x, -y)$. Esta aplicação designa-se por **simetria**.

O mecanismo da colinearidade parte do princípio que uma recta $y - \mu - \lambda\, x$ contém exactamente 3 pontos da curva $y^2 - x^3 - 1$. A figura 6 ilustra um conjunto de rectas que parecem contrariar esta assumpção.

A recta $y - 2\, x + 1$, que contém R e $-Q$ só parece conter estes dois pontos. A recta horizontal $y = 0$, que contém Q, não contém qualquer outro ponto da curva.

Por outro lado, a recta horizontal $y = 0$ (que passa por P) e as rectas verticais $x = 2$ (que passa por R e $-R$), $x = 0$ (que passa por Q e $-Q$) e $x = -1$ (que passa só por P) parecem conter exclusivamente os pontos indicados.

Tudo depende, porém, da forma como entendemos a noção de "ponto da curva" e como contamos esses pontos.

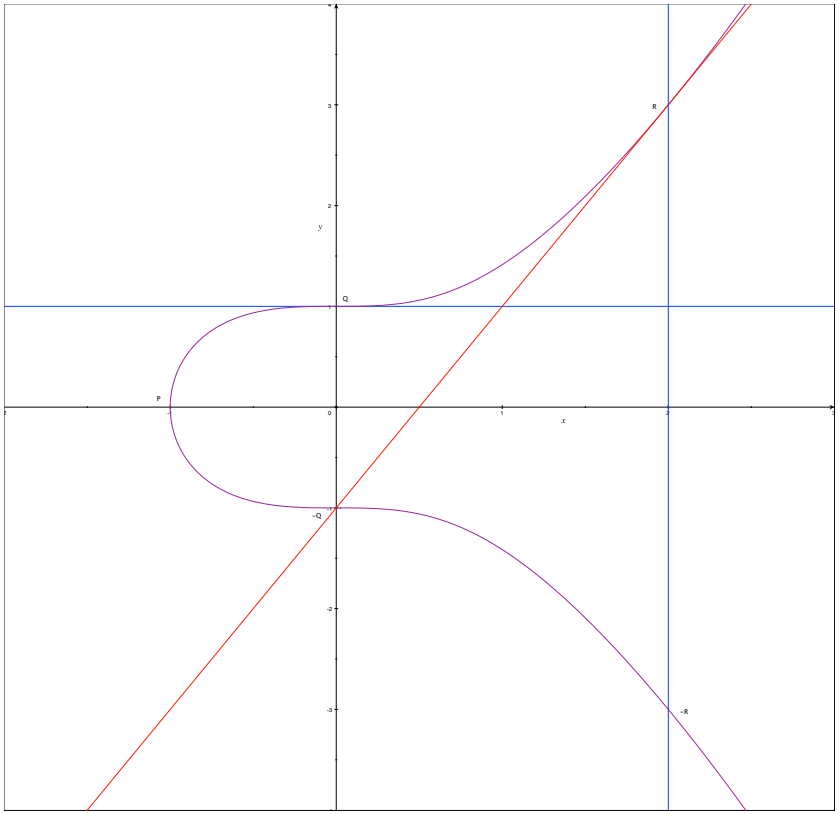


Figura 6: Ponto no infinito na curva $y^2 - x^3 - 1$.

Os pontos comuns a uma recta não-vertical $y + \lambda\, x + \mu = 0$ e à curva $y^2 - 1 - x^3 = 0$ são soluções deste sistema de equações. Substituindo a 1ª equação na segunda obtém-se

$$x^3 - \lambda^2\, x^2 - 2\,\lambda\,\mu\, x - \mu^2 + 1 \;=\; 0 \tag{135}$$

Este polinómio de 3º grau em x tem, no fecho algébrico de $\mathbb{Q}$ (i.e. os complexos $\mathbb{C}$), exactamente 3 raízes distintas. Pode ter uma raiz dupla quando a 1ª derivada também se anula nesse ponto, ou até uma raiz tripla se a 2ª derivada também se anular no mesmo ponto.

Raízes racionais múltiplas do polinómio dão origem a pontos onde a recta é tangente à curva. Quando a raiz é dupla (como no ponto $R = (2, 3)$ para a recta que também passa pelo ponto $-Q = (0, -1)$) interpretamos isso como se a recta intersecta-se duas vezes a curva nesse ponto. A relação de colinearidade deve, neste caso, escrever-se

$$(-Q) \;\oplus\; R \;\oplus\; R \;=\; P_\infty$$

A recta horizontal $y - 1 = 0$ (definida por $\lambda = 0$ e $\mu = -1$) dá origem a um polinómio muito simples; o polinómio (135) reduz-se a x^3 que tem uma raiz tripla no ponto $x = 0$. Neste caso a recta "intersecta" a curva 3 vezes no ponto $Q = (0, 1)$; a relação de colinearidade será, aqui,

$$Q \;\oplus\; Q \;\oplus\; Q \;=\; P_\infty$$

Outra situação deriva da existência de raízes complexas de (135). Por exemplo, a recta horizontal $y = 0$ (definida por $\lambda = \mu = 0$) conduz ao polinómio $x^3 + 1$ que tem uma raiz racional $x_1 = -1$ e duas raízes complexas

$x_2 = -\zeta$ e $x_3 = -\zeta^2$, em que $\zeta \neq 1$ é uma raíz cúbica da unidade[65]. A figura 6 indica apenas o ponto de intersecção $P = (-1, 0)$ definido pela raíz racional; os pontos de intersecção definidos pelas duas raízes complexas, $(-\zeta, 0)$ e $(-\zeta^2, 0)$, não são, aqui, representáveis.

Uma situação distinta ocorre com rectas verticais; tais rectas não podem ser descritas pelo polinómio $y + \lambda\, x + \mu$ [66] mas são descritas, simplesmente, por um polinómio da forma $x - \mu$. Os eventuais pontos racionais comuns à recta e à curva são determinados pelas possíveis raízes quadradas racionais de $1 + \mu^3$ com $\mu \in \mathbb{Q}$. Isto é, serão pontos da forma $(\mu, \pm\sqrt{1 + \mu^3})$ caso μ seja racional e a ráiz quadrada também seja racional.

Portanto uma recta vertical contém, quanto muito, duas raízes racionais do polinómio $y^2 - x^3 - 1$. No entanto, se acrescentar-mos ao conjunto de raízes um ponto extra por onde passam, por definição, todas as rectas verticais, resolve-mos a questão de ter sempre a propriedade da colinearidade estabelecida em triplos de pontos da curva.

Para justificar a introdução do **ponto no infinito** temos de recorrer a algum formalismo de Geometria Algébrica, o que faremos na próxima secção.

Vamos aceitar, para já, que um tal ponto existe, que é representado por P_∞ e por ele passam todas as rectas verticais. Nessa perspectiva a nossa curva vai ser constituída por duas componentes: a primeira é formada pela

[65] As raízes cúbicas complexas da unidade ζ são as raízes em $\mathbb{C}$ do polinómio $X^2 + X + 1$.

[66] Teria que ser $\lambda = \infty$.

raízes racionais do polinómio $\phi(x,y) = y^2 - x^3 - 1$, e se designa-se por "componente afim", e uma segunda componente formada exclusivamente pelo ponto P_∞.

Com esta definição de curva podemos verificar que, pelo menos para este exemplo, duas propriedades importantes:

1. Cada recta intersecta a curva em exatamente 3 pontos, desde que cada ponto conte tantas vezes quantas a respectiva multiplicidade e se entre em conta com o ponto no infinito P_∞ e pontos de coordendas complexas.
2. Cada recta (mesmo que seja vertical), se passa por dois pontos da curva de coordenadas racionais, passa sempre por um terceiro ponto de coordenadas racionais na mesma curva.

Por exemplo, a recta $x = 0$ passa pelos pontos $Q = (0,1)$ e $-Q = (0,-1)$; como é uma recta vertical passa também pelo ponto no infinito P_∞. A colinearidade exprime-se, aqui, por

$$Q \oplus (-Q) \oplus P_\infty = P_\infty$$

A recta vertical $x = -1$ é tangente à curva no ponto $P = (-1,0)$; passa, portanto, duas vezes por esse ponto. Como é vertical passa por P_∞; por isso a colinearidade é

$$P \oplus P \oplus P_\infty = P_\infty$$

8.1 Curvas Planas

A formalização do conceito de curva plana requer algumas noções elementares de Geometria Algébrica. Para não correrrmos o risco de enveredar-mos de forma excessiva por uma área da Matemática que, apesar de ser extremamente rica e interessante, tem objectivos que ultrapassam em muito o âmbito deste curso, vamos impor algumas limitações a esse estudo.

Assim, neste curso, vamos entender como "curvas planas" as curvas definidas no espaço bidimensional-dimensional pelas raízes de um polinómio a duas variáveis. Serão apenas estas o objecto do nosso estudo. Procuraremos, desta forma, evitar as complexidades de derivam do estudo das variedades algébricas. Procuraremos também, sempre que possível, usar o chamado sistema de coordenadas afins $\mathbb{A}^2$ e evitar um estudo detalhado de curvas em espaços projectivos.

Essencial ao nosso estudo é não impor limitações ao corpo $\mathbb{K}$ onde vão estar definidas as curvas. Apesar de as intuições geométricas serem mais óbvias em curvas definidas no plano real $\mathbb{R}^2$, não nos podemos esquecer que o nosso objectivo é estudar curvas com interesse criptográfico e isso implica, normalmente, usar outro tipo de corpos, nomeadamente corpos finitos. Como um polinómio de coeficientes no corpo $\mathbb{K}$ tem raízes no seu fecho algébrico $\overline{\mathbb{K}}$, é conveniente pensar, desde o início, em polinómios cujos coeficientes pertencem também a $\overline{\mathbb{K}}$.

Tomemos, então, um corpo $\mathbb{K}$ e $\overline{\mathbb{K}}[x, y]$ o anel dos polinómios a duas variáveis com coeficientes no fecho algébrico $\overline{\mathbb{K}}$ de $\mathbb{K}$. O conjunto dos polinómios $\overline{\mathbb{K}}[x, y]$ tem a estrutura algébrica de um anel. De facto estes polinómios têm

uma estrutura algébrica ainda mais rica: é também um **domínio de factorização única**; isto é, cada elemento do anel pode ser decomposto (de forma única a menos da ordem dos factores) no produto de um número finito de elementos irredutíveis.

□

Curvas planas são conjuntos de pontos que são, de alguma forma, "raízes" de um polinómio irredutível ϕ. Existem dois sistemas possíveis de representar estes pontos: em **coordenadas afins** ou em **coordenadas projectivas**.

Coordenadas Afins

Cada curva é determinada por um polinómio a duas variáveis $\phi(x, y)$ que é irredutível em $\overline{\mathbb{K}}[x, y]$.

Note-se que os coeficientes dos polinómios são elementos do fecho algébrico do corpo $\mathbb{K}$. Note-se também que um polinómio irredutível em $\mathbb{K}[x, y]$ pode não ser irredutível em $\overline{\mathbb{K}}[x, y]$.

Por exemplo, o polinómio $x^2 + 2y^2$ é irredutível em $\mathbb{Q}[x, y]$ mas não é irredutível no anel de polinómios sobre o fecho algébrico. De facto tem-se $(x^2 + 2\,y^2) = (x - i\,\sqrt{2}\,y)\,(x + i\,\sqrt{2}\,y)$ em $\mathbb{C}[x, y]$. Por isso $x^2 + 2\,y^2$ não define uma curva plana no espaço $\mathbb{Q}^2$.

Cada par $(a, b) \in \overline{\mathbb{K}}^2$ determina um ponto P em coordenadas afins. Cada polinómio ϕ mapeia pontos $P \in \overline{\mathbb{K}}^2$ em elementos de $\bar{\mathbb{K}}$ definindo $\phi(P)$ como $\phi(a, b)$. O ponto $P = (a, b) \in \overline{\mathbb{K}}^2$ é **raiz** de ϕ quando $\phi(P) = 0$.

Um polinómio da forma $(x-a)^i\,(y-b)^j$ é um **factor local** em P. O polinómio ϕ é **m-factorizável em P**, se é divisível por um factor local em P de grau m.

193 PROPOSIÇÃO
*Para toda a raíz P de ϕ, existem um inteiro $m \geq 1$ e uma decomposição $\phi = \phi_1 + \cdots + \phi_l$ em que todos os polinómios ϕ_i são m-factorizáveis em P. O maior de tais m designa-se por **multiplicidade** de ϕ em P e representa-se por $\eta_P(\phi)$.*

Este resultado é um corolário de um importante teorema da Álgebra, o Nullstellensatz, que estudaremos com um pouco mais detalhe na secção seguinte.

Note-se que não se exige que todos os polinómios ϕ_i, na decomposição de ϕ, tenham o mesmo factor de grau m. O que tem de ser comum a todas as componentes é o grau do factor e não o próprio factor.

EXEMPLO 39: Considere-se a origem $P = (0,0)$; um factor local em P de grau m é um polinómio da forma $x^i\,y^j$, com $i+j=m$.

Considere-se também o polinómio $\phi = 2\,x\,y + x^3$; obviamente que P é raiz de ϕ. O polinómio é a soma de duas componentes, $2\,x\,y$ e x^3 , ambas 2-factorizáveis em P. A primeira componente tem o factor local $x\,y$; a segunda tem o factor local x^2. Os factores locais em P são distintos, mas ambos têm grau 2.

Qualquer das componentes tem outros factores locais em P: ambas têm factores de grau 1 e a componente x^3 tem um factor de grau 3. Porém o grau 2 é o maior grau que é comum a factores locais em P de ambas as componentes.

Os polinómios x, y e $x+y$ têm todos uma raíz em P de multiplicidade 1. Isto é, $\eta_P(x) = \eta_P(y) = \eta_P(x+y) = 1$. Tem-se $\eta_P(x^2) = \eta_P(y^2) = \eta_P(x\,y) = 2$. Somando um polinómio de multiplicidade 1 com um de multiplicidade 2 (por exemplo, $x + x\,y$) obtém-se um polinómio de multiplicidade 1 em P. O polinómio $2\,x\,y + x^3$ tem, como vimos no exemplo 39, multiplicidade 2 em P.

Como resultado imediato da proposição 193 tem-se

194 TEOREMA
Para toda a raíz P de ϕ, existem polinómios p_{ij}, em que $p_{ij} \neq 0$ implica $p_{ij}(P) \neq 0$, tais que

$$\phi(x,y) \quad = \sum_{i+j=\eta_P(\phi)} (x-a)^i\,(y-b)^j\,p_{ij}(x,y) \tag{136}$$

Se $\eta_P(\phi) > 1$, os polinómios $\partial\phi/\partial x(x,y)$ e $\partial\phi/\partial y$ têm em P uma raíz de multiplicidade $\eta_P(\phi) - 1$. Consequentemente verifica-se $\eta_P(\phi) = 1$ se e só se $\partial\phi/\partial x(P) \neq 0$ e $\partial\phi/\partial y(P) \neq 0$.

A decomposição em (136) pode ser generalizada para polinómios com qualquer número finito de variáveis e, desta forma, pode-se estender a definição de multiplicidade de raíz (proposição 193) para este tipo de polinómios. Por exemplo, se for $\phi \in \mathbb{K}[x,y,z]$ e $P = (a,b,c)$ uma raíz de ϕ em $\mathbb{K}^3$, o polinómio decompõe em $\phi(x,y,z) = \sum_{i+j+k=m} (x-a)^i\,(y-b)^j\,(z-c)^k\,p_{ijk}(x,y,z)$; a multiplicidade de ϕ em P é o maior m para o qual existe esta decomposição de ϕ.

□

Nem todos os pontos das curvas são definidos por pares $(a, b) \in \overline{\mathbb{K}}^2$. Nomeadamente o comportamento assimptotico de curvas é expresso pela existência dos chamados "pontos no infinito".

Considere-se o caso simples das rectas; sabemos que uma recta no plano pode ser determinada por dois pontos distintos ou, em alternativa, por um ponto e um declive ("direcção"). Numa recta o declive pode ser infinito (se a recta for vertical) ou então, sendo finito, é um elemento de $\overline{\mathbb{K}}$.

O polinómio para uma recta que passe pelos pontos (a, b) e (a', b') é $(x - a)(b - b') - (y - b)(a - a')$. O polinómio para a recta que passa pelo ponto (a, b) tem declive μ, exige um pouco mais cuidado: se o declive for infinito (recta vertical) o polinómio é $(x - a)$; se μ for finito, o polinómio é $\mu(x - a) - (y - b)$.

Idealmente deveríamos ter apenas uma situação: uma recta é definida por dois pontos. Para isso, e para tentar unificar estas três situações, os matemáticos do século XVII introduziram a noção de **pontos no infinito**.

Nesta perspectiva cada declive μ (finito ou infinito) introduz um ponto no infinito P_μ; diz-se que uma recta passa pelo ponto P_μ se e só se tem declive μ. Uma curva C passa pelo ponto P_μ se tem uma assimptota com declive μ.

A unificação completa destas representações e um sistema de pontos que contenha os pontos no infinito só pode ser feito recorrendo às coordenadas projectivas. No entanto, mesmo nas coordenadas afins, interessa-nos ver o papel dos pontos no infinito na caractarização do comportamento assimptótico de curvas.

195 NOÇÃO

A **homogenização** *de* $\phi \in \mathbb{K}[x,y]$ *com grau total* d*, é o polinómio* $\phi_h \in \mathbb{K}[x,y,z]$ *tal que*

$$\phi_h(x,y,z)/z^d \;=\; \phi(x/z, y/z) \tag{137}$$

Diz-se que $\phi(x,y)$ *tem uma raiz de multiplicidade* m *em* P_∞ *quando* $\phi_h(z,y,z)$*, tem uma raiz de multiplicidade* m *em* $(0,1,0)$*; identicamente, para* μ *finito, diz-se que* ϕ *tem uma raiz de multiplicidade* m *em* P_μ *quando* $(1,\mu,0)$ *for uma raiz de multiplicidade* m *de* $\phi_h(z,y,z)$*.*

Tento em atenção que $\phi(x,y) = \phi_h(x,y,1)$, constata-se que as raizes (x,y) de ϕ são precisamente as raízes de ϕ_h da forma $(x,y,1)$. Portanto ϕ_h captura não só todas as raízes afins de ϕ como também as raízes no infinito. Este incremento em representatividade tem, obviamente, um custo: a variável adicional z. Para ϕ, as raízes procuram-se num espaço a duas dimensões; ao invés as raízes de ϕ_h procuram-se num espaço a três dimensões.

EXEMPLO 40:

1. Uma recta $\phi = a\,x + b\,y + c$ homogeniza em $\phi_h = z\,(a\,x/z + b\,y/z + c) \;=\; a\,x + b\,y + c\,z$. Temos $\phi_h(0,1,0) = b$ e $\phi_h(1,\mu,0) = a + b\,\mu$. Portanto P_∞ é raíz da recta se e só se $b = 0$; i.e., se a recta é vertical e passa pelo ponto $x = -c/a$. Se $b \neq 0$, P_μ é raíz da recta se for $\mu = -a/b$.
2. O polinómio $\phi = y^2 + x^3 + x\,y + 1$ tem grau total é 3 e a sua homogenização é

$$\phi_h \;=\; z^3\,((y/z)^2 + (x/z)^3 + (x/z)\,(y/z) + 1) \;=\; y^2\,z + x^3 + x\,y\,z + z^3$$

Tem-se $\phi_h(0,1,0) = 0$ e $\phi_h(1,\mu,0) = 1$; portanto ϕ tem P_∞ como raíz de ϕ mas nenhum P_μ, com μ finito, é raíz.

Não é possível construir ϕ_h como uma soma de múltiplos de monómios $x^i\,(y-1)^j\,z^k$ cujo grau total $i+j+k$ seja 2 ou superior; assim a multiplicidade da raíz P_∞ é, apenas, 1.

3. Considere-se finalmente $\phi = x^2\,y + x$ cujo grau total é 3 e tem homogenização $\phi_h = x^2\,y + x\,z^2$. Tem-se $\phi_h(0,1,0) = 0$ e $\phi_h(1,\mu,0) = \mu$. Portanto P_∞ e P_0 são ambas raízes no infinito de ϕ.

   Claramente, a multiplicidade de ϕ_h é 2 em $(0,1,0)$ (atente-se à forma $\phi_h = x^2\,p + z^2\,q$, com $p = y$ e $q = x$) e é 1 em $(1,0,0)$ (atente-se à forma $\phi_h = (x-1)\,p + y + z\,q$, com $p = (x+1)\,y$ e $q = x\,z$).

196 Noção

A **curva plana** *em* $\mathbb{A}^2$, *definida por um polinómio* $\phi \in \mathbb{K}[x,y]$ *que é irredutível em* $\overline{\mathbb{K}}\,[x,y]$, *é o conjunto formado pelas raízes afins ou no infinito de* ϕ. *Um* **ponto singular** *é uma raíz de* ϕ *com multiplicidade* > 1. *A curva diz-se* **não-singular** *se não contém pontos singulares. Se* K *é uma qualquer extensão de* $\mathbb{K}$, *os pontos* K-**racionais** *da curva são os pontos afins de coordenadas* $(x,y) \in K^2$.

Notas

1. **Curvas Triviais**

   Os polinómios 1 e 0 são ambos irredutíveis e definem duas "curvas" triviais. O polinómio 1 não tem qualquer raíz; por isso, a "curva" é o conjunto vazio de pontos $\emptyset$. O polinómio 0, ao invés, tem como raizes qualquer ponto $(x,y) \in \overline{\mathbb{K}}^2$ e qualquer ponto no infinito; é o espaço total que representamos por $\mathbb{P}^2$.

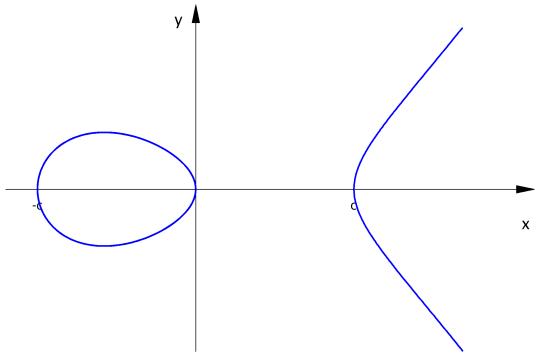
2. **Pontos Singulares**

   Para detectar pontos singulares pode-se usar o teorema 194 e o critério das derivadas parciais.

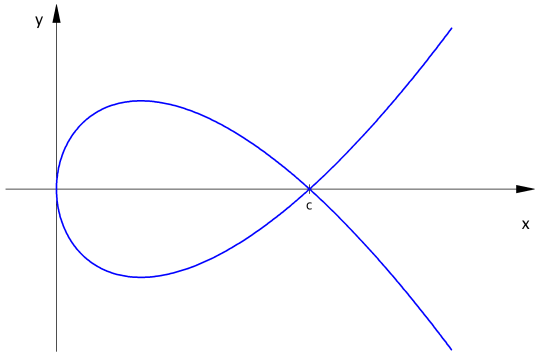
   Por exemplo, cúbica $\phi = y^2 - x\,(x^2-1)$ define uma curva plana formada por todos os pontos (x,y) que são raízes deste polinómio e ainda pelo ponto P_∞ já que, se verifica facilmente, o polinómio tem essa raiz no infinito.

Note-se que $\partial\phi/\partial x = 1 - 3\,x^2$ e $\partial\phi/\partial y = 2\,y$; os únicos pontos que são raízes de ambas as derivadas são $(\pm\sqrt{1/3}, 0)$. Porém nenhum destes pontos pertence à curva; por isso ela é não-singular.

Já o polinómio $\phi' = y^2 - x\,(x-1)^2$ tem derivadas parciais $\partial\phi'/\partial y = 2\,y$ e $\partial\phi'/\partial x = (x-1)\,(1-3\,x)$. As raízes comuns a ambas estes dois polinómios são os pontos $(1,0)$ e $(1/3,0)$. Note-se que $(1,0)$ é um ponto da curva; por isso ela é singular.



$y^2 - x\,(x^2-1)$



$y^2 - x\,(x-1)^2$

3. **Pontos Racionais**

É preciso ter em conta que as raízes afins de $\phi \in \mathbb{K}[x,y]$ podem ter coordenadas fora do corpo $\mathbb{K}$. Tome-se, por exemplo, $\mathbb{K} \equiv \mathbb{Q}$ e considere-se $\phi = y^2 - x^3 - 1$. Genericamente as raízes afins de ϕ têm coordendas complexas, uma vez que $\overline{\mathbb{Q}} \equiv \mathbb{C}$.

Fixe-se um racional qualquer b e procure-se pontos afins da curva da forma (a,b). O valor de a tem de ser raíz do polinómio $x^3 - (b^2-1)$. Se for $b^2 \neq 1$ existem três raízes distintas deste polinómio: um valor algébrico, $a = \sqrt[3]{b^2-1}$, e dois valores complexos $a\,\zeta$ e $a\,\zeta^2$, sendo ζ uma raíz cúbica complexa da unidade. Se for $b^2 = 1$ o polinómio tem uma raíz tripla em 0. A menos deste último caso, as raízes de ϕ da forma (a,b), com b racional, muito provavelmente não têm uma cordenada a que seja racional: duas são complexas e uma é algébrica, provavelmente irracional.

Existem, no entanto, raízes racionais do polinómio $x^3 - (b^2-1)$, para determinados valores de b. Por exemplo, para $b = 0$, temos

uma raíz racional $a = -1$; para $b = 3$ temos a raíz $a = 2$, etc. Estes pontos, $(-1, 0), (2, 3)$, etc, são pontos racionais da curva.

Seja C a curva plana determinada pelo polinómio ϕ; esse facto denota-se por $C : \phi$. Uma maneira de interpretar a curva C é através do conjunto formado por todos os polinómios que se anulam em todos $P \in C$.

$$\mathbf{I}(C) \;=\; \{\, f \in \overline{\mathbb{K}}\,[x, y] \;\mid\; f(P) = 0 \quad \text{para todo} \quad P \in C \,\} \tag{138}$$

É fácil verificar que o conjunto $\mathbf{I}(C)$ é um ideal; isto é, é fechado por somas e por multiplicação por um qualquer polinómio. O facto de ϕ ser irredutível em $\overline{\mathbb{K}}\,[x, y]$ assegura que o ideal é primo; isto é, se $f \cdot g$ pertence ao ideal, um dos polinómios f ou g tem de pertencer ao ideal. Veremos adiante (ver noção 215, página 493) uma exposição sucinta da noção de ideal e suas aplicações à Teoria das Curvas.

O anel quociente $\overline{\mathbb{K}}\,[x, y]/\mathbf{I}(C)$ identifica como equivalentes dois polinómios que são iguais em todos os pontos da curva; isto é, $p \sim q$ sse $p - q \in \mathbf{I}(C)$ ou, equivalentemente, sse $p(P) = q(P)$ para todo $P \in C$. Este anel representa-se por $\mathbb{A}(C)$ e designa-se por **anel afim** ou **anel de coordenadas** da curva C.

As noções de m-factorização e multiplicidade podem ser estendidas a

197 NOÇÃO
Seja ϕ um polinómio que tem uma raíz P sobre uma curva C. Representamos por $\eta_P(\phi; C)$, e designa-se por

multiplicidade *de* ϕ *em* P *sobre* C*, o maior* m *para o qual existe um polinómio* $u \in \mathbf{I}(C)$ *tal que* $\phi - u$ *tem uma raíz de multiplicidade* m *em* P*.*

Quando $\eta_P(\phi; C) = 1$*, então* ϕ ***intersecta*** *a curva* C *em* P*; se* $\eta_P(\phi; C) > 1$ *,* ϕ *é* ***tangente*** *a* C *em* P*.*

Comparando com a noção de multiplicidade simples (proposição 193) vemos que a mudança essencial está no facto de não se exigir que ϕ tenha multiplicidade m em P mas, em vez disso, exigir-se que a diferença $\phi - u$, para algum $u \in \mathbf{I}(C)$, tenha essa multiplicidade. Desta forma a multiplicidade de ϕ em P, $\eta_P(\phi)$, é equivalente à multiplicidade $\eta_P(\phi; 0)$ de ϕ em P sobre a curva trivial definida pelo polinómio 0.

EXEMPLO 41: Considere-se a recta $\phi = (y - 1)$. Seja C a curva definida pelo polinómio $\psi = y^2 - 1 - x^3$. O ponto $P = (0, 1)$ é raíz de ϕ e de ψ; por isso é um ponto de C comum com a curva definida por ϕ.

Com um pouco de manipulação pode-se constatar que

$$(y - 1) \; - \; \frac{1}{4}(3 - y)(y^2 - 1 - x^3) \quad = \quad \frac{1}{4}(3 - y)\, x^3 + \frac{1}{4}(y - 1)^3$$

O lado direito da igualdade é um polinómio com uma raíz em P de multiplicidade 3 (atente-se aos factores locais x^3 e $(y - 1)^3$). O lado esquerdo é uma diferença da forma $\phi - u$ para um polinómio $u = \frac{1}{4}(3 - y)\,\psi$ que, por ser múltiplo de ψ, é um elemento de $\mathbf{I}(C)$.

Consequentemente, atendendo à definição, o polinómio $y - 1$ tem uma multiplicidade 3 em P sobre a curva C. De facto $(y - 1)$ representa uma recta tangente à curva C onde o ponto de contacto P tem multiplicidade 3.

A noção de intersecção ou contacto de duas curvas é caracatrizada por um importante teorema[67].

198 TEOREMA (BEZOUT)
Sejam $C:\phi$ e $D:\psi$ duas curvas distintas. Então $C \cap D$ é um conjunto finito e verifica-se

$$\sum_{P \in C \cap D} \eta_P(\phi;\psi) \quad = \quad \sum_{P \in C \cap D} \eta_P(\psi;\phi) \quad = \quad \deg(\phi) * \deg(\psi) \tag{139}$$

É importante ter-se em atenção que nas curvas C e D estão incluídos não só os pontos afins como os pontos no infinito. Se uma das curvas (por exemplo $C:\phi$) for uma recta, tem grau 1 e, por isso, a soma (139) é igual ao grau do polinómio ψ. Isso significa que uma recta contacta uma curva ψ em tantos pontos (incluindo os pontos no infinito e contando cada ponto tantas vezes quantas a sua multiplicidade) quantos o grau de ψ.

EXEMPLO 42:
Considere-se a curva elíptica $\psi = y^2 - x^3 - 1$. Como o grau de ψ é 3, o teorema de Bézout diz-nos que qualquer recta ϕ contacta a curva em exactamente 3 pontos.

Por exemplo, recta $\phi = y$ intersecta a curva em 3 pontos distintos: o ponto racional $(-1,0)$ e dois pontos de ordenada complexa $(-\zeta,0)$ e $(-\zeta^2,0)$, sendo ζ uma raíz cúbica, complexa da unidade. Todos eles têm multiplicidade 1.

[67] Para prova ver HARTSHORNE, *Algebraic Geometry*.

A recta $\phi = y - 1$ contacta a curva ψ no ponto $P = (0, 1)$ e, como vimos no exemplo 41, a multiplicidade do contacto é 3. Portanto esta recta não contacta a curva em qualquer outro ponto.

A recta $\phi = x$ contacta a curva em dois pontos afins $(0, 1)$ e $(0, -1)$ e ainda no ponto do infinito P_∞.

Coordenadas Projectivas

Desde pelo menos o século XVII que os matemáticos se aperceberam que, adicionando certos pontos fictícios a rectas e outras curvas, a geometria Euclidiana poderia ser muito simplificada. Como exemplo, considere-se um par de asserções duais da geometria plana clássica:

(1) Duas rectas distintas determinam um único ponto: o seu ponto de intersecção.

(2) Dois pontos distintos determinam uma única recta: a recta que passa por ambos os pontos.

A asserção (1) não é válida quando as rectas são paralelas; esta excepção pode ser resolvida assumindo que rectas contêm um "ponto no infinito" e que as rectas paralelas se intersectam nesse "ponto no infinito"; a asserção, agora, é universalmente válida.

Para que a 2ª asserção continue válida com a introdução dos pontos no infinito temos de assumir que um ponto no plano e um ponto no infinito determinam também uma única recta. Isto faz supor que "ponto no infinito" seja equivalente ao conceito de "direcção" ou "inclinação" das rectas: um ponto no plano e uma directação determinam, realmente, uma única recta. Do mesmo modo, para que (2) continue a ser válida com dois pontos no infinito

distintos (duas direcções diferentes), somos levados naturalmente à conclusão que todos os pontos no infinito estão colocados sobre uma mesma recta e que tal recta só contém pontos no infinito; isto é, existe uma recta totalmente situada no infinito.

Estes conceitos têm resultados algébricos importantes; no entanto, em coordenadas afins, são difíceis de visualizar e conduzem a noções pouco naturais; por exemplo, pontos que são direcções. As coordenadas projectivas apareceram nos princípios do século XIX para ser possível lidar facilmente com este tipo de situações sem ter necessidade de introduzir interpretações "estranhas" para certos pontos, rectas ou outras curvas e todas estas entidades serem representados de uma única forma.

Esta representação unificada exige uma representação das entidades (pontos, rectas e curvas) segundo vários pontos de vista que são, de alguma forma, equivalentes. Nomeadamete, para representação de pontos, não basta apenas um tuplo de coordenadas (como nas coordenadas afins) mas vários tuplos ligados por uma relação de equivalência.

199 Noção

Representa-se por $\mathbb{P}^2$ *o conjunto das rectas em* $\mathbb{K}^3$ *que passam pela origem. Os elementos de* $\mathbb{P}^2$ *designam-se por* ***pontos projectivos*** *ou* ***pontos em coordenadas projectivas*** *de dimensão 2.*

Cada recta em $\mathbb{K}^3$ que passa pela origem é determinada por por um polinómio da forma $a\,x + b\,y + c\,z$ em que pelo menos um dos coeficientes (a, b, c) é diferente de zero. Note-se que a mesma recta pode ser representada por outro polinómio $a'\,x + b'\,y + c'\,z$ desde que se verifique $a = \lambda\,a' \wedge b = \lambda\,b' \wedge c = \lambda\,c'$ para algum $\lambda \neq 0$.

Esta observação conduz-nos a uma forma alternativa de definir $\mathbb{P}^2$ através de uma relação de equivalência sobre triplos de coordenadas. Considere-se triplos $P = (a, b, c)$ e $Q = (a', b', c')$ em $\mathbb{K}^3 \setminus \{(0, 0, 0)\}$ (i.e., pelo menos uma das componentes de cada tuplo é $\neq 0$); defina-se a a seguinte relação nesse espaço

$$P \sim Q \quad \Leftrightarrow \quad (\exists \lambda \neq 0)\, [a = \lambda a' \ \wedge \ b = \lambda b' \ \wedge \ c = \lambda c'] \tag{140}$$

A relação $\sim$ é claramente uma relação de equivalência. Os pontos em coordenadas projectivas são as classes de equivalência definidas em $\mathbb{K}^3 \setminus \{(0, 0, 0)\}$ por esta relação de equivalência.

200 NOÇÃO

No contexto de $\mathbb{P}^2$ *os* **pontos afins** *são as classes de equivalência que contêm triplos da forma* $(x, y, 1)$ *. Os* **pontos no infinito** *são classes de equivalência que contém triplos da forma* $(x, y, 0)$ *, em que* $x \neq 0$ *ou* $y \neq 0$*; nomeadamente* P_∞ *designa o ponto determinado pelo triplo* $(0, 1, 0)$ *e, para cada* $\mu \in \overline{\mathbb{K}}$*,* P_μ *designa o ponto no infinito determinado pelo triplo* $(1, \mu, 0)$ *.*

Um polinómio $\phi \in \overline{\mathbb{K}}\,[x, y, z]$ em que todos os monómios têm o mesmo grau d diz-se **homogéneo** de grau d. Um tal polinómio verifica $\phi(\lambda\, x, \lambda\, y, \lambda\, z) = \lambda^d\, \phi(x, y, z)$ para todo λ e todo triplo (x, y, z). Por isso se um triplo (x, y, z) é raiz do polinómio, qualquer outro triplo que lhe seja equivalente é também raiz do polinómio.

Um ponto P é **raiz** de um polinómio homogéneo quando existe um representante desta classe que é raiz do polinómio. Se tal acontecer, então (como vimos) qualquer outro triplo que lhe seja equivalente é também raiz do mesmo polinómio. Nestas circunstância escreve-se $\phi(P) = 0$.

Por exemplo, P_∞ é raíz do polinómio homogéneo $z\,y^2 + x^3 + x\,z^2$.

201 NOÇÃO

Um polinómio homogéneo $\phi \in \mathbb{K}[x, y, z]$ *que seja irredutível no fecho algébrico* $\overline{\mathbb{K}}\,[x, y, z]$ *determina uma* **curva plana** *em coordenadas projectivas (ou* **curva projectiva***) definida como o conjunto das raízes desse polinómio. A curva é* **singular** *quando existe um ponto da curva que é raiz, simultaneamente, das três derivadas parciais* $\partial\phi/\partial x$, $\partial\phi/\partial y$ *e* $\partial\phi/\partial z$.

Na representação de pontos, a vantagem das coordenadas projectivas está no facto de situações exceptionais (como o ponto no infinito) não exigirem nenhum tratamento especial; todos os pontos são referenciados do mesmo mod. A desvantagem está no facto de precisarmos de 3 coordenadas (em vez de 2) para definir o ponto e esse triplo de coordenadas ser apenas um representante da classe de equivalência que determina o ponto. Isto significa que qualquer propriedade que quisermos mostrar para um ponto tem de ser invariante pela multiplicação das coordenadas por um factor de escala $\lambda \neq 0$ arbitrário.

Uma consequência desta exigência é o facto de apenas se poder usar polinómios homogéneos. Enquanto que nas coordenadas afins qualquer polinómio irredutível definia uma curva, nas coordenadas afins só os polinómios homogéneos definem curvas.

□

Vamos colocar de novo a questão de curvas projectivas em $\mathbb{P}^2$ sobre um corpo algebricamente fechado K.

Como vimos na noção 201 na página 482, cada curva é determinada por um polinómio homogéneo e irredutível $\phi \in K[x,y,z]$ irredutível[68]. Dada uma curva $C:\phi$, representa-se por $\mathbf{I}(C)$ o seu ideal

$$\mathbf{I}(C) \;=\; \{\, p \in K\,[x,y,z] \;\mid\; p(P)=0 \text{ para todo } P \in C \,\} \tag{141}$$

Como consequência do Nullstellensatz, do facto de K ser algebricamente fechado e ϕ ser irredutível, tem-se

202 FACTO
Tem-se $p \in \mathbf{I}(C)$ se e só se p é divisível por ϕ.

Quando passamos à segunda parte desta definição (a noção de curva não-singular) surge a exigência de as três derivadas principais de ϕ não se anularem simultânemamente em nenhum ponto da curva. Para vermos o alcance desta restricção, é importante ver o seguinte morfismo e os resultados seguintes.

203 NOÇÃO
Seja C uma curva projectiva não-singular determinada por um polinómio homogéneo ϕ; seja d o seu grau. O morfismo $\mathcal{J}: C \longrightarrow \mathbb{P}^2$ determinado pelo triplo de polinómios homogéneos de grau $d-1$

$$\mathcal{J} \;=\; [\,\partial\phi/\partial x \;,\; \partial\phi/\partial y \;,\; \partial\phi/\partial z\,] \tag{142}$$

designa-se por ***jacobiano*** *de C.*

[68] Atente-se que, neste caso, K coincide com o seu fecho algébrico.

Porque C é não-singular, em qualquer zero do polinómio ϕ pelo menos uma das três componentes de $\mathcal{J}^C$ não se anula. Por isso $\mathcal{J}$ define realmente um morfismo.

204 TEOREMA
Seja $\mathcal{J}$ o jacobiano da curva projectivas C; então, imagem $\mathcal{J}^C(C)$ define em $\mathbb{P}^2$ uma curva projectiva que designamos por **curva dual** *de C.*

205 LEMA *Se $\phi \in K[x,y,z]$ é um qualquer polinómio homogéneo de grau d, verifica-se*

$$x\,\partial\phi/\partial x + y\,\partial\phi/\partial y + z\,\partial\phi/\partial z \quad = \quad d\cdot\phi \tag{143}$$

Prova O polinómio pode-se escrever como $\phi = \sum_{i+j+k=d} a_{ijk}\,x^i\,y^j\,z^k$. Temos

$$x\,\partial\phi/\partial x \;=\; \sum_{i+j+k=d} i\cdot a_{ijk}\,x^i\,y^j\,z^k$$

e formas análogas para $y\cdot\partial\phi/\partial y$ e $z\cdot\partial\phi/\partial z$. Donde

$$x\,\partial\phi/\partial x + y\,\partial\phi/\partial y + z\,\partial\phi/\partial z \;=\; \sum_{i+j+k=d} (i+j+k)\cdot a_{ijk}\,x^i\,y^j\,z^k \;=\; d\cdot\phi$$

Curvas definidas por polinómios do 1º grau, $a\,x + b\,y + c\,z$ designam-se por **rectas projectivas**.

Claramente, cada triplo $(a, b, c) \in \mathbb{K}^3$ determina uma recta projectiva a menos da relação de equivalência nos pontos de $\mathbb{P}^2$; isto é, os triplos (a, b, c) e $(\lambda\, a, \lambda\, b, \lambda\, c)$, com $\lambda \neq 0$, determinam exactamente a mesma recta. Consequentemente

206 FACTO
Existe um isomorfismo entre $\mathbb{P}^2$ *e o conjunto de todas as rectas projectivas em* $\mathbb{P}^2$*, isomorfismo esse que ao ponto* $P = [a, b, c]$ *faz corresponder a recta definida por* $a\, x + b\, y + c\, z$.

A recta projectiva (e o respectivo polinómio homogéneo de 1º grau) determinados por $P \in \mathbb{P}^2$ são, aqui, representados por $\mathbf{l}(P)$.

207 NOÇÃO
A recta $\mathbf{l}(\mathcal{J}^C(P))$ *designa-se por* **tangente** *à curva* C *no ponto* P.

□

No espaço afim $\mathbb{A}^3$ uma curva projectiva C determina uma superfície cónica com vértice na origem. Considere-se o ideal $\mathbf{I}(C)$ e o anel afim $\mathbb{A}^3(C)$. Recorde-se que este anel é definido como o quociente $K\,[x, y, z]\,/\mathbf{I}(C)$.

No anel afim $\mathbb{A}^3(C)$, a noção de multiplicidade de um polinómio p num ponto $P \in \mathbb{A}^3$ sobre a superfície C é definido da forma usual.

Sumariamente: um **factor local** de $P = (a, b, c)$ é um polinómio da forma $(x-a)^i\,(y-b)^j\,(z-c)^k$; p é m-**factorizável** em P se é divisível por um factor local em P de grau m; a **multiplicidade** de p em P é o maior $m \geq 0$ tal que p é decomponível numa soma de polinómios m-factorizáveis em P. O Nullstellensatz assegura que p tem uma raíz em P se e só se tem multiplicadade maior que zero nesse ponto.

Finalmente, se $P \in C$, a **multiplicidade** de $p \notin \mathbf{I}(C)$ em P **sobre** C, representada por $\eta_P(p; C)$, é a maior multiplicidade em P de polinómios u tais que $p - u \in \mathbf{I}(C)$.

Se $p(P) \neq 0$, convencionamos que $\eta_P(p; C) = 0$. Se $p \in \mathbf{I}(C)$ convencionamos que $\eta_P(p; C) = \infty$.

208 PROPOSIÇÃO
Seja $p \in K[x, y, z]$ um polinómio homogéneo e $C : \phi$ uma curva projectiva. Então, para todos $P = (a, b, c) \in K^3$ e $\lambda \neq 0$, tem-se $\eta_P(p; C) = \eta_{\lambda P}(p; C)$.

Prova Se $\eta_P(p; C) = m$ então existem polinómios u, v tais que $p = u\,\phi + v\,f$ sendo f um factor local em $P = (a, b, c)$ de grau m. Isto é, $f = (x-a)^i\,(y-b)^j\,(z-c)^k$ com $i + j + k = m$. Seja H a aplicação que mapeia qualquer polinómio $h(x, y, z)$ em $\lambda^m\,h(x/\lambda, y/\lambda, z/\lambda)$. Se h for homogéneo de grau d tem-se $H(h) = \lambda^{m-d}\,h$. Verifica-se facilmente que $f' = H(f)$ é um factor local em λP de grau m.

Consequentemente $H(p) = H(u)H(\psi) + H(v)\,H(f)$ conduz à igualdade $p = u'\,\phi + v'\,f'$ já que tanto p como ϕ são homogéneos. Consequentemente a multiplicidade de p sobre C em λP é, pelo menos, m. Dada a simetria da afirmação, terá de ser exactamente m.

Uma multiplicidade importante é a multiplicidade da tangente a uma curva no ponto de tangência e sobre a curva. Em consequência do lema 205, a tangente à curva C no ponto P, contém sempre esse ponto P. Portanto a tangente intersecta a curva. De facto, tem-se

209 LEMA *Seja $t_P = \mathbf{l}(\mathcal{J}^C(P))$ a recta tangente à curva C no ponto P. Então tem-se sempre $\eta_P(t_P; C) > 1$.*

EXEMPLO 43: Considere-se a curva C definida por $y^2\,z - x^3 - z^3$. São pontos da curva,

$$P = [0,1,1] \qquad Q = [1,0,1] \qquad P_\infty = [0,1,0]$$

Pretende-se determinar as tangentes nesses pontos assim como a multiplicidade respectiva. Nesta curva tem-se

$$\mathfrak{D} \;=\; [-3\,x^2, 2\,y\,z, y^2 - 3\,z^2] \qquad \mathcal{J}^C(P) = [0,-1,1] \qquad \mathcal{J}^C(Q) = [1,0,1] \qquad \mathcal{J}^C(P_\infty) = [0,0,1]$$

As tangentes respectivas são as rectas

$$t_P \;=\; z - y \qquad t_Q \;=\; x + z \qquad t_{P_\infty} \;=\; z$$

Como o polinómio tem grau 3 e as rectas tangente têm grau 1, o teorema de Bézout (teorema **??**) diz-nos que a multiplicidade não excede 3. O lema 209 diz-nos que a multiplicidade é > 1.

As funções racionais em coordenadas projectivas são definidas, também, através de fracções de polinómios homogéneos com o mesmo grau.

210 NOÇÃO
Seja C/K uma curva projectiva em $\mathbb{P}^2$ e K um corpo algebricamente fechado. Um par de polinómios $f, g \in K[x, , y, z]$ determina uma **fracção homogénea** *quando ambos são homogéneos e têm o mesmo grau.*

O espaço $K(C)$ *das* ***funções racionais*** *sobre* C *é o espaço quociente definido no conjunto das fracções homogéneas pela relação de equivalência*

$$f/g \sim p/q \quad \Leftrightarrow \quad f\,q - g\,p \in \mathbf{I}(C)$$

211 DEFINIÇÃO
A ***ordem*** *de* $f = p/q \in \mathbb{K}(C)^*$ *(isto é,* $f \neq 0$*) em* P*, representado por* $\mathsf{ord}_P(f)$ *, é a diferença*

$$\mathsf{ord}_P(f) \;=\; \eta_P(p;C) - \eta_P(q;C)$$

Se for $\mathsf{ord}_P(f) > 0$ *diz-se* f *tem um* ***zero*** *de ordem* $\mathsf{ord}_P(f)$ *em* P*; se for* $\mathsf{ord}_P(f) < 0$ *diz-se que* f *tem um* ***pólo*** *de ordem* $-\mathsf{ord}_P(f)$ *em* P*.*

Por convenção, a função racional nula $f = 0$ *tem um zero de ordem* ∞ *em todo o ponto de* C*.*

É fácil verificar que estas noções são independentes do representante p/q escolhido para a função racional f. Tem-se também, para todo o par de funções racionais $f, g \in \bar{\mathbb{K}}(C)^*$ e todo o ponto P da curva C

$$\mathsf{ord}_P(f\,g) \;=\; \mathsf{ord}_P(f) + \mathsf{ord}_P(g) \tag{144}$$

O seguinte resultado é essencial para o entendimento do papel das funções racionais e pode ser facilmente demonstrado. Vamos considerar uma curva projectiva $C/\mathbb{K}$ e uma qualquer função racional $f \in \bar{\mathbb{K}}(C)^*$. Então

212 FACTO

A função f tem um número finito de zeros e de pólos em C e a ordem de cada zero ou pólo é finita. Se f não for uma função constante sobre C (isto é, $f \notin \mathbb{K}^*$) então, pelo menos num ponto $P \in C$ tem-se $\mathrm{ord}_P(f) \neq 0$. *Adicionalmente verifica-se sempre*

$$\sum_{P \in C} \mathrm{ord}_P(f) \;=\; 0$$

Este resultado diz-nos que o número de pólos de f (cada um contanto tantas vezes quanto a sua ordem) tem de ser igual ao número de zeros. Diz-nos também que, a menos do caso trivial das funções constantes sobre na curva[69], existem sempre pólos e zeros e em número finito.

□

Frequentemente interessa-nos resolver o problema inverso:

> *dado um conjunto de pontos eventuais pólos $P_1, P_2, \cdots, P_n$ e de eventuais zeros $Z_1, Z_2, \cdots, Z_n$ quere-se construir uma função racional f que tenha exactamente estes pólos e estes zeros*

Essencialmente quere-se $f = p/q$ definindo dois polinómios p e q homogéneos e com o mesmo grau; o primeiro deve ter os pontos Z_i como raízes e o segundo deve ter os pontos P_i como raízes.

[69] Note-se que f pode ser constante sobre C sem ser uma função constante; por exemplo, se ϕ determinar a curva C, a fracção $(\lambda + \phi)/(\mu + \phi)$, com $\lambda, \mu \in \mathbb{K}^*$, não é constante mas determina uma função racional que é constante sobre C.

Vamos começar por considerar versões simplificadas deste problema começando por polinómios homogéneos lineares; isto é **rectas**. Para isso precisamos de algumas ferramentas que nos ajudem a construir e manipular rectas.

213 DEFINIÇÃO
Sejam $P, Q \in \mathbb{P}^2$ *determinados por representantes* (a, b, c) *e* (a', b', c') *respectivamente. Se* $P \neq Q$ *define-se*

$$P \otimes Q \doteq [bc' - cb' \ , \ ca' - ac' \ , \ ab' - ba'] \tag{145}$$

Representa-se por $\mathrm{l}(P)$ *o polinómio homogéneo do* 1^o *grau* $a\,x + b\,y + c\,z$ *ou, indistintamente, a recta definida por isso polinómio.*

Facilmente se verifica que $P \otimes Q$ e $\mathrm{l}(P)$ são independentes do representante escolhido para os pontos P e Q.

Como $P \otimes$ Q só está definido para pontos distintos, pode-se estender a definição acrescentando um ponto $\mathfrak{O}$ extra ao espaço $\mathbb{P}^2$ (identificado com o triplo de coordenadas todas nulas $(0, 0, 0)$) e fazendo $P \otimes P = P \otimes \mathfrak{O} = \mathfrak{O}$. Também $\mathrm{l}(\mathfrak{O}) \doteq \mathbb{P}^2$. Nestas circunstâncias

214 PROPOSIÇÃO
O operador $\otimes$ *definido em* $\mathbb{P}^2 \cup \{0\}$ *por* (145) *é comutativo. Para* $P, Q \in \mathbb{P}^2$ *verifica-se* $P \otimes Q = \mathfrak{O}$ *se e só se* $P = Q$. *Adicionalmente, para todo* $P, Q, R \in \mathbb{P}^2$, *verifica-se*

$$R \in \mathrm{l}(P \otimes Q) \quad \textit{sse} \quad P \in \mathrm{l}(R \otimes Q) \quad \textit{sse} \quad Q \in \mathrm{l}(P \otimes R)$$

Assim cada ponto determina, através das suas coordenadas, uma recta. A recta que passa pelos pontos P e Q é determinada pelas coordenadas do ponto $P \otimes Q$.

□

Nestas circunstâncias, voltando ao problema inicial de construir funções racionais dados os seus pólos e zeros, tem-se

1. Se pretendermos um polinómio que tenha exactamente dois zeros, P e Q, basta construir a recta $\mathrm{l}(P \otimes Q)$.
2. Se pretender-mos uma função racional que tenha um zero Z e um polo P, podemos começar por escolher um outro qualquer ponto O que seja distinto de Z e de P e construir duas rectas, ambas passando por O, e passando por Z e por P.
$$p = \mathrm{l}(Z \otimes O) \quad , \quad q = \mathrm{l}(P \otimes O)$$
A fracção $f = p/q$ determina a função racional pretendida. Note-se que ela tem um polo em O que se anula com o zero que tem em O; por isso f acaba por ter só o polo P e o zero Z.
3. A estratégia anterior pode ser usada para construir funções racionais com pólos e zeros com ordem superior a 1. Vamos supor que se quer um polo P de ordem 2 e um zero Z também de ordem 2. Então escolhem-se dois quaisquer pontos O_1 e O_2 distintos entre si e distintos de P e Z. Em seguida constroem-se rectas
$$p_1 = \mathrm{l}(Z \otimes O_1) \qquad p_2 = \mathrm{l}(Z \otimes O_2) \qquad q_1 = \mathrm{l}(P \otimes O_1) \qquad q_2 = \mathrm{l}(P \otimes O_2)$$
A fracção $f = (p_1\, p_2) \,/\, (q_1\, q_2)$ determina a função racional pretendida. Note-se que tanto O_1 como O_2 aparecem simultaneamente como zeros e pólos e, por isso, anulam-se.

4. Vamos agora considerar que se quer n zeros $Z_1, \cdots, Z_n$, todos distintos, e n pólos $P_1, \cdots, P_n$ também todos distintos e distintos dos zeros. Basta escolher um ponto auxiliar O e construir as rectas

$$p_i = \mathrm{l}(Z_i \otimes O) \qquad q_i = \mathrm{l}(P_i \otimes O) \qquad i = 1 \cdots n$$

e definir a função racional $f = \prod_{i=1}^{n} (p_i/q_i)$.
Obviamente, se um zero ou um polo aparecer repetido (ordem > 1) temos de usar mais pontos auxiliares tal como fizemos no caso anterior.

Estes casos indicam um algoritmo simples para construir uma função racional dados os seus conjuntos de zeros e pólos. Este algoritmo é particularmente importante em curvas elípticas na construção de emparelhamentos.

8.2 Introdução a ideais e variedades

Problema simportantes, como a caracterização da intersecção de duas curvas, que são fundamentais ao estudo das curvas elípticas requerem uma análise, mesmo resumida, da noção de variedade e, por isso, da noção de ideal.

Seja R um anel; no que se segue vamos sempre assumir que os anéis são formam um domínio integral: isto é, são comutativos e $r, s \neq 0 \in R$ implica sempre $r\,s \neq 0$.

Dados subconjuntos $I, J \subseteq R$ define-se $I+J = \{\, r+s \mid r \in I\ ,\ s \in J \,\}$ e $I\,J = \{\, r\,s \mid r \in I\ ,\ s \in J \,\}$. Define-se, $I^0 = R$ e $I^{n+1} = I\,I^n$. O conjunto $\{s\}J$ escreve-se como sJ.

215 NOÇÃO

Um subconjunto não vazio $I \subseteq R$ é um **ideal** *quando $I\,R = I$ e $I + I = I$.*

Um ideal $\mathfrak{p} \neq R$ é **primo** *quando $r\,s \in \mathfrak{p}$ implica $r \in \mathfrak{p}$ ou $s \in \mathfrak{p}$. O ideal $\mathfrak{m}$ é* **máximo** *quando não está contido em nenhum outro ideal primo.*

Claramente, a soma e o produto de ideais são ideais. Se I é um ideal então o conjunto $\{\, r \mid r^n \in I$ para algum $n \,\}$ é também um ideal. Tal conjunto representa-do por $\sqrt{I}$ e designa-se por **radical** de I. Um **ideal radical** é qualquer ideal I que coincida com o seu radical. Todo o ideal primo $\mathfrak{p}$ é radical.

Cada ideal primo $\mathfrak{p} \subset R$ determina uma relação de equivalência em R da forma usual: $r \sim s$ sse $r - s \in \mathfrak{p}$. Porque $\mathfrak{p}$ é primo, o espaço quociente respectivo tem a estrutura de um domínio integral; representa-se $R/\mathfrak{p}$ tal anel. Seja $I \subset R$ um ideal que contenha $\mathfrak{p}$; define-se $I/\mathfrak{p}$ como o ideal $\mathfrak{q}$ tal que $I = \mathfrak{p} + \mathfrak{q}$. Verifica-se facilmente que $I/\mathfrak{p}$ determina um ideal em $R/\mathfrak{p}$; adicionalmente, todos os ideais em $R/\mathfrak{p}$ têm esta forma.

216 NOÇÃO

Um ideal da forma $s\,R$, com $s \in R$, diz-se **principal** *e representa-se por $\langle s \rangle$. Um ideal I é* **finitamente gerado** *quando se pode escrever como $I = s_1R + s_2R + \cdots + s_nR$, para um conjunto finito $\{s_1, s_2, \ldots, s_n\}$ de elementos de R designados por* **geradores**. *Neste caso escreve-se $I = \langle s_1, s_2, \ldots, s_n \rangle$.*

Se $\mathfrak{p}$ é um ideal primo, o seu **peso** *é o maior k tal que existe uma cadeia $\mathfrak{p}_0 \subset \mathfrak{p}_1 \subset \cdots \subset \mathfrak{p}_{k-1} \subset \mathfrak{p}$ em que os vários $\mathfrak{p}_i$ são ideais primos distintos entre si e distintos de $\mathfrak{p}$. O supremo dos pesos de todos os ideais primos de R designa-se por* **dimensão de Krull** *(ou, simplesmente,* **dimensão***) de R. Um anel de dimensão finita diz-se* **Noeteriano**.

217 TEOREMA (BÁSICO DE HILBERT)

Um anel R é Noeteriano se e só se for finitamente gerado. Adicionalmente, sendo R Noteriano, qualquer anel de polinómios $R[x_1, \cdots, x_n]$ é noeteriano.

□

Considere-se agora um corpo $\mathbb{K}$ e o anel $S_n = \overline{\mathbb{K}}[x_1, \ldots, x_n]$ dos polinómios nas variáveis $x_1, \ldots, x_n$ com coeficientes no fecho algébrico de $\mathbb{K}$. Interessa-nos considerar o anel S_n mas também os anéis quociente

$R = S_n/\mathfrak{p}$, sendo $\mathfrak{p}$ um ideal primo. Qualquer dos casos será designado por um **anel de polinómios**.

Uma observação importante resulta do facto de qualquer corpo $\mathbb{K}$, visto como um anel, ser noeteriano. De facto o único ideal primo de $\mathbb{K}$ é o anel trivial $\{0\}$ que tem peso 1. Então, pelo teorema básico de Hilbert, todo S_n é noeteriano; o que implica

218 COROLÁRIO *Todo o ideal $I \subset S_n$ é finitamente gerado.*

EXEMPLO 44: Tomemos $\mathbb{K}$ como $\mathbb{Q}$ (o corpo dos números racionais) e o anel de polinómios a duas variáveis S_2. Vamos ver alguns exemplos de ideais neste anel. Tenha-se em atenção que $S_2 = \overline{\mathbb{Q}}[x, y]$ e, dado que o fecho algébrico de $\mathbb{Q}$ é o corpo dos complexos $\mathbb{C}$, os elementos de S_2 são polinómios nas variáveis x, y com coeficientes complexos.

Como todo o ideal de S_2 é finitamente gerado (corolário 218) para definir um ideal basta indicar os seus geradores. É possível definir o ideal de outras formas: através de operações sobre outros ideais ou através de definição do conjunto por compreensão a partir de uma propriedade dos polinómios.

Via geradores temos, por exemplo, os ideais

$$I = \langle x, y - x \rangle \qquad J = \left\langle y^2 - x^3 - 1 \right\rangle$$

Pode-se definir ideias através das operações de soma e produto. Por exemplo

$$I + J = \left\langle x, y - x, y^2 - x^3 - 1 \right\rangle \quad , \quad I\,J = \left\langle x\,(y^2 - x^3 - 1), (y - x)\,(y^2 - x^3 - 1) \right\rangle$$

Pode-se finalmente definir ideiais por compreensão: seja $U = \{p_1, \ldots, p_n\}$ um conjunto finito de $\mathbb{K}^2$-pontos, então pode-se definir

$$I_U = \{ f \in S_2 \mid f(p) = 0 \quad \text{para todo } p \in U \}$$

Considere-se, de novo, $S_n = \overline{\mathbb{K}}[x_1, \ldots, x_n]$. Os ideais no anel S_n têm uma relação clara com determinados conjuntos $X \subset \mathbb{K}^n$ designados por **algébricos**. Para todo subconjunto $X \subseteq \mathbb{K}^n$ define-se

$$\mathbf{I}(X) = \{ f \mid f(p) = 0 \text{ para todo } p \in X \} \tag{146}$$

Pode-se verificar que $\mathbf{I}(X)$ é um ideal. Alternativamente seja $I \subset S_n$ um qualquer ideal; define-se

$$\mathbf{Z}(I) = \{ p \in \mathbb{K}^n \mid f(p) = 0 \text{ para todo } f \in I \} \tag{147}$$

Conjuntos $X \subseteq \mathbb{K}^n$ da forma $\mathbf{Z}(I)$, para algum ideal I dizem-se **algébricos**. $\mathbf{Z}$ mapeia ideais em conjuntos algébricos; a construção $\mathbf{I}(X)$ mapeis quaisquer conjuntos em ideais. A relação entre estas duas construções é expressa num dos resultados mais importantes da Álgebra.

219 TEOREMA (NULLSTELLENSATZ)
Todo o ideal $I \subset S_n$ verifica $\mathbf{I}(\mathbf{Z}(I)) = \sqrt{I}$.

Alguns corolários que são consequência imediata deste teorema e do facto do corpo $\overline{\mathbb{K}}$ ser algebricamente fechado.

220 COROLÁRIO *Sejam* $I, J \subset S_n$ *ideais e* $X, Y \subset S_n$ *conjuntos algébricos. Então verifica-se* $\mathbf{Z}(I+J) = \mathbf{Z}(I) \cap \mathbf{Z}(J)$, $\mathbf{Z}(I \cap J) = \mathbf{Z}(I) \cup \mathbf{Z}(J) = \mathbf{Z}(I\ J)$, $\mathbf{I}(X \cap Y) = \sqrt{\mathbf{I}(X) + \mathbf{I}(Y)}$ *e* $\mathbf{I}(X \cup Y) = \mathbf{I}(X) \cap \mathbf{I}(Y)$.

221 COROLÁRIO *Dado um qualquer conjunto* $X \subseteq \mathbb{K}^n$ *(algébrico ou não), seja* $\overline{X}$ *a intersecção de todos os conjuntos algébricos* Y *que contém* X*. Então* $\overline{X} = \mathbf{Z}(\mathbf{I}(X))$.

O conjunto $\overline{X}$ designa-se por **fecho** de X. Se X é algébrico e $X = \overline{X}$, então X diz-se **algébricamente fechado**.

Se existir uma partição $X = Y \cup Y'$, sendo Y e Y' subconjuntos próprios de X que são algébricamente fechados, então X diz-se **redutível**. Se não existir tal partição, X diz-se **irredutível**.

222 COROLÁRIO *Um conjunto algébrico* $X \subset \mathbb{K}^n$ *é irredutível se e só se* $\mathbf{I}(X)$ *é primo. Adicionalmente, se* X *for irredutível, o seu fecho* $\overline{X}$ *também é irredutível.*

223 COROLÁRIO *Um ideal principal* $\langle \phi \rangle$ *é primo se e só se* ϕ *for irredutível em* $\overline{\mathbb{K}}[x_1, \ldots, x_n]$.

224 COROLÁRIO *Para todo ponto* $p = (a_1, a_2, \cdots, a_n) \in \mathbb{K}^n$ *, o conjunto singular* $\{p\}$ *é algébrico.*

O ideal $\mathfrak{m}_p = \mathbf{I}(\{p\})$ *é máximo e é gerado pelos* n *polinómios* $\{(x_i - a_i)\}_{i=1}^n$ *. Isto é*

$$\mathfrak{m}_p = \langle\, x_1 - a_1\ ,\ x_2 - a_2\ ,\ \cdots\ ,\ x_n - a_n \,\rangle \tag{148}$$

Adicionalmente todo o ideal máximo em S_n *tem a forma* $\mathfrak{m}_p$ *, para algum ponto* $p \in K^n$ *. Finalmente, para todo o ideal* $I \subset R$*, tem-se* $p \in \mathbf{Z}(I)$ *se e só se* $I \subseteq \mathfrak{m}_p$.

Notas Provas para o Nullstellensatz e para estes corolários podem ser obtidas no textos standard de Geometria Algébrica; por exemplo, Commutative Algebra de David Eisenbud ou Algebraic Geometry de Robin Hartshorne, publicados no Graduate Texts in Mathematics da Springer-Verlag.

225 Definição
Uma **variedade afim** *(ou, simplesmente,* **variedade***) é um conjunto algébrico* $V \subseteq \mathbb{K}^n$ *irredutível e algebricamente fechado.*

A **dimensão** *de* V *é o maior* k *para o qual existe uma cadeia* $X_1 \subset X_2 \subset \cdots \subset X_k$, *de subconjuntos próprios* $X_i \subset V$ *que são distintos, irredutíveis e algebricamente fechados. Uma variedade designa-se por* **ponto**, **curva**, **superfície** *ou* **hipersuperfície**, *consoante a sua dimensão é, respectivamente,* 0, 1, 2 *ou* > 2.

O domínio integral $S_n/\mathbf{I}(V)$ *representa-se por* $\mathbb{A}(V)$ *e designa-se por* **anel afim** *de* V.

O conjunto $\mathbb{K}^n$, visto como uma variedade, representa-se por $\mathbb{A}^n$. O respectivo ideal $\mathbf{I}(\mathbb{A}^n)$ é o ideal trivial $\{0\}$. O conjunto vazio, visto como conjunto algébrico, é também uma variedade gerada pelo ideal $\langle 1 \rangle$.

Note-se que, sendo V uma variedade, o corolário 222 diz-nos que $\mathbf{I}(V)$ é um ideal primo e, por isso, o anel quociente $\mathbb{A}(V) = S_n/\mathbf{I}(V)$ é um domínio integral.

Claramente $\mathbb{A}(V)$ tem a estrutura de um espaço vectorial sobre $\overline{\mathbb{K}}$. Anéis que são extensões de um corpo K e são espaços vectoriais sobre esse corpo designam-se por K-**álgebras**. Assim $\mathbb{A}(V)$ é uma $\overline{\mathbb{K}}$-álgebra.

Como espaço vectorial, tem uma dimensão que coincide com o número de elementos numa sua base. No entanto $\mathbf{I}(V)$, como todo o ideal de S_n, é finitamente gerado; isto faz-nos pensar que o espaço vectorial $S_n/\mathbf{I}(V)$ tem uma dimensão finita. De facto, temos um resultado bastante mais forte.

226 TEOREMA
Seja V uma variedade. O anel afim $\mathbb{A}(V)$ é uma $\overline{\mathbb{K}}$-álgebra finitamente gerada. A dimensão de $\mathbb{A}(V)$ como espaço vectorial coincide com sua dimensão de Krull como anel e coincide também com a dimensão da variedade V.

Uma série de resultados importantes resultam deste teorema.

227 COROLÁRIO *A dimensão de $\mathbb{A}^n$ é n. Uma variedade $V \subset \mathbb{A}^n$ tem dimensão $n-1$ se e só se tem $I(V) = \langle \phi \rangle$ para algum polinómio ϕ irredutível em S_n.*

228 PROPOSIÇÃO
Seja V uma variedade de dimensão d e seja $\mathfrak{p}$ um ideal primo do anel afim $\mathbb{A}(V)$ de peso p. Então

$$\dim \mathbb{A}(V)/\mathfrak{p} \;=\; d - p$$

□

Os ideias máximos $\mathfrak{m}_p$ e as suas diversas potências $\mathfrak{m}_p^m$, definidos por pontos $p \in K^n$, são essenciais para a definição de multiplicidade (das raízes de um polinómio, da intersecção de duas curvas, etc.). Note-se que, para todo $n \geq 0$, o ideal $\mathfrak{m}_p^n$ está contido em $\mathfrak{m}_p$ e, genericamente, em todos os $\mathfrak{m}_p^k$, com $k < n$.

229 Noção

Seja $\mathfrak{p}$ *um ideal primo e* $p \in K^n$ *um ponto em* $\mathbf{Z}(\mathfrak{p})$ *(isto é, verifica-se* $\mathfrak{p} \subseteq \mathfrak{m}_p$*). Dado um qualquer ideal* $\mathfrak{a} \not\subseteq \mathfrak{p}$*, define-se a* **multiplicidade** *de* $\mathfrak{a}$ *em* p **sobre** $\mathfrak{p}$*, como a maior potência* $m \geq 0$ *tal que* $\mathfrak{a} \subseteq \mathfrak{m}_p^m/\mathfrak{p}$*. Representa-se essa multiplicidade por* $\eta_p(\mathfrak{a};\mathfrak{p})$*.*

Nomeadamente, quando $\mathfrak{p}$ *coincide com o ideal trivial* $\langle 0\rangle$*,* $\eta_p(\mathfrak{a};\langle 0\rangle)$ *representa-se simplesmente por* $\eta_p(\mathfrak{a})$ *e designa-se por* **multiplicidade** *de* $\mathfrak{a}$ *em* p*.*

A situação mais comum verifica-se quando se tem $\mathfrak{p} = \mathbf{I}(V)$, para uma determinada variedade V, e $\mathfrak{a} = \langle f\rangle$ para um polinómio f. Nestas circunstâncias, temos a noção de multiplicidade de um polinómio f num ponto p sobre a variedade V ou, simplesmente, multiplicidade do polinómio f no ponto p.

Exemplo 45: Considere-se o espaço afim $\mathbb{A}^3$ e a variedade afim E definida pelo polinómio $\phi \equiv x^2+y^2+z^2-z$. É facil verificar que E é uma esfera de raio $1/2$ centrada no ponto $(0,0,1/2)$. O plano definido pelo polinómio $p \equiv z$ é tangente à esfera na origem $(0,0,0)$. Trivialmente tem-se

$$z = (x^2+y^2+z^2) - \phi$$

Isto significa que, em $\mathbb{A}(E)$, temos $z \sim (x^2+y^2+z^2)$. O termo $(x^2+y^2+z^2)$ é um factor local na origem $(0,0,0)$ de grau 2; portanto o plano tangente z tem multiplicidade 2 na origem sobre a esfera E.

Numa variedade V, um ponto $p \in V$ diz-se **singular** quando $\eta_p(\mathbf{I}(V)) > 1$. As variedades não-singulares V são aquelas que não têm pontos singulares.

EXEMPLO 46: Tomemos de novo a esfera $E : \phi$, com $\phi \equiv x^2 + y^2 + z^2 - z$, que vimos no exemplo 45. Como determinar a multiplicidade de ϕ num ponto genérico $p = [a, b, c]$ da esfera?

Para isso a melhor solução é usar um sistema de computação algébrica, como o MAPLE, e um pacote de funções, como `PolynomialIdeal`, para computar as várias operações com ideais.

1. Começa-se por determinar o ideal máximo genérico $m_p = \langle x - a, y - b, z - c \rangle$ e as respectivas potências $m_p^2 = m_p\, m_p$, $m_p^3 = m_p^2\, m_p$, etc.
2. Tratando a, b, c como 3 novas variáveis, constrói-se o ideal $S := \langle \phi(a, b, c) \rangle$ que denota o facto de o ponto (a, b, c) pertencer à variedade E. Neste caso tem-se $S = \left\langle a^2 + b^2 + c^2 - c \right\rangle$.
3. No espaço de polinómios a 6 variáveis (x, y, z, a, b, c) constroem-se sucessivamente os ideais $s_1 = m_p + S$, $s_2 = m_p^2 + S$, $s_3 = m_p^3 + S$, etc.
4. Testa-se sucessivamente, $\phi \in s_1$, $\phi \in s_2$, $\phi \in s_3$, etc. O último k onde o teste $\phi \in s_k$ tem sucesso, é a multiplicidade pretendida.

Executando este algoritmo verifica-se que a multiplicidade é sempre 1 e, portanto, a variedade é não-singular.

A noção de ponto singular está assocoada à noção de tangente a uma variedade num ponto e, baseado neste conceito, é possível verificar facilmente se um ponto é ou não singular. Para isso precisamos de uma extensão das noções de jacobiano (noção 242, página 514) e de tangente (noção 207) a variedades.

230 NOÇÃO
Seja $V \subset \mathbb{A}^n$ *a variedade definida pelo ideal* $\langle g_1, \ldots, g_l \rangle$. *O* **jacobiano** *de* V *é a matrix de polinómios* $\mathcal{J}$ *cujo elemento genérico é* $\mathcal{J}_{ij} = \partial g_j / \partial x_i$.

O seguinte teorema[70] permite detectar os eventuais pontos singulares das variedades.

231 TEOREMA
Seja $\mathcal{J}$ *o jacobiano da variedade* V*; um ponto* $p \in V$ *é não-singular se e só*

$$\text{Rank}(\mathcal{J}(p)) \;=\; n - \dim(V)$$

EXEMPLO 47: Considere-se de novo a esfera nos exemplos 45 e 46. Nesta variedade a dimensão é 2 e o espaço tem dimensão 3. Donde, neste caso, $n - \dim(V) = 1$.

O ideal que define a esfera tem apenas um gerador (o polinómio ϕ). Portanto o jacobiano é o vector coluna das derivadas parciais $(\partial\phi/\partial x, \partial\phi/\partial y, \partial\phi/\partial z)$. Neste caso será $\mathcal{J} = (2\,x, 2\,y, 2\,z - 1)$ que, como matriz, tem *rank* 1 para todo ponto da variedade.

[70] Para prova veja-se HARTSHORNE, *Algebraic Geometry*.

Se considerar-mos a intersecção da esfera com o plano $x = 0$ temos um círculo; a dimensão da variedade é 1 donde, neste caso, $n - \dim(V) = 2$.

Os geradores do ideal que define o círculo são $\left\langle x^2 + y^2 + z^2 - z, x \right\rangle$. O jacobiano é a matriz $\begin{bmatrix} 2\,x & 1 \\ 2\,y & 0 \\ 2\,z - 1 & 0 \end{bmatrix}$.

No círculo tem-se $x = 0$. Para que esta matriz tenha *rank* < 2, a primeira coluna terá de ser múltipla da segunda; isto só acontece se for $y = 0$ e $z = 1/2$. No entanto o ponto $(0, 0, 1/2)$ não pertence à variedade. Por isso, neste círculo, o *rank* do jacobiano é sempre 2 e não existem pontos singulares.

Um terceiro exemplo é a variedade de dimensão 2 definida pelo polinómio $\psi = y^2\,z - x\,(x - z)^2$. O jacobiano é a matriz coluna $\mathcal{J} = \left[(x - z)\,(z - 3\,x)\ ,\ 2\,y\,z\ ,\ y^2 + 2\,x\,(x - z)\right]$.

Note-se que qualquer ponto onde seja $x = z$ e $y = 0$ é um ponto da variedade definida por ψ; nesses pontos o jaconiano reduz-se a $\mathcal{J} = [0, 0, 0]$; portanto tem *rank* 0. Como consequência todos os pontos da forma $(x, 0, x)$ são pontos singulares da variedade definida por ψ.

A noção de tangente a uma variedade é um conceito um pouco mais complexo do que o de tangente a uma curva. Por exemplo, em relação a uma superfície $V \subset \mathbb{A}^3$ (como a esfera dos exemplos anteriores) pode-se ver a tangente como um plano, uma recta ou mesmo só um ponto.

Por isso convém estender cuidadosamente este conceito.

Cada ponto $p = (a_1, \ldots, a_n) \in \mathbb{A}^n$ define a variedade $\mathbf{l}(p)$, designada por **hiper-plano** de p, gerada pelo polinómio do 1º grau $a_1\, x_1 + \cdots + a_n\, x_n$. A dimensão de $\mathbf{l}(p)$ é $n-1$, se for $p \neq (0, \ldots, 0)$ ou é n em caso contrário.

Um vector de pontos $L = (p_1, \ldots, p_n) \in (\mathbb{A}^n)^l$ gera a variedade $\mathbf{l}(L) = \bigcap_{j=1}^{l} \mathbf{l}(p_j)$. A dimensão desta variedade depende do número de pontos não nulos que são linearmente independentes. Vendo L como uma matriz, o número de pontos não-nuos linearmente independentes é dado pelo *rank* da matriz. Por isso,

232 FACTO
$\mathbf{l}(L)$ *é uma variedade e um* K*-espaço vactorial de dimensões* $n - \mathrm{Rank}(L)$.

233 NOÇÃO
Dada uma variedade afim $V \subset \mathbb{A}^n$ *de jacobiano* $\mathcal{J}$, *seja* $\mathcal{T}(V)$ *a variedade dada por*

$$\mathcal{T}(V) \;=\; \{\, (p, X) \in V \times \mathbb{A}^n \;\mid\; X \in \mathbf{l}(\mathcal{J}(p)) \,\} \tag{149}$$

Caso V *seja não-singular então* $\mathcal{T}(V)$ *designa-se por* **feixe tangente** *de* V.

Note-se nesta definição que, sendo V é não-singular, temos $\dim(V) = n - \mathrm{Rank}(\mathcal{J}(p)) = \dim(\mathbf{l}(\mathcal{J}(p)))$, independentemente de $p \in V$. Portanto a variedade $\mathbf{l}(\mathcal{J}(p))$ tem, para todo $p \in V$, exactamente a mesma dimensão que V.

8.3 Divisores

Dado que o conjunto de zeros e de pólos caracterizam completamente as funções racionais sobre uma curva C, é conveniente introduzir uma noção que faça esta abstracção; isto é, represente os conjuntos de pólos e zeros com as suas ordens associadas. Esta é a noção de **divisor**.

234 DEFINIÇÃO
Um **divisor** *numa curva projectiva* C *é uma soma formal escrita*

$$D = \sum_{P \in C} n_P (P) \qquad com \quad n_p \in \mathbb{Z} \tag{150}$$

onde o número de inteiros n_P *diferentes de zero é finito.*

A **ordem** *do divisor* D *no ponto* P, *representada* $\mathsf{ord}_P(D)$, *é o inteiro* n_P. *O* **suporte** *de* D, $\mathsf{supp}(D)$, *é o conjunto dos pontos* $P \in C$ *em que* $\mathsf{ord}_P(D) \neq 0$. *A* **grau** *do divisor* (150) *é o inteiro*

$$\mathsf{deg}(D) = \sum_{p \in C} \mathsf{ord}_P(D)$$

O conjunto dos divisores de C *é representado por* Div_C *e o conjunto dos divisores de grau* 0 *(i.e., os que verificam* $(\sum_P n_P) = 0$ *) representa-se por* Div_C^0 .

Um divisor é **efectivo** *(e escreve-se* $D \geq 0$ *) se* $\mathsf{ord}_P(D) \geq 0$ *para todo* P. $D \geq D'$ *é uma abreviatura para* $D - D' \geq 0$.

Se $f \in \mathbb{K}(C)$, *o divisor de* f, *escrito como* (f) *ou* $\mathsf{div}(f)$ *é a soma formal*

$$(f) \doteq \sum_{P \in C} \mathsf{ord}_P(f)\,(P) \tag{151}$$

Divisores D *para os quais existe* $f \in \mathbb{K}(C)$ *tal que* $D = (f)$ *chamam-se* ***divisores principais***.

Como cada $f \in \mathbb{K}(C)$ tem um número finito de pólos e zeros, a soma formal (151) é um divisor.

Todo o divisor D pode ser escrito, de forma única, como $D_0 - D_\infty$ em que D_0 e D_∞ são divisores efectivos de suportes disjuntos. O par de divisores (D_0, D_∞) designa-se por **fraccionamento efectivo** de D.

235 FACTO
O conjunto dos divisores Div_C *determina um grupo abeliano que contém os divisores de ordem zero,* Div_C^0, *como sub-grupo.*

Esboço de prova Pode-se ver o conjunto dos divisores de C, Div_C embebido no espaço vectorial $\mathbb{Z}^C$: cada divisor é um vector de componentes em $\mathbb{Z}$ e com tantas componentes quantos os pontos $P \in C$. Os divisores D são, assim, os elementos de Div_C que têm um

número finito de componentes não nulas. A soma de vectores gera a operação de grupo $+$ nos divisores divisores

$$\sum_{P\in C} n_P P \;+\; \sum_{P\in C} m_P P \;\doteq\; \sum_{P\in C} (n_P + m_p)\, P \tag{152}$$

O elemento neutro é o divisor nulo $\mathcal{N} \doteq \sum_{P\in C} 0\, P$. Com tal soma e elemento neutro, Div_C tem a estrutura de um grupo.

O conjunto dos divisores de grau 0, Div_C^0 , formam um sub-grupo porque, como facilmente se verifica, a soma de dois divisores de grau zero gera de novo um divisor de grau zero.

Os divisores principais (divisores da forma $\mathrm{div}(f)$, com f uma função racional em $\mathbb{K}(C)$) têm um papel especial na teoria dos divisores.

Note-se que f é determinado por fracções homogéneas (onde o grau do numerador é igual ao grau do denominador); por isso é natural pensar-se que o número de zeros do numerador seja igual ao número de zeros do denominador.

A noção de divisor prende-se, obviamente, com a distribuição dos pólos e dos zeros das funções racionais; por isso faz sentido a seguinte definição:

236 DEFINIÇÃO
Para cada divisor $D \in \mathrm{Div}(C)$ *seja*

$$L(D) \;=\; \{0\} \cup \{\, f \in \mathbb{K}(C)^* \;\mid\; D + (f) \geq 0 \,\} \tag{153}$$

$L(D)$ contém, em primeiro lugar e como caso particular, a função constante 0; isto é essencial, como veremos adiante, para a estrutura vectorial que queremos impor a este espaço.

Essencialmente porém, $L(D)$ contém todas as funções racionais não-nulas que têm zeros que "anulam" os pólos de D e que não introduzem pólos adicionais. Note-se que, porque f é racional, o número de pólos de f deve ser igual ao número de zeros de f.

Para percebermos o papel fundamental que estes espaços $L(D)$ têm na construção de curvas, o seguinte exemplo ilustra alguns divisores e a construção do espaço respectivo.

EXEMPLO 48:

1. Vamos supor que se tem $D = (P) + (Q) - (R)$, em que $P, Q, R \in C$ são pontos distintos da curva C.

   Tome-se uma qualquer função racional f candidata pertencer a $L(D)$ com apenas um zero e um pólo; isto é, (f) tem a forma $(A) - (B)$ (com A e B por definir). Temos $D + (f) = (P) + (Q) + (A) - (R) - (B)$ e pretende-se que seja $D + (f) \geq 0$.
   Como nesta soma existem dois pólos que têm de ser anulados e os graus de liberdade são A e B, pode-se escolher $A = R$ e $B = Q$. Isto basta para que $D + (f) = (P) \geq 0$.
   Poderíamos também ter escolhido $B = P$ e obtínhamos $D + (f) = (Q) \geq 0$. Note-se que o valor de A não pode ser alterado.

Seria possível usar um $f \in L(D)$ que tivesse dois pólos e dois zeros?

$$(f) \;=\; (A) + (A') - (B) - (B')$$

Neste caso os 4 pontos A, A', B, B' têm de ser distintos e será

$$D + (f) \;=\; (P) + (Q) + (A) + (A') - (R) - (B) - (B')$$

Pode-se usar P e Q para anular B e B' e usar A ou A' para anular R.

Este é o caso limite: não existe nenhum $f \in L(D)$ com três zeros e três pólos porque D só tem 2 zeros para anular os pólos de f.

2. $D = n(P) - m(Q)$, com $n, m > 0$ e $P \neq Q$.

Considere-se uma função racional f tal que $(f) = k(A) - k(B)$. Note-se que o número de zeros tem de ser igual ao número de pólos.

Neste caso, $D + (f) = n(P) + k(A) - m(Q) - k(B)$; para este divisor ser efectivo, terá de ser

$$A = Q \quad \text{e} \quad k \geq m \quad \text{e} \quad B = P \quad \text{e} \quad k \leq n$$

Se for $n = m$ (isto é, se D tiver grau zero), então existe uma única possibilidade: f tem de ter o divisor $n(Q) - n(P) = -D$.

237 LEMA *Seja D um divisor de uma curva projectiva C e $L(D) \doteq \{ f \in \mathbb{K}(C)^* \mid D + (f) \geq 0 \} \cup \{0\}$. Então $L(D)$ é um espaço vectorial sobre $\bar{\mathbb{K}}$ cuja dimensão, denotada por $\ell(D)$, é finita.*

Prova Afirmar que $L(D)$ é um espaço vectorial é equivalente a dizer que, para todos $f, g \in L(D)$ e $a \in \bar{K}^*$, se verifica $f + g \in L(D)$ e $a\, f \in L(D)$.

Como af ou é zero (se $a = 0$) ou, se $a \neq 0$, tem os mesmos zeros e pólos que f, então $f \in L(D)$ implica $af \in L(D)$. Do mesmo modo, para todo ponto $P \in C$, a ordem $\mathrm{ord}_P(f + g)$ é sempre maior ou igual que $\mathrm{ord}_P(f)$ e $\mathrm{ord}_P(g)$. Por isso, $f, g \in L(D)$ implica $(f + g) \in L(D)$.

Provar que a dimensão do espaço vectorial é finita é mais complexo. No entanto basta recordar que, se fosse infinita, seria possível construir uma combinação linear infinita de funções racionais linearmente independentes.

238 FACTO
Se $D' = D + (h)$, para algum $h \in \mathbb{K}(C)^*$, então os espaços vectoriais $L(D)$ e $L(D')$ são isomórficos.

Prova Considere-se o morfismo $f \mapsto h\, f$ entre os espaços vectoriais $L(D')$ e $L(D)$. Como $D' = D + (h)$ então $f \in L(D')$, se não for 0, se e só se $(f) + (h) + D = (fh) + (D) \geq 0$. Portanto $f \in L(D')$ se e só se $fh \in L(D)$.

Este resultado justifica a seguinte definição

239 Definição
Dois divisores D, D' *numa curva projectiva* C *são equivalentes, e escreve-se* $D \sim D'$, *se existir* $h \in \mathbb{K}(C)$ *tal que* $D - D' = (h)$.

Esta relação (que facilmente se verifica ser uma relação de equivalência) induz um espaço quociente $\mathrm{Div}^0_C/\sim$ no conjunto de divisores de grau zero que vai ter um papel fundamental na construção das curvas elípticas. Como claramente $\mathrm{Div}^0_C/\sim$ herda de Div^0_C a estrutura do grupo abeliano, designa-se este espaço por **grupo de Picard** , normalmente representado por Pic_C .

O teorema que permite relacionar a estrutura de grupo das curvas abelianas com divisores.

240 Teorema (Riemann-Roch)
Dada uma curva projectiva absolutamente irredutível C *existe uma constante* $g \geq 0$ *tal que, para todo o divisor* $D \in \mathrm{Div}(C)$,

$$\ell(D) \geq \deg(D) + 1 - g$$

Adicionalmente, se $2g \leq \deg(D) + 1$, *verifica-se*

$$\ell(D) = \deg(D) + 1 - g \tag{154}$$

Prova A prova deste teorema requer noções que saem fora do âmbito deste trabalho. A estrutura essencial da prova (na sua forma mais recente) pode ser vista no artigo

* http://planetmath.org/encyclopedia/ProofOfRiemannRochTheorem.html.
Uma breve história do teorema e da sua importância pode ser vista em
* http://en.wikipedia.org/wiki/Riemann-Roch_Theorem.

A constante g é um invariante da curva C e designa-se por **genus** da curva. A informação essencial que resulta deste teorema é que g é independente do divisor D e a igualdade (154) verifica-se para todo o divisor cujo grau seja maior ou igual que $2\,g-1$.

Algumas consequência imediatas do teorema de Riemann-Roch

241 PROPOSIÇÃO
Nas condições do teorema 243.

(1) Se C tem genus $g=0$ então existem pontos distintos $P\neq Q\in C$ tais que $(P)\sim(Q)$.

(2) Se C tem genus $g=1$ verifica-se $(P)\sim(Q)$ se e só se $P=Q$.

Prova Sejam P,Q dois pontos tais que $(P)\sim(Q)$ e seja $h\in\bar{\mathbb{K}}(C)$ tal que $(P)=(Q)+(h)$. Daqui conclui-se que $h\in L((Q))$ e, caso seja $P\neq Q$, o morfismo $f\mapsto hf$ estabelece um isomorfismo entre $L((P))$ e $L((Q))$ (ver lema 240). Como $(P)\geq 0$ e $(Q)\geq 0$, ambos os espaços $L((P))$ e $L((Q))$ contém a função constante 1 (já que 1 não tem zeros nem pólos).

Ambos os divisores (P) e (Q) têm grau 1. Se a curva tem genus 0, então $\deg((P))+1\geq 2g$ e a dimensão $\ell((P))=\ell((Q))$ é, pelo teorema de Riemann-Roch, igual a 2. Neste caso é possível existir $h\neq 1$ que mapeia a função $1\in L((P))$ na função $h\in L((Q))$; portanto pode ser $P\neq Q$.

Caso a curva tenha genus 1, já a dimensão $\ell((P)) = \ell((Q)) = 1$ e, por isso, tanto $L((P))$ como $L((Q))$ não podem conter outras funções que não sejam constantes; por isso h tem de ser constante e daí só pode ser $P = Q$.

242 PROPOSIÇÃO

Nas condições do teorema 243, se C tem genus 1 e um ponto $\mathcal{O}$ então existe um morfismo $\sigma : \mathrm{Div}_C^0 \to C$ que para cada divisor $D \in \mathrm{Div}_C^0$, existe um único ponto da curva $\sigma(D)$ tal que $D \sim (\sigma(D)) - (\mathcal{O})$. Adicionalmente

(i) σ induz um isomorfismo entre o grupo de Picard Pic_C e a curva C.

(ii) $D \in \mathrm{Div}_C^0$ é principal se e só se $\sigma(D) = \mathcal{O}$.

Prova

(1) Seja D um qualquer divisor de grau 0; então $D + (\mathcal{O})$ tem grau 1 e, numa curva de genus 1, o teorema de Riemann-Roch diz-nos que $\ell(D + (\mathcal{O})) = 1$.
Seja f um gerador de $L(D + (\mathcal{O}))$. Então será simultaneamente, $\deg((f)) = 0$, porque f é racional, $\deg(D) = 0$, por hipótese, e $(f) + D + (\mathcal{O}) \geq 0$ porque $f \in L(D + (\mathcal{O}))$.
O único modo de compatibilizar estas três relações é existir um P tal que

$$(f) + D + (\mathcal{O}) \;=\; (P)$$

ou seja $D + (f) = (P) - (\mathcal{O})$ e, portanto

$$(P) - (\mathcal{O}) \;\sim\; D \tag{155}$$

Este P é único. De facto se tivermos $D \sim D'$ e $D' \sim (P') - (\mathcal{O})$, teríamos necessariamente

$$(P) - (\mathcal{O}) \sim (P') - (\mathcal{O}) \;\Rightarrow\; (P) \sim (P')$$

e, como acurva tem genus 1, tal implica (como acabámos de ver) $P = P'$.

Portanto fica bem definido uma função $\sigma : \mathrm{Div}_0(C) \to C$ que mapeia D no ponto P que verifica (155).

(2) Como consequência adicional vemos que esta construção associa dois divisores equivalentes exactamente ao mesmo ponto. Ou seja, $D \sim D'$ implica $\sigma(D) = \sigma(D')$. Por isso fica definido uma função que mapeia classes de equivalência de divisores em pontos da curva

$$\tilde{\sigma}\colon \mathrm{Pic}(C) \to C \qquad \tilde{\sigma}([D]) = \sigma(D)$$

Esta função tem uma inversa óbvia: a aplicação $\tilde{\sigma}^{-1}\colon P \mapsto [(P) - (\mathcal{O})]$ que associa um ponto $P \in C$ à classe de equivalência do divisor $(P) - (\mathcal{O})$. Assim, $\tilde{\sigma}$ é bijectiva.

(3) Se $\sigma(D) = O$ então $D \sim (\mathcal{O}) - (\mathcal{O})$. Isto significa que, para algum $f \in K(C)$,

$$D = (f) + (\mathcal{O}) - (\mathcal{O}) = (f)$$

Inversamente, se $D = (f)$, então $D + (f) = (\mathcal{O}) - (\mathcal{O})$ o que implica $D \sim (\mathcal{O}) - (\mathcal{O})$ e, por isso, $\sigma(D) = \mathcal{O}$.

243 DEFINIÇÃO

Seja C uma curva projectiva de genus 1 onde está identificado um ponto $\mathcal{O}$. Seja $\sigma\colon \mathrm{Div}^0_C \to C$ definido na proposição 245. Sejam $P, Q \in C$ pontos arbitrários da curva e $n \in \mathbb{Z}$ um inteiro arbitrário. Define-se

$$\begin{array}{llllll} P \oplus Q & = \sigma((P) + (Q) - 2(\mathcal{O})) & & -P & = \sigma((\mathcal{O}) - (P)) \\ [n]P & = \sigma(n(P) - n(\mathcal{O})) & [0]P = \mathcal{O} & [-n]P & = \sigma(n(\mathcal{O}) - n(P)) \end{array}$$

244 PROPOSIÇÃO

Nas condições da definição 246,

(i) $\langle C, \oplus, O\rangle$ *tem a estrutura de um grupo abeliano e o isomorfismo* $\mathrm{Pic}_C \tilde{\to} C$ *preserva a estrutura de grupos.*

(ii) Seja $D = \sum_{P\in C} n_P(P)$, *um divisor de grau* 0 *arbitrário; então* $\sigma(D) = \bigoplus_{P\in C} [n_P]P$.

Prova

(i) Vimos que $\tilde{\sigma}$ é uma bijecção. Esta função transforma-se num homomorfismo de grupos abelianos se, em C, se optar pela a estrutura mapeada por $\tilde{\sigma}$ a partir das operações de grupo de $\mathrm{Pic}(C)$. Isto é o que ocorre quando se define

$$P \oplus Q = \sigma((P) + (Q) - 2(\mathcal{O})) = \sigma((P) - (\mathcal{O}) + (Q) - (\mathcal{O})) = \sigma(\sigma^{-1}(P) + \sigma^{-1}(Q))$$

(ii) Note-se que a definição de $[n]P$ é equivalente a $([n]P) - (\mathcal{O}) \sim n(P) - n(\mathcal{O})$. Seja $Q \doteq \bigoplus_{P\in C} [n_P]P$. Pela definição da operação $\oplus$ temos

$$\begin{aligned}(Q) - (\mathcal{O}) \sim & \sum \left(([n_P]P) - (\mathcal{O})\right) \\ \sim & \sum n_P(P) - \sum n_P(\mathcal{O}) = D - \left(\sum n_p\right)(\mathcal{O}) = \\ = & D - 0(\mathcal{O}) \;=\; D\end{aligned}$$

Portanto temos $D \sim (Q) - (\mathcal{O})$ o que significa que $Q = \sigma(D)$.

Como consequência imediata de (ii) temos

245 COROLÁRIO *Seja C uma curva elíptica e $D = \sum_{P \in C} n_P (P)$ um qualquer divisor de grau 0. Então*

$$D \sim 0 \quad \text{sse} \quad \bigoplus_{P \in C} [n_P]P = \mathcal{O}$$

Vimos que uma função racional $f \in K(C)$ determina uma aplicação dos pontos da curva em $\bar{K}$: para cada ponto P é bem definido o valor $f(P)$.

Faz sentido tentar estender este conceito para divisores. Note-se que divisores representam, essencialmente, arranjos de zeros e de pólos de funções racionais; por isso faz sentido pensar que uma soma formal de zeros e pólos associamos o produto dos valores da função nestes pontos.

246 NOÇÃO
Seja C uma curva elíptica sobre K e $h \in K(C)$ uma qualquer função racional sobre essa curva. Para todo o divisor de grau zero $D = \sum_{P \in C} n_P (P)$ define-se

$$h(D) = \prod_{P \in C} h(P)^{n_P} \tag{156}$$

A valoração de uma função racional homogénea h num divisor de grau 0 mantém-se invariante se a função h, nos pontos da curva, sofrer uma "mudança de escala". Isto é,

247 Facto
Sejam $h, h' \in K(C)$ funções tais que, para alguma constante $\lambda \in K^*$, verifica-se $h'(P) = \lambda\, h(P)$, para todo o ponto $P \in C$. Então $h'(D) = h(D)$ para todo o divisor D de grau 0.

Prova Seja $D_0 = \sum_i n_i(P_i)$ e $D_\infty = \sum_j m_j\,(Q_j)$ (com $n_i, m_j > 0$) o fraccionamento efectivo de D. Seja k o grau destes divisores; isto é $k = \sum_i n_i = \sum_j m_j$.

$$h'(D) = \frac{\prod_i \lambda^{n_i}\, h(P_i)^{n_i}}{\prod_j \lambda^{m_j}\, h(Q_j)^{m_j}} = \frac{\lambda^{\sum_i n_i}\, \prod_i h(P_i)^{n_i}}{\lambda^{\sum_j m_j}\, \prod_j h(Q_j)^{m_j}} = \frac{\lambda^k\, \prod_i h(P_i)^{n_i}}{\lambda^k\, \prod_j h(Q_j)^{m_j}} = h(D)$$

O seguinte teorema é uma caracterização fundamental da valoração de funções racionais em divisores principais.

248 Teorema (Reciprocidade de Weil)
Seja C uma curva elíptica sobre K. Sejam $f, g \in K(C)$ funções racionais sobre C. Se o suporte de (f) e de (g) forem disjuntos, então

$$f((g)) = g((f))$$

□

Numa curva projectiva C de genus 1 onde está identificado um ponto específico O, são importantes os diferentes

divisores $n(\mathcal{O})$ com $n = 1, 2, \cdots$. Seja

$$L_n \;=\; L(n(\mathcal{O})) \;=\; \{\, f \in K(C) \;\mid\; (f) + n(\mathcal{O}) \geq 0 \,\} \cup \{0\}$$

Pode-se ver L_n como o espaço das funções racionais de C que não têm qualquer pólo a não ser em $\mathcal{O}$ e aí a ordem não é superior a n. Note-se que estes espaços vectoriais formam, por inclusão, uma cadeia ascendente

$$L_1 \subset L_2 \subset \cdots \subset L_n \subset \cdots$$

Como a curva tem genus 1, a dimensão de L_n é n (pelo teorema de Rieman-Roch). Portanto o espaço L_n tem n geradores linearmente independentes. Quais são estes geradores?

O espaço L_1 contém todas as funções racionais constantes e, como a sua dimensão é 1, estas definem todos os seus elementos. Deste modo não pode haver nenhuma função racional $f \in K(C)$ que tenha apenas um pólo simples em O: ou não tem qualquer pólo (e é uma constante) ou contém pelo menos outro pólo.

Portanto L_1 tem, por gerador a função unidade 1 : isto é,

$$L_1 \;=\; \{\, a\,1 \;\mid\; a \in K \,\}$$

L_2 tem dimensão 2 e contém L_1; portanto vai existir um gerador não constante g tal que

$$L_2 \;=\; \{\, a\,1 + b\,g \;\mid\; a, b \in K \,\}$$

Note-se, pelo que foi dito antes, que g tem um pólo único de ordem 2 em $\mathcal{O}$.

Considere-se, agora, L_3. Como contém L_2 e tem dimensão 3 será da forma

$$L_3 \;=\; \{\, a\,1 + b\,g + c\,h \;\mid\; a, b, c \in K \,\}$$

em que o novo gerador h é uma função racional com um pólo único de ordem 3 em $\mathcal{O}$.

Os geradores de L_3, $\{1, g, h\}$, são linearmente independentes uma vez que têm ordens dos pólos em $\mathcal{O}$ diferentes. Como consequência, este triplo define uma aplicação racional

$$[g, h, 1] \colon C \setminus \{O\} \to \mathbb{P}^2 \tag{157}$$

entre curva C e o espaço projectivo $\mathbb{P}^2$. Como g e h não têm pólos em C, para além de $\mathcal{O}$, esta aplicação é definida para todo $P \in C \setminus \{O\}$.

Escolham-se polinómios homogéneos do mesmo grau $u, v, w \in K[C]$ tais que w tem um zero triplo em $\mathcal{O}$ e mais nenhum zero em pontos de C, v/w é um representante de h e u/w é um representante de g. É possível escolher tais polinómios porque, em C, h e g só têm pólos em $\mathcal{O}$, sendo triplo o pólo de h e duplo o pólo de g. Assim, em $\mathcal{O}$, u tem de ter um zero simples (já que o pólo de g em $\mathcal{O}$ tem ordem 2) e v não tem nenhum zero.

Define-se agora $\varphi\colon C \longrightarrow \mathbb{P}^2$ por este triplo de polinómios

$$\varphi = [u, v, w] \quad \text{com} \quad g = [u/w] \ , \ h = [v/w] \tag{158}$$

249 LEMA *φ definido em* (158) *é um morfismo injectivo* $\varphi\colon C \longrightarrow \mathbb{P}^2$ *tal que* $\varphi(\mathcal{O}) = P_\infty$ *e, para* $P \neq \mathcal{O}$, $\varphi(P) = [g(P), h(P), 1]$.

Prova Se $P = \mathcal{O}$ tem-se $\varphi(P) = [u(\mathcal{O}), v(\mathcal{O}), w(\mathcal{O})] = [0, v(\mathcal{O}), 0] = [0, 1, 0] = P_\infty$. Se $P \neq \mathcal{O}$ tem-se $w(P) \neq \mathcal{O}$ e, por isso, $\varphi(P) = [u(P), v(P), w(P)] = [u(P)/w(P), v(P)/w(P), 1] = [g(P), h(P), 1]$. Resta provar que ϕ é injectivo. Vamos supor que existiam dois pontos $P \neq Q$ tais que $\varphi(P) = \varphi(Q)$. Então necessariamente, para qualquer $f \in L_3$, seria $f(P) = f(Q)$ já que f é uma combinação linear de g e h. Tome-se agora um qualquer ponto $A \in C$ distinto de O, P ou Q e considere-se duas funções distintas $u, v \in L_3$ que partilhem o mesmo zero A. Então, como $u(P) = u(Q)$ e $v(P) = v(Q)$, a função racional $(u - v)$ pertence a L_3 e tem zeros em A, P e Q; logo $(A) + (P) + (Q) \sim 3(\mathcal{O})$; como A é arbitrário , isto não é possível. Consequentemente, φ é um morfismo injectivo.

O seguinte lema estabelece uma relação particular entre divisores de C efectivos de ordem 3 e rectas em $\mathbb{P}^2$.

250 LEMA *Seja* $D \geq 0$ *um divisor efectivo sobre* C *de grau* 3 *que verifica* $D \sim 3(\mathcal{O})$. *Então* φ, *definido em* (158), *mapeia todos os pontos de* C *de ordem não nula em* D *sobre uma mesma recta de* $\mathbb{P}^2$.

Prova Seja f tal que $D = (f) + 3(\mathcal{O})$; porque $D \geq 0$, a função f é um elemento de $L_3 = L(3(\mathcal{O}))$. Então f pode-se escrever como uma combinação linear dos geradores $\{1, g, h\}$; isto é, $f = ag + bh + c$ para alguns $a, b, c \in K$.

Seja $\mathbf{l}$ a recta em $\mathbb{P}^2$ determinada pelo polinómio $aX + bY + cZ$. Como, para todo $P \in C$, se tem $\mathbf{l}(\varphi(P)) = f(P)$, concluímos que P é um zero de f se e só se $\varphi(P)$ for um ponto da recta $\mathbf{l}$. Mas os zeros de f são precisamente os pontos de C que têm ordem não nula em D. Portanto todos os pontos de ordem não nula de D são pontos da recta $\mathbf{l}$.

A partir dos geradores $\{1, g, h\}$ de L_3 constrói-se os geradores dos espaços seguintes. Por exemplo, g^2 tem um pólo único de ordem 4 em $K(C)$; mais nenhum polinómio construído com estas 3 funções tais pólos. Logo os geradores de L_4 são $\left\{1, g, h, g^2\right\}$.

Da mesma forma se conclui que $g\,h$ tem um pólo único de ordem 5 em $\mathcal{O}$ e mais nenhuma combinação polinomial das três funções tem tais pólos. Logo os geradores de L_5 são $\left\{1, g, h, g^2, g\,h\right\}$.

Quando se chega a L_6, porém, algo de novo ocorre. Pode-se construir o pólo de ordem 6 de dois modos diferentes: ou com g^3 ou, então, com h^2. Desta forma existem 7 candidatos a geradores, $\left\{1, g, h, gh, g^2, g^3, h^2\right\}$. Como o espaço só tem dimensão 6, as 7 funções não podem ser linearmente independentes. Como as 5 primeiras têm de pertencer à base de geradores (porque são geradores de L_5), então h^2 (ou g^3) devem ser representáveis como combinação linear dos restantes 6 elementos.

Por isso tem de existir coeficientes $c_i \in K$ tais que

$$h^2 = g^3 + c_4 g\,h + c_3 g^2 + c_2 h + c_1 g + c_0 \tag{159}$$

O coeficiente de g^3 tem de ser $\neq 0$ porque não seria possível, de outra forma, que esta igualdade se verificasse e que h^2 tivesse o pólo de ordem 6 no ponto $\mathcal{O}$; sendo assim, pode-se assumir que o coeficiente é 1.

Tradicionalmente esta equação escreve-se (usando uma outra sequência de coeficientes) de forma algo diferente

$$h^2 + a_1 g\, h + a_3 h = g^3 + a_2 g^2 + a_4 g + a_6 \tag{160}$$

que se designa por **forma de Weierstraß**. Isto permite-nos enunciar um segundo lema

251 LEMA *Seja E a curva em $\mathbb{P}^2$ determinada pelo polinómio*

$$\phi = Y^2\, Z + a_1\, X\, Y\, Z + a_3\, Y\, Z^2 - X^3 - a_2\, X^2\, Z - a_4\, X\, Z^2 - a_6\, Z^3 \tag{161}$$

E é uma curva não-singular e φ definido no lema 252 é um isomorfismo entre C e E

Prova Substituindo h por v/w e g por u/w a igualdade (160) pode-se escrever

$$v^2\, w + a_1\, u\, v\, w + a_3\, v\, w^2 - u^3 - a_2\, u^2 w - a_4\, u\, w^2 - a_6\, w^3 \;=\; 0$$

Isto é equivalente à afirmação de que $\phi(\varphi(P)) = 0$ para todo o ponto P da curva; portanto φ, definido no lema 252, é um morfismo de C para E. O referido lema diz-nos que φ é um isomorfismo entre C e a sua imagem; como é surjectivo, concluímos que a imagem tem de coincidir com E e que φ é um isomorfismo entre C e E.

A consequência essencial destes lemas é

252 TEOREMA
Qualquer curva elíptica C/K sobre o espaço projectivo $\mathbb{P}^2$ é isomórfica com uma curva plana não singular determinada por uma polinómio $\phi \in \mathbb{K}[X, Y, Z]$ da forma (161).

No que se segure C/K é uma curva de genus 1 em $\mathbb{P}^2$ com ponto identificado $\mathcal{O}$.

253 COROLÁRIO *As funções racionais $g = x/z$ e $h = y/z$ são geradores g e h de L_3.*

Como consequência do lema 253 tem-se ainda,

254 LEMA *Seja $\langle C, \oplus, \mathcal{O}\rangle$ a estrutura de grubo abeliano definido em C (definição 246). Então os pontos P, Q e R de C são mapeados por φ em pontos sobre uma mesma recta de $\mathbb{P}^2$ se e só se $-R = P \oplus Q$.*

Prova Seja D o divisor $(P) + (Q) + (R)$ que é efectivo e tem grau 3. Pela definição das operações de grupo tem-se $(P) + (Q) - 2(\mathcal{O}) \sim (\mathcal{O}) - (R)$ o que é equivalente a ser $D = (P) + (Q) + (R) \sim 3(\mathcal{O})$. Pelo lema 253 tem-se $D \sim 3(\mathcal{O})$ se e só se os pontos P, Q e R são mapeados em pontos sobre uma mesma recta de $\mathbb{P}^2$.

O seguinte lema é verdadeiramente "multiusos"

255 LEMA *Sejam P, Q, R, S pontos de C tais que tanto P como Q são distintos de R ou S e tais que $P \oplus Q = R \oplus S$.*

(i) *Se* $P \neq Q$ *seja* $p = \mathbf{l}(P \otimes Q)$ *a recta que passa pelos pontos* P *e* Q. *Se for* $P = Q$ *seja* $p = \mathbf{l}(\mathcal{J}^C(P))$ *a recta tangente à curva em* P.

(ii) *Se* $R \neq S$ *seja* $q = \mathbf{l}(R \otimes S)$ *a recta que passa pelos pontos* R *e* S. *Se for* $R = S$ *seja* $q = \mathbf{l}(\mathcal{J}^C(R))$ *a recta tangente à curva em* R.

Então $(P) + (Q) = (R) + (S) + (p/q)$.

Prova

1. No caso em que todos os pontos são distintos. Pelo lema 253 a recta $p = \mathbf{l}(P \otimes Q)$ verifica $(p) = (P)+(Q)+(-(P \oplus Q))-3(\mathcal{O})$. Pela mesma razão a recta $q = \mathbf{l}(R \otimes S)$ verifica $(q) = (R) + (S) + (-(R \oplus S)) - 3(\mathcal{O})$. Dado que, por hipótese, se tem $P \oplus Q = R \oplus S$ conclui-se que $(p/q) = (p) - (q) = (P) + (Q) - (R) - (S)$.
2. Se $P = Q$, e pelo mesmo motivo, a recta p verifica $(p) = 2(P) + (-(P \oplus P)) - 3(\mathcal{O})$. O resto da prova é idêntica.

Quando $S = P \oplus Q$, $R = \mathcal{O}$ a função racional p/q é determinada por P e Q (a menos da multiplicação por uma constante) e faz sentido representá-la por um símbolo apropriado. Assim

256 NOÇÃO

Dados pontos P, Q *numa curva elíptica* C *com ponto no infinito* $\mathcal{O}$; *Define-se* $\mu(P, Q) = p/q$ *em que o par de rectas* p *e* q *é dado por:*

(i) *Se* $P \neq Q$ *então* $p = \mathrm{l}(P \otimes Q)$ *é a recta que passa pelos pontos* p *e* Q.

(ii) *Se* $P = Q$, *então* $p = \mathrm{l}(\mathcal{J}^C(P))$ *é a recta tangente à curva em* P.

(iii) *Se* $P \neq -Q$ *então* $q = \mathrm{l}((P \oplus Q) \otimes \mathcal{O})$.

(iv) *Se* $P \oplus Q = \mathcal{O}$ *então* $q = \mathrm{l}(\mathcal{J}^C(\mathcal{O}))$ *é a recta tangente à curva em* $\mathcal{O}$.

257 PROPOSIÇÃO
Para todo $P, Q \in C$ *verifica-se* $(\mu(P, Q)) = (P) + (Q) - (P \oplus Q) - (\mathcal{O})$.

Prova Consequência imediata do lema 258.

EXEMPLO 49: Considere-se uma curva elíptica sobre o corpo $\mathbb{Q}$ aqui determinada pela sua parte afim

$$E\colon y^2 = x^3 - x + 1$$

Nessa curva tomemos por referência um "ponto central" R de coordenadas afins $(0, 1)$.

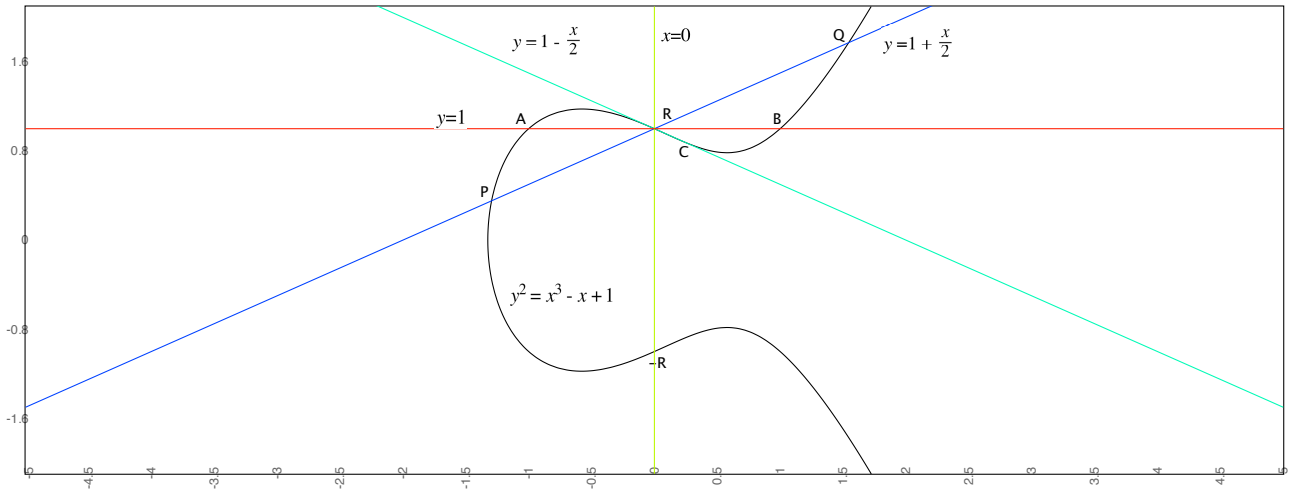


Figura 7: Rectas sobre uma curva elíptica

Vamos considerar a família das rectas que passam por este pontos, como se ilustra na figura 7. Contando com o ponto no infinito P_∞ cada uma das rectas "intersecta" a curva em mais outros 2 pontos.

Na figuras estão indicadas 4 dessas rectas:

1. Uma recta $y = 1 + \frac{x}{2}$ que intersecta a curva em outros dois pontos que designaremos por P e Q.

A recta é determinada pelo polinómio homogéneo

$$x - 2y + 2z$$

que, como elemento de $K[C]$, tem zeros precisamente nesses três pontos. Portanto

$$(x - 2y + 2z) \;=\; (P) + (R) + (Q)$$

Resolvendo o sistema de equações definido pela recta e pela equação da curva pode-se calcular as coordenadas de P e Q. Verifica-se facilmente que

$$P = [\frac{1 - \sqrt{129}}{8} \;,\; \frac{15 + \sqrt{129}}{16} \;,\; 1] \qquad Q = [\frac{1 + \sqrt{129}}{8} \;,\; \frac{17 + \sqrt{129}}{16} \;,\; 1]$$

2. Uma recta $y = 1$ que intersecta a curva em outros dois pontos A e B.
   Em coordenada projectivas o polinómio homogéneo definidor da recta é

$$y - z$$

e os pontos A e B têm coordenadas $[-1, 1, 1]$ e $[1, 1, 1]$ respectivamente. Neste caso temos

$$(y - z) \;=\; (A) + (R) + (B)$$

3. Uma recta $x = 0$ que intersecta a curva (para além de R) no ponto $-R = [0, -1, 1]$ e no ponto do infinito $P_\infty = [0, 1, 0]$.
   O polinómio homogéneo que define esta recta é simplesmente x. Por isso tem-se

$$(x) \;=\; (R) + (-R) + (P_\infty)$$

4. Finalmente temos uma recta $y = 1 - \frac{x}{2}$ que é tangente à curva no ponto R.
   Isto equivale a dizer que o polinómio homogéneo correspondente

$$x + 2y - 2z$$

   tem um zero duplo em R. Resolvendo o sistema de equações verifica-se que tem um terceiro zero no ponto C de coordenadas $[1/4, 7/8, 1]$. Por isso

$$(x + 2y - 2z) \;=\; 2(R) + (C)$$

A soma dos três pontos de uma curva elíptica que estão sobre uma mesma recta é sempre P_∞. Em cada uma destas 4 rectas, um dos pontos é sempre R; logo, a soma dos dois restantes tem de ser constante e igual a $-R$.

$$P \oplus Q = A \oplus B = R \oplus C = -R \oplus P_\infty = -R$$

Combinando as rectas duas a duas construímos várias funções racionais e determinamos os respectivos divisores; por

exemplo

$$\left(\frac{x-2y+2z}{y-z}\right) = (P)+(Q)-(A)-(B)$$

$$\left(\frac{x-2y+2z}{x}\right) = (P)+(Q)-(-R)-(P_\infty)$$

$$\left(\frac{x+2y-2z}{y-z}\right) = (R)+(C)-(A)-(B)$$

etc $\cdots$

8.4 Isogenias e Grupos de Torsão

Vamos continuar a considerar curvas projectivas de genus 1 com um ponto $\mathcal{O}$ definidas no espaço projectivo $\mathbb{P}^2$ gerado por um corpo finito K. Uma tal curva genérica, que designamos por **curva elíptica**, é representada por E/K.

No que se segue vamos sempre assumir que o corpo K é finito e, por isso, o número de pontos de E/K, representado por $|E/K|$ é finito.

Vimos que estas curvas estava definida uma estrutura de um grupo abeliano $\langle E/K, \oplus, \mathcal{O}\rangle$, com a operação do grupo apresentada na definição 246 (ver página 517). Temos, por isso, um grupo abeliano finito.

O objectivo do uso de curvas elípticas em Criptografia reside na possibilidade de construção de grupos cíclicos que sejam sub-grupos deste grupo finito. Para isso vamos considerar o **produto escalar** também apresentado na definição atrás referida.

Nessa mesma definição, para cada $n \in \mathbb{Z}$, definimos uma função $[n]\colon E/K \to E/K$ que mapeia um ponto genérico $P \in E/K$ num ponto $[n]\,P$.

Não é difícil verificar que esta função preserva a estrutura do grupo abeliano. Isto é $[n]\,(P \oplus Q) = [n]\,P \oplus [n]\,Q$ e $[n]\,\mathcal{O} = \mathcal{O}$. De facto esta função pertence a uma classe de transformações entre pontos que se designa por **isogenias** que, por seu lado, estão dentro da classe dos **homomorfismos** de curvas.

258 NOÇÃO

Dada uma curva elíptica $C = E/K$ *uma* **aplicação racional** *é um triplo de polinómios homogéneos do mesmo grau* $\phi = [\phi_1, \phi_2, \phi_3] \in \overline{\mathbb{K}}[x, y, z]^3$ *em que pelo menos um deles não pertence a* I_C .

A aplicação ϕ *é equivalente à aplicação* $\psi = [\psi_1, \psi_2, \psi_3]$, *e escreve-se* $\phi \cong \psi$, *quando,*

$$\phi_i\,\psi_j - \phi_j\,\psi_i \;\in\; I_{E/K} \qquad \textit{para todo} \quad i, j \in 1, 2, 3$$

As classes de equivalência definidas no espaço das aplicações racionais pela relação de equivalência $\cong$, *formam a espaço dos* **homomorfismos** $\mathrm{Homo}(C)$.

É fácil verificar que, dado um qualquer ponto da curva $P \in C$, cada homomorfismos $\phi \in \mathrm{Homo}(C)$ associa P a um ponto bem definido do espaço $\mathbb{P}^2$. Para a determinação desse ponto é indiferente qual a aplicação racional que se escolhe (dentro da mesma classe) ou qual o triplo de coordenadas que usamos para P. Fica definida assim uma função $\phi\colon C \to \mathbb{P}^2$.

Interessa-nos estender esta noção da definir homomorfismos entre duas curvas elípticas. Temos assim

259 NOÇÃO

No seguimento da definição 261 seja C' *for uma segunda curva elíptica sobre o mesmo corpo* K. *Um homomorfismo* ϕ *determina um* **homomorfismo de curvas** *se se verifica* $\phi(P_\infty) = P_\infty$ *e*

$$p(\phi_1(x, y, z)\ ,\ \phi_2(x, y, z)\ ,\ \phi_3(x, y, z)) \in I_C \qquad \textit{para todo} \quad p \in I_{C'} \tag{162}$$

O conjunto dos homomorfismos entre C *e* C' *representa-se por* $\mathsf{Homo}(C, C')$.

Se $\phi \in \mathsf{Homo}(C, C')$ *é surjectivo então designa-se por* **isogenia** *entre curvas.*

As condições aqui impostas, para além de assegurar que o ponto no infinito seja sempre mapeado no ponto no infinito, asseguram que qualquer ponto $P \in C$ é mapeado num ponto $\phi(P) \in C'$.

O resultado fundamental, que se pode provar recorrendo aos divisores, é

260 TEOREMA
Se $\phi \colon C \to C'$ *é uma isogenia entre curvas elípticas* C *e* C' *então* ϕ *preserva a estrutura dos grupos abelianos.*

Vimos que as funções $[n]\cdot \colon C \to C$ são isogenias dentro da mesma curva C. Um outro exemplo particularmente importante de isogenia é a **isogenia de Frobenius**.

Recordemos que K é um corpo finito, por hipótese. Assim terá a forma $K = \mathbb{F}_{p^d}$ em que o primo p é a sua característica e d a sua sua dimensão. Recordemos que a função $\sigma_p \colon x \mapsto x^p$ é designada por **morfismo de Frobenius** no corpo K e é um endomorfismo (preserva a estrutura algébrica e é um isomorfismos) que fixa os elementos de $\mathbb{F}_p$ nesse corpo.

□

No que se segue vamos considerar uma curva elíptica $C = E/K'$ definida numa extensão K' de K. Note-se que K' continua a ter característica p e o polinómio que determina a curva é um elemento de $K[x, y, z]$.

A **isogenia de Forbenius** estende este morfismo aplicando σ_p componente-a-componente. Ou seja

$$\sigma_p : (x, y, z) \mapsto (x^p , y^p , z^p) \tag{163}$$

Não é verdade, normalmente, que σ_p seja uma isogenia de uma curva C para a mesma curva C. No entanto é fácil construir uma segunda curva C' de tal forma que σ_p seja uma isogenia entre estas duas curvas.

EXEMPLO 50: Suponhamos que $K = \mathbb{F}_{2^n}$ é um corpo de característica 2 e que que C é uma curva elíptica cuja componente afim é determinada pelo polinómio

$$y^2 + x\,y + x^3 + \upsilon \qquad \text{com} \quad \upsilon \in K$$

Se for $y^2 + x\,y + x^3 + \upsilon = 0$, elevando ao quadrado temos

$$(y^2)^2 + (x^2)\,(y^2) + (x^2)^3 + \upsilon^2 = 0$$

Então, se (x, y) define uma raiz do polinómio original, o par (x^2, y^2) define uma raiz do polinómio $y^2 + x\,y + x^3 + \upsilon^2$ que é idêntico ao anterior excepto no facto de a constante μ ser substituída por μ^2.

Porém, considerando $K = \mathbb{F}_p$ e atendendo que E/K' é definida por um polinómio $p \in \mathbb{F}_p[x, y, z]$ (i.e, todos os coeficientes do polinómio pertencem a $\mathbb{F}_p$), então, dado que σ_p fixa os elementos de $\mathbb{F}_p$, a isogenia de Frobenius

$(x, y) \mapsto (x^p, y^p)$ preserva o polinómio. Por isso σ_p é, neste caso, uma **endo-isogenia** ; isto é, uma isogenia da curva C para, de novo, a curva C.

Uma generalização simples consiste em considerar o homomorfismo σ_P^d (a potência de ordem d da isogenia de Frobenius). Neste caso qualquer polinómio $p \in K[x, y, z]$ é fixado por esta aplicação e, por isso, ela define uma endo-isogenia.

Note-se que, como as coordenadas x, y estão contidas na extensão K', elas não são fixadas pelo morfismo; por isso, normalmente, será $(x, y) \neq (x^{p^d}, y^{p^d})$.

EXEMPLO 51: Neste exemplo vamos usar curvas elípticas sobre o corpo $\mathbb{F}_4$ e sobre a sua extensão $\mathbb{F}_{16}$. Para determinar os elementos de $\mathbb{F}_4$ usaremos polinómio característico $\beta^2 + \beta + 1$. Assim os elementos de $\mathbb{F}_4$ serão $\{0, 1, \beta, 1 + \beta\}$.

Considere-se agora a curva $E/\mathbb{F}_4 : y^2 + x\,y + x^3 + 1$.

Pode-se verificar que esta curva tem 7 pontos afins, para além do ponto no infinito P_∞. São eles

$$\{ (0,1) \ , \ (1,0) \ , \ (1,1) \ , \ (\beta,0) \ , \ (\beta,\beta) \ , \ (\beta+1,0) \ , \ (\beta+1,\beta+1) \}$$

O morfismo de Frobenius é, neste caso, $x \mapsto x^2$. Notando que $\beta^2 = \beta + 1$ e que $(\beta + 1)^2 = \beta$ neste corpo, a

isogenia de Frobenius $(x, y) \mapsto (x^2, y^2)$ mapeia os pontos da lista anterior, respectivamente, em

$$\{ (0,1) \ , \ (1,0) \ , \ (1,1) \ , \ (\beta+1, 0) \ , \ (\beta+1, \beta+1) \ , \ (\beta, 0) \ , \ (\beta, \beta)\}$$

Portanto, neste caso, a isogenia mapeia pontos da curva em pontos da mesma curva.

Considere-se agora uma outra curva $E/\mathbb{F}_4 : y^2 + x\,y + x^3 + \beta$.

Os seus pontos afins são apenas

$$\{(0, \beta+1) \ , \ (\beta, 1) \ , \ (\beta, \beta+1)\}$$

A isogenia de Frobenius $(x, y) \mapsto (x^2, y^2)$ estabelece-se entre esta curva e a curva $E/\mathbb{F}_4 : y^2 + x\,y + x^3 + (\beta+1)$ cujos pontos são

$$\{(0, \beta) \ , \ (\beta+1, 1) \ , \ (\beta+1, \beta)\}$$

□

Vamos agora considerar curvas sobre a extensão $\mathbb{F}_{16}/\mathbb{F}_4$ usando o polinómio característico $\alpha^4 + \alpha + 1$.

A imersão de $\mathbb{F}_4$ em $\mathbb{F}_{16}$ faz-se pelo morfismo que mapeia $\beta \mapsto \alpha^2 + \alpha$; de facto, calculando $\beta^2 + \beta + 1$ tem-se

$$\beta^2 + \beta + 1 \quad \to \quad (\alpha^4 + \alpha^2) + (\alpha^2 + \alpha) + 1 \quad \to \quad \alpha^4 + \alpha + 1 \quad \to \quad 0$$

Vamos considerar, nesta extensão, curvas definidas pelos dois polinómios considerados atrás (tendo em atenção a substituição $\beta \longrightarrow \alpha^2+\alpha$). É óbvio que as curvas $E/\mathbb{F}_{16} : y^2+x\,y+x^3+1$ e $E/\mathbb{F}_{16} : y^2+x\,y+x^3+(\alpha^2+\alpha)$ contêm todos os pontos das curvas correspondentes em $\mathbb{F}_4$ e ainda alguns pontos adicionais.

A componente afim de $E/\mathbb{F}_{16} : y^2 + x\,y + x^3 + 1$ contém 15 pontos em vez dos 7 pontos em $\mathbb{F}_4$

$(0,1)$, $(1,0)$, $(1,1)$, $(\alpha^3, \alpha^2+\alpha+1)$, $(\alpha^3, \alpha^3+\alpha^2+\alpha+1)$, $(\alpha^2+\alpha, 0)$, $(\alpha^2+\alpha, \alpha^2+\alpha)$, $(\alpha^3+\alpha^2, \alpha^2+\alpha)$, etc

Na componente afim de $E/\mathbb{F}_{16} : y^2 + x\,y + x^3 + \alpha^2 + \alpha$ a diferença é mais substancial; em vez dos 3 pontos em $\mathbb{F}_4$ tem-se 23 pontos em $\mathbb{F}_{16}$.

Ambos os polinómios mantêm-se invariantes pela transformação $x \mapsto x^4$; por isso o morfismo $(x, y) \mapsto (x^4, y^4)$ é uma endo-isogenia.

Regressemos às isogenias sobre uma curva elíptica $E/K : \phi$, com $\phi \in K_0[x, y, z]]$ e K um extensão K/K_0 . Seja p a característica de K_0 (e também de K).

261 NOÇÃO
O núcleo de $[n]\colon E/K \longrightarrow E/K$ *(isto é, o conjunto* $\{\, P \in E/K \mid [n]P = \mathcal{O} \,\}$*) é representado por* $E/K[n]$ *e designa-se por* **grupo de torção de** E/K **de ordem** n*. Os pontos de* $E/K[n]$ *designam-se por* K-**pontos de torção** *de ordem* n*.*

O seguinte resultado é fundamental ao nosso estudo.

262 TEOREMA

Se K é algebricamente fechado e a característica p é primo relativamente a n, então $E/K[n] \simeq \mathbb{Z}_n \times \mathbb{Z}_n$. Se $n = p^s$, para algum $s \geq 1$, então um de dois casos ocorre: ou $E/K[n] = \{\mathcal{O}\}$ ou então $E/K[n] \simeq \mathbb{Z}_{p^s}$.

Se ocorrer a primeira hipótese $E/K[p^s] = \{\mathcal{O}\}$, a curva diz-se **super-singular** e esta propriedade é independente do valor de s. No caso contrário ($E/K[p^s] \simeq \mathbb{Z}_{p^s}$) a curva diz-se **ordinária**.

É importante notar-se que este teorema exige que K coincida com $\bar{K_0}$ (o fecho algébrico do corpo onde característica p onde o polinómio gerador ϕ está definido). Quando K é uma outra qualquer extensão de K_0, contida em $\bar{K_0}$, o K-grupo de torção tem, naturalmente, menos pontos.

Obviamente, se escolhermos um qualquer $P \neq \mathcal{O} \in E/K[n]$, qualquer ponto $Q = [k]P$ continua a ser um elemento $E/K[n]$. Por isso a órbita de P é um sub-grupo cíclico de $E/K[n]$.

Sabe-se, da teoria genérica dos grupos de torção, que $E/K[n]$ é a soma directa dos seus subgrupos primos. Isto é a soma directa de subgrupos de ordem da forma q^s, sendo q um primo. Nomeadamente em criptografia são extremamente importantes os sub-grupos de ordem prima dos grupos de torção.

8.5 Aritmética nas Curvas Elípticas

Vimos na secção anterior que uma **curva elíptica** sobre o corpo $\mathbb{K}$ é determinada, em coordenadas afins, por um polinómio $\phi \in \overline{\mathbb{K}}[x, y]$ da forma de Weitstraß que, de forma simplificada, se pode escrever

$$\phi = y^2 + y\,h(x) + f(x) \tag{164}$$

com $h, g \in \mathbb{K}[x]$ sendo $\deg(h) \leq 1$ e $\deg(f) = 3$ de tal forma que as derivadas $\partial\phi/\partial x$ e $\partial\phi/\partial y$ não se anulam simultâneamente.

Vimos também que as raízes deste polinómio definem a chamada **parte afim** da curva. Existe ainda um outro ponto da curva, que representamos por P_∞ ou $\mathcal{O}$, que completa a curva e que não pertence à parte afim.

□

Sendo $h = a_1\,x + a_3$ e $f = x^3 + a_2\,x^2 + a_4\,x + a_6$, a condição de não anulação simultânea das derivadas parciais ocorre se e só se

$$16\,{a_2}^2 - 8\,a_2{a_1}^2 + {a_1}^4 - 48\,a_4 + 24\,a_1a_3 \;<\; 0$$

Se $\mathbb{K}$ tem característica diferente de 2 ou 3 então, através de substituição de variáveis $y \leftarrow 2\,y + h$ e $f \leftarrow f - h^2/4$, transforma a forma genérica de Weitsrass em (164) simplificando-a em

$$\phi = y^2 + f(x) \tag{165}$$

Se $\mathbb{K}$ tem característica 3 a mudança de variáveis $y \leftarrow y + h$ e $f \leftarrow f + h^2$ conduz à mesma forma equivalente $\phi = y^2 + f(x)$. Em ambos os casos ϕ define uma curva elíptica se e só se f não tem raízes múltiplas; isto é, todas as raízes de f têm de ser distintas.

Quando a característica do corpo K é 2, a forma de Weierstraß pode ainda ser simplificada. Para isso vamos definir $\xi \doteq \partial h/\partial x$. Como h tem grau 0 ou 1, o valor ξ é um elemento de $\mathbb{K}$ que pode ser nulo (se h tiver grau 0) ou não. Vamos considerar separadamente estes dois casos.

$\xi = \partial h/\partial x = 0$ Estas curvas dizem-se **super-singulares**.

Neste caso $h(x)$ é uma constante e, escrevendo $f(x) = x^3 + a\,x^2 + b\,x + c$, pode-se efectuar a mudança de variáveis $x \leftarrow x + a$, $\mu = a^2 + b$, $\upsilon = ab + c$ que conduz à curva equivalente

$$\phi = y^2 + hy + x^3 + \mu x + \upsilon \tag{166}$$

Nesta curva tem-se $\partial\phi/\partial y = h$ e a curva será não-singular (isto é, ϕ determina uma curva elíptica) se e só se for $h \neq 0$.

$\xi = \partial h/\partial x \neq 0$ Neste caso tem-se $h(x) = \kappa + \xi\,x$. A curva diz-se **ordinária**.

Usando a mesma representação de f existe uma mudança de variáveis que conduz à forma simplificada,

$$\phi = y^2 + xy + x^3 + \mu x^2 + \upsilon \tag{167}$$

Aqui $\partial\phi/\partial x = y + x^2$ e $\partial\phi/\partial y = x$. Isto significa que a curva é não-singular desde que não contenha o ponto $(0,0)$.

Para as curvas ordinárias existe ainda uma simplificação adicional. Suponhamos que se toma $c = \mu$ ou o seu complemento $c = 1 + \mu$ consoante o traço de μ é 0 ou 1. Em qualquer dos casos tem-se $\mathrm{Tr}(c) = 0$.

Neste caso a equação $\lambda^2 + \lambda + c = 0$ tem solução. A mudança de variáveis $y \mapsto y + \lambda\, x$ conduz à curva equivalente

$$\phi' \;=\; y^2 + xy + x^3 + (c+\mu)\, x^2 + \upsilon$$

Escolhendo $c = \mu$ ou $c = 1 + \mu$ obtém-se as curvas

$$\begin{aligned} \phi &= y^2 + xy + x^3 + \upsilon \qquad \text{ou} \qquad (168)\\ \phi &= y^2 + xy + x^3 + x^2 + \upsilon \end{aligned}$$

que são as formas mais genéricas de curvas ordinárias em corpos de característica 2. Qualquer destas formas pode ser adoptada escolhendo uma ou outra consoante se pretende um coeficiente de x^2 com traço 0 ou com traço 1.

□

Na secção anterior vimos que as curvas elípticas $E/\mathbb{K}$ se identificam, no contexto geral do estudas das curvas

planas, com as curvas de genus 1 que têm um ponto identificado $\mathcal{O}$. Desta interpretação resultaram, na secção anterior, dois resultados essenciais sobre a estrutura algébrica das curvas elípticas:

1. Em primeiro lugar a proposição 247 (ver pag. 517) identifica em cada curva $E/\mathbb{K}$ a estrutura de um grupo abeliano $\langle E/\mathbb{K}\ ,\ \oplus\ ,\ \mathcal{O}\rangle$.
2. Em segundo lugar que a operação de grupo $\oplus$ tem uma simples interpretação geométrica; o lema 257 diz-nos que, se for $S = P \oplus Q$ então $P, Q, -S$ estão sobre uma mesma recta (são colineares). O mesmo lema diz-nos também que, para todo ponto P, o triplo de pontos $P, -P, \mathcal{O}$ também está sempre sobre uma mesma recta. Se P tiver coordenadas $[P_1, P_2, P_3]$, tem-se $P \otimes \mathcal{O} = [P_3, 0, -P_1]$ e a recta respectiva será $xP_3 - zP_1$

Esta interpretação geométrica permite facilmente encontrar fórmulas explícitas para calcular as coordenas afins de $P \oplus Q$ e de $-P$ em função das coordenadas afins de P e Q. Assumimos a forma genérica de Weierstraß em coordenadas afins

$$y^2 + a_1\, x\, y + a_3\, y - x^3 - a_2\, x^2 - a_4\, x - a_6 \tag{169}$$

1. **Determinar $-P$**
   Se P tem coordenadas afins $[p_1, p_2]$ então a recta $\mathbf{l}(P \otimes \mathcal{O})$ é $xp_3 - zp_1$.
   O terceiro ponto de intersecção com a curva é $-P$. Atendendo a que P pertence à curva, verifica-se que

$$-P \ = \ (p_1, -\mu) \qquad \text{sendo} \quad \mu = p_2 + a_1\, p_1 + a_3 \tag{170}$$

2. **Determinar $P \oplus Q$**
   Sekam $P = (p_1, p_2)$ e $Q = (q_1, q_2)$ as coordenadas afins dos pontos.
   Determina-se a recta $p = \mathbf{l}(P \oplus Q)$ que passa por P e Q. Caso seja $P = Q$ essa recta é a tangente à curva em P. Determina-se o terceiro ponto R de intersecção da recta com a curva e faz-se $P \oplus Q = -R$.
   Temos duas situações para cálculo do declive λ da recta p.

$$\lambda = (p_2 - q_2)/(p_1 - q_1) \qquad \text{quando} \quad P \neq \pm Q \tag{171}$$

$$\lambda = (3\,p_1^2 + 2\,a_2\,p_1 + a_4 - p_2)/(2\,p_2 + a_1\,p_1 + a_3) \qquad \text{quando} \quad P = Q \tag{172}$$

   Então

$$P \oplus Q = (\,\eta\ ,\ \lambda\,(p_1 - \eta) - \mu\,) \tag{173}$$

$$\text{sendo} \quad \eta = \lambda^2 + a_1\,\lambda - a_2 - p_1 - p_2 \qquad \text{e} \qquad \mu = p_2 + a_1\,\eta + a_3$$

Esta é a forma mais geral para determinar o valor da soma $P \oplus Q$. Consoante a carcaterística do corpo K e as formas simplificadas de curvas que daí resultam várias simplificações e optimizações são possíveis.

8.6 Emparelhamentos de Tate e Weil

Sejam G_1, G_2 dois sub-grupos de curvas elípticas ambos de expoente n; isto é, para todo $p \in G_1, G_2$ tem-se $[n]P = \mathcal{O}$. Considere-se ainda um grupo cíclico Γ de ordem n escrito de forma multiplicativa. Recordemos que

263 NOÇÃO

Um **emparelhamento** *é uma função* $e\colon G_1 \times G_2 \to \Gamma$ *que é*

- ***bilinear:*** *para todo* $P, P' \in G_1$ *e* $Q, Q' \in G_2$ *tem-se* $e(P \oplus P', Q) = e(P,Q) \cdot e(P',Q)$ *e* $e(P, Q \oplus Q') = e(P,Q) \cdot e(P,Q')$.
- ***não-degenerado:*** *para todo* $P \neq \mathcal{O} \in G_1$ *existe* $Q \in G_2$ *tal que* $e(P,Q) \neq 1$ *e, dualmente, para todo* $Q \neq 0 \in G_2$ *existe* $P \in G_1$ *tal que* $e(P,Q) \neq 0$.

Como consequência imediata da definição, temos

264 FACTO

Se $e\colon G_1 \times G_2 \to \Gamma$ *é um emparelhamento, então para todo* $P \in G_1$ *e todo* $Q \in G_2$

- $e(P,0) = e(0,Q) = 1$.
- $e(-P,Q) = e(P,-Q) = e(P,Q)^{-1}$
- $e([k]P, [j]Q = e(P\,Q)^{k\,j}$ *para todos* $k, j \in \mathbb{Z}$.

Emparelhamento de Tate

Escolha do corpo

1. *Escolha-se um primo p e um* ***corpo base*** $K_0 = \mathbb{F}_q$ *de característica p.*
2. *Escolha-se um primo r distinto de p. Seja k a ordem de q em $\mathbb{Z}_r^*$: isto é, o menor inteiro k tal que $q^k = 1 \mod r$. Seja $l = s \cdot (s^{-1} \mod r)$ sendo $s = (q^k - 1)/r$.*
3. *Seja $\boldsymbol{\mu}_r$ o conjunto de todas as raízes do polinómio $(X^n - 1) \in \overline{K_0}[X]$; isto é, as* ***r-raízes da unidade*** *de K_0.*
4. *Define-se $K = K_0(\boldsymbol{\mu}_r)$ como a menor extensão de K_0 que contém $\boldsymbol{\mu}_r$; isto é, a* ***r-extensão ciclotómica*** *de K_0.*
5. *Seja $(K^*)^r = \{ u^r \mid u \in K \}$ o subgrupo de K^* formado pelas r-potências de K. O grupo quociente $K^*/(K^*)^r$ é representado por K_r^*.*

O estudo das raízes da unidade, extensões ciclotómicas, polinómios ciclotómicos, etc, é uma tema da Álgebra bastante analisado[71]. Os principais resultados que nos interessam podem ser sumariados no seguinte teorema.

265 Teorema (Extensão ciclotómica)
Nas condições acima enumeradas,

[71] Consultar, por exemplo, o livro de *Steven Roman*, Field Theory, 2nd edn, Springer Verlag, nº 158 da série *Graduate Texts in Mathematics*, principalmente os Capítulos 11 e 12, para uma síntese desses resultados.

1. *O corpo K é uma extensão de grau k de $\mathbb{F}_q$ e identifica-se com $\mathbb{F}_{q^k}$.*
2. *O grupo de Galois, $\mathbb{G}(K/K_0)$ é um grupo cíclico de ordem k gerado pelo automorfismo $\sigma_q : x \mapsto x^q$.*
3. *O conjunto das r-raízes da unidade $\boldsymbol{\mu}_r$ forma um subgrupo ciclico de ordem r de K^* e o morfismo $x \mapsto x^l$ induz um isomorfismo entre o grupo quociente K_r^* e $\boldsymbol{\mu}_r$.*

Nota A relação de equivalência que gera o grupo quociente $K^*/(K^*)^r$ é, neste caso,

$$x \simeq_r y \quad \Leftrightarrow \quad (\exists u \in K^*)\,[\, x \cdot y^{-1} = u^r \,] \tag{174}$$

Pela definição de k, $(q^k - 1)$ é divisível por r; seja $s = (q^k - 1)/r$. Então, para qualquer $x \in K^*$, x^s é uma r-raíz da unidade. Sendo $l = s \cdot (s^{-1} \mod r)$, também x^l é uma r-raiz da unidade porque l é um múltiplo de s. Por outro lado, $l = 1 \mod r$ o que significa que $l - 1$ é um múltiplo de r e, por isso, $x^l \cdot x^{-1}$ é uma r-potência de um elemento de K^*. Ou seja, x e x^l são equivalentes.

Grau de embebimento

Vimos que, sendo k a ordem de q em $\mathbb{Z}_r^*$, o corpo K é uma extensão de grau k de $\mathbb{F}_q$. A constante k chama-se **grau de embebimento** de K_0 em K.

Na escolha do corpo, as várias constantes (q, r, k, l) podem-se determinar de vários modos. Por exemplo, pode-se fixar q e r e calcular grau de embebimento k como o menor inteiro k tal que r divide $q^k - 1$. Em alternativa pode-se fixar q e um grau de embebimento aceitável k, e determinar o maior primo r que divide $q^k - 1$.

Para uma implementação eficiente de emparelhamentos interessa-nos ter o grau de embebimento k tão pequeno quanto possível. Isto porque k pequeno implica que o número de elementos q^k do corpo K é relativamente pequeno e, consequentemente, são necessários poucos de bits para representar pontos de curvas elípticas E/K. Os valores ideais para k serão 2 ou 3; de facto 6 é considerado o limite superior para uma implementação razoável de emparelhamentos.

Por outro lado, r vai determinar a ordem dos grupos cíclicos e, por isso, convém que seja um primo tão grande quanto possível. Quanto maior for r mais complexo será a resolução dos problemas básicos dos grupos cíclicos (DLP, CDHP, etc.) e, por isso, mais seguras serão as técnicas criptográficas assentes nesses grupos. No mínimo r deve ser um primo só representável com 160 bits ou mais.

Estes objectivos contraditórios conduzem a uma escolha criteriosa de r que seja suficientemente grande para os grupos cíclicos serem seguros e, simultaneamente, determine uma valor de k pequeno.

EXEMPLO 52: Considere-se o corpo $K_0 = \mathbb{F}_7$. Pode-se verificar que a curva $E/\mathbb{F}_7 : y^2 - x^3 - x + 3$ tem exactamente 9 pontos e que esses pontos são gerados como múltiplos do ponto $P = (0, 2)$; os pontos $[k]P$, com $k = 1..9$, são

$$(0,2)\ ,\ (4,4)\ ,\ (5,6)\ ,\ (6,3)\ ,\ (2,0)\ ,\ (6,4)\ ,\ (5,1)\ ,\ (4,3)\ ,\ (0,5)\ ,\ P_\infty$$

Escolha-se uma ordem $r \neq p$ que seja um número primo; por exemplo $r = 13$. Neste caso $\boldsymbol{\mu}_{13}$ é formado pela unidade 1 e pelas raízes $\zeta \neq 1$ do polinómio ciclotómico $\Phi_{13}[X] = \frac{(X^{13}-1)}{(X-1)} \in \mathbb{F}_7[X]$; isto é, os ζ que verificam $\zeta^{12} + \zeta^{11} + \cdots + \zeta + 1 = 0$.

O grau de embebimento é o menor k tal que 13 divide $7^k - 1$; pode-se verificar que $k = 12$ e, portanto, K tem 7^{12} elementos. Neste caso, tem-se $l = 3194143200$. Podia-se também verificar que, aqui, tem de ser $k = r - 1$ porque o polinómio ciclotómico Φ_{13} é absolutamente irredutível em $\mathbb{F}_7$. A 13-extensão ciclotómica de $\mathbb{F}_7$, $\mathbb{F}_7(\boldsymbol{\mu}_{13})$ tem 7^{12} elementos cuja forma genérica é $a_0 + a_1\,\zeta + a_2\,\zeta^2 + \cdots + a_{11}\,\zeta^{11}$, com $a_k \in \mathbb{F}_7$.

Este valor para o grau de embebimento é péssimo: cada ponto da curva E/K exige 12 vezes mais bits que um ponto sobre a curva $E/\mathbb{F}_7$. E isto para ter apenas 13 elementos no grupo cíclico.

No entanto um grau de embebimento $k = 5$ (dentro da gama dos valores aceitáveis e bastante inferior ao valor de 12 atrás encontrado) conduz a um primo $r = 2801$ que divide $7^5 - 1$. Note-se que, mesmo para um k menor, o primo r resultante é bastante maior do que 13. O parâmetro l é também consideravelmente menor; tem-se $l = 2802 = r + 1$.

Escolha da curva

Seja $C = E/K : \phi$ uma curva elíptica cuja ordem $|E/K|$ é um múltiplo de r.

Para garantir que a ordem r divida a ordem da curva $E/K : \phi$, seria possível começar por exigir, na escolha de r, que a ordem da curva base $E/K_0 : \phi$ fosse já um múltiplo de r. Porém isso implica que o corpo de base K_0 tivesse, já à partida, um número de elementos suficientes para que tal ocorra.

Em alternativa pode-se tentar escolher (eventualmente, por tentativas sucessivas) a ordem r, o grau de embebimento k e o polinómio ϕ de modo que todas estas condições ocorram: o grau de embebimento k é pequeno, o corpo base $\mathbb{F}_q$ é pequeno, r é grande e r divide a ordem da curva definida no corpo $\mathbb{F}_{q^k}$.

□

A ordem r determina em C vários subgrupos com interesse:

- O grupo de torção $C[r] = \{ P \in C \mid [r]P = \mathcal{O} \}$.
- O grupo $r\,C = \{[r]U \mid U \in C\}$ dos r-múltiplos de pontos da curva; $r\,C$ é a imagem da isogenia $[r]\cdot$.
- O grupo quociente $C_r = C/rC$.

Note-se a analogia entre C_r e o grupo K_r^*: os elementos de C_r são as classes de equivalência geradas em C, pela relação $P \cong Q \Leftrightarrow (\exists U \in C)\,[\,P - Q = [r]U\,]$. Note-se também que as condições impostas em r e p, implicam que todos os grupos $C[r], C_r, K_r^*$ e $\boldsymbol{\mu}_r$ têm exactamente r elementos.

266 Definição (Emparelhamento de Tate)
O ***emparelhamento de Tate*** *e ordem r em C, é a função*

$$\mathbf{e}_r : C[r] \times C_r \to \boldsymbol{\mu}_r$$

tal que, para todo ponto $P \in C[r]$ e classe de equivalência $q \in C_r$, o valor de $\mathbf{e}_r(P, q)$ é o elemento de $\boldsymbol{\mu}_r$ gerado da seguinte forma:

1. *Determina-se $f \in K(C)$ tal que $(f) = r(P \oplus R) - r(R)$, para algum ponto $R \in C$.*

Porque $P \in C[r]$, a função (f) existe. Note-se que P verifica $[r]P = \mathcal{O}$ e, pela definição 246, tem-se $r(P) - r(\mathcal{O}) \sim 0$; logo, para todo R, será $r(P \oplus R) - r(R) \sim 0$. Usualmente escolhe-se $R = \mathcal{O}$.

2. *Escolhe-se $Q \in q$ e determina-se um divisor $D \sim (Q) - (\mathcal{O})$ que tenha um suporte disjunto do de (f).*

   Normalmente, basta escolher um ponto $S \in C$, tal que tanto S como $Q \oplus S$ não sejam nem zero nem pólo de f, e definir $D = (Q \oplus S) - (S)$. Pela definição 246, tem-se $(Q \oplus S) - (\mathcal{O}) \sim (Q) + (S) - 2\,(\mathcal{O})$ e, portanto, $(Q \oplus S) - (S) \sim (Q) - (\mathcal{O})$.

3. *Determina-se $f(D)$ e define-se $\mathbf{e}_r(P, q) = f(D)^l$; ou seja, como a r-raiz da unidade equivalente a $f(D)$.*

   Como D não contém pólos ou zeros de f, a função f é regular em D e tem-se $f(D) \neq 0$; logo $f(D) \in K^*$. O valor $f(D)^l$ determina a raiz da unidade $\mu \in \boldsymbol{\mu}_r$ tal que $f(D) \simeq_r \mu$.

No passo (1) temos várias funções racionais $f \in K(C)$ que verificam $(f) = r(P \oplus R) - r(R)$. O valor de $\mathbf{e}(P, q)$ deve ser independente do ponto R e da função f escolhidas. É também importante ter em atenção, no passo (2), que Q é um representante da classe de equivalência $q \in C_r$ e o valor $f(D)$ depende de Q; são possíveis vários pontos Q dentro da mesma classe de equivalência $q \in C_r$ e vários divisores $D \sim (Q) - (\mathcal{O})$. Escolhendo um diferente $Q' = Q \oplus [r]U$ e um diferente $D' \sim (Q') - (\mathcal{O})$, o valor para $f(D')$ e o valor de $f(D)$ devem conduzir, no final, ao mesmo valor de $\mathbf{e}_r(P, q)$.

Em resumo: para a correcção do resultado de $\mathbf{e}(P,q)$, deve ser indiferente o ponto R, a função f, o representante Q e o divisor D, escolhidos desde que verifiquem as condições estipuladas na respectiva escolha. No entanto, a liberdade na selecção de determinados valores específicos de R, f, Q e D é importante para a eficiência da implementação deste algoritmo.

Todos estes factos levam a que o resultado $f(D)$ seja visto como um representante de uma classe de equivalência em K_r^* e, consequentemente, um elemento de $\boldsymbol{\mu}_r$. A razão porque o algoritmo na noção 269 define correctamente uma função $C[r] \times C_r \longrightarrow \boldsymbol{\mu}_r$ é resultado dos seguintes lemas.

267 LEMA *Sejam $f, g \in K(C)$ funções racionais homogéneas tais que $(f) = r(P) - r(\mathcal{O})$ e, para um qualquer ponto $R \in C$, $(g) = r(P \oplus R) - r(R)$. Então, para todo o divisor de grau zero D sobre K com suporte disjunto do de (f) e do de (g), tem-se $f(D) \simeq_r g(D)$.*

268 LEMA *Seja $f \in K(C)$ tal que que $(f) = r(P) - r(\mathcal{O})$. Sejam D e D' divisores sobre K de grau zero tais que $D \sim D'$, e com suporte disjunto do de (f). Então $f(D) \simeq_r f(D')$.*

269 LEMA *Seja $f \in K(C)$ funções racional homogénea tal que $(f) = r(P) - r(\mathcal{O})$. Sejam D e D' divisores sobre K de grau zero tais que $D \sim (Q) - (\mathcal{O})$ e $D' \sim (Q \oplus [r]U) - (\mathcal{O})$, cujo suporte é disjunto do de (f). Então $f(D) \simeq_r f(D')$.*

Prova As provas destes três lemas são uma simples aplicação dos resultados da teoria dos divisores sobre curvas elípticas. Note-se que, em todas as provas, as funções racionais f e g estão sempre definidas, na curva, a menos da multiplicação por uma constante $\lambda \neq 0$. No entanto, pelo facto 250, quando aplicadas a divisores D de grau zero, os valores de $f(D)$ e $g(D)$ são independentes dessa constante.

Lema 270 Pela definição 246 temos $(P \oplus R) - (P) - (R) + (\mathcal{O}) \sim 0$. Seja $u \in K(C)$ tal que $(u) = (P \oplus R) - (P) - (R) + (\mathcal{O})$; então $(u^r) = r(u) = r(P \oplus R) - r(R) - r(P) + r(\mathcal{O})$ e consequentemente $(g) = (u^r) + (f) = (u^r\, f)$. Donde, para alguma constante $\lambda \neq 0$, tem-se $\lambda\, g(X) = u(X)^r\, f(X)$, para todo $X \in C$ onde g e f sejam regulares. Para qualquer divisor cujo suporte seja disjunto do suporte de (f) e de (g) temos $g(D) = f(D)\, u(D)^r$ e, consequentemente, $g(D) \simeq_r f(D)$.

Lema 271 Seja $u \in K(C)$ tal que $D = D' + (u)$. Então o suporte de u é disjunto do de f e $f(D) = f(D')\, f((u))$. Pela reciprocidade de Weil (teorema 251) temos $f((u)) = u((f)) = u(r(P) - r(\mathcal{O})) = u(P)^r / u(\mathcal{O})^r$. Consequentemente $f(D) \simeq_r f(D')$.

Lema 272 Temos $D' - D \sim (Q \oplus [r]U) - (Q) \sim r(U) - r(\mathcal{O})$ e, portanto, como o suporte de (f) é disjunto do de D e D', temos $f(D')/f(D) = f(U)^r / f(\mathcal{O})^r$. Logo $f(D') \simeq_r f(D)$.

Estamos agora em condições de provar o resultado fundamental.

270 Teorema (Emparelhamento de Tate)
O algoritmo apresentada na definição 269 determina uma função $\mathbf{e}_r : C[r] \times C_r \longrightarrow \boldsymbol{\mu}_r$ *e esta função:*

1. *É um emparelhamento (noção 266),*
2. *Para todo o automorfismo* $\sigma \in \mathbb{G}(K/K_0)$, *verifica* $\mathbf{e}_r(\sigma(P), \sigma(q)) = \sigma(\mathbf{e}_r(P, Q))$.

Esboço de Prova Os lemas (270), (271) e (272), dizem-nos que, independentemente do ponto R, da função f, do representante $Q \in q$ e do divisor D, $f(D)$ é sempre um elemento da mesma classe de equivalência em K_r^*; o isomorfismo entre K_r^* e $\boldsymbol{\mu}_r$ completa a definição da função.

Para provar (1) (isto é, ver que $(P,q) \mapsto \mathbf{e}_r(P,q)$ é um emparelhamento de acordo com a noção 266) temos de provar que é bilinear e não-degenerada.

Biliear no 1^{o} argumento, $\mathbf{e}(P \oplus P', q) = \mathbf{e}(P,q) \cdot \mathbf{e}(P',q)$: Sejam $f, f' \in K(C)$ tais que $(f) = r(P) - r(\mathcal{O})$ e $(f') = r(P') - r(\mathcal{O})$. Seja $u \in K(C)$ tal que $(u) = (P \oplus P') - (P) - (P') + (\mathcal{O})$ e seja $h = f \cdot f' \cdot u^r$. Então $(h) = (f) + (f') + r(u) = r(P \oplus P') - r(\mathcal{O})$. Para um qualquer divisor $D \sim (Q) - (\mathcal{O})$, com $Q \in q$, com suporte disjunto do de h, tem-se $h(D) \simeq_r f(D)\, f'(D)$ e este valor determina $\mathbf{e}_r(P \oplus P', q)$; como $f(D)$ e $f'(D)$ determinam, respectivamente, $\mathbf{e}_r(P,q)$ e $\mathbf{e}_r(P,q)$, tem-se $\mathbf{e}_r(P \oplus P', q) = \mathbf{e}_r(P,q) \cdot \mathbf{e}_r(P',q)$.

As restantes 3 condições do emparelhamento (bilinear no 2^{o} argumento e não-degenerado no 1^{o} e 2^{o} argumentos) provam-se de forma semelhante.

Para provar (2) (invariância por automorfismos de Galois), basta ver que $\sigma(f(P)) = f_\sigma(\sigma(P))$, sendo f_σ a aplicação do automorfismo σ aos coeficientes de f. Se $(f) = r(P) - r(\mathcal{O})$ então $(f_\sigma) = r(\sigma(P)) - r(\mathcal{O})$. Sendo $D \sim (Q) - (\mathcal{O})$, tem-se $\mathbf{e}_r(\sigma(P), \sigma(Q)) = f_\sigma(\sigma(D)) = \sigma(f(D)) = \sigma(\mathbf{e}_r(P,Q))$.

Emparelhamento de Weil

Fixemos um corpo base $K_0 = \mathbb{F}_q$, e uma ordem $r \neq p$. Como anteriormente $\boldsymbol{\mu}_r$ é o grupo cíclico das r-raízes da unidade em K_0.

Seja $E/\overline{K_0}$ uma curva elíptica sobre o fecho algébrico de K_0, cuja ordem $|E/\overline{K_0}|$ é um múltiplo de r. Como habitualmente $E/\overline{K_0}[r]$ denota o grupo de torção de ordem r.

Escolhe-se K como a mínima extensão de K_0 que contém as coordenadas de todos os pontos neste grupo de torção. Abusando um pouco da notação podemos escrever $K = K_0(E/\overline{K_0}[r])$.

271 Noção (Emparelhamento de Weil)
O ***emparelhamento de Weil*** *é a função* $\mathbf{w}_r \colon E[r] \times E[r] \to \boldsymbol{\mu}_r$ *com* $\mathbf{w}_r(P,Q)$ *gerado por:*

1. *Escolhe-se dois divisores* $D \sim (P) - (\mathcal{O})$ *e* $D' \sim (Q) - (\mathcal{O})$ *de suporte disjuntos.*

   Basta escolher pontos R e S apropriados e fazer $D = (P \oplus R) - (R)$ e $D' = (Q \oplus S) - (S)$.

2. *Escolhe-se funções* $f, g \in K(E)$ *tais que* $(f) = r\,D$ *e* $(g) = r\,D'$.

   Porque P e Q pertencem a $E[r]$ tem-se $[r]P = [r]Q = \mathcal{O}$. Logo $r(P) - r(\mathcal{O}) = r\,D \sim 0$ e $r(Q) - r(\mathcal{O}) = r\,D' \sim 0$. Consequentemente, existem $f, g \in K(E)$ tais que $(f) = r\,D$ e $(g) = r\,D'$.

3. *Calcula-se* $w = f(D')/g(D)$ *e faz-se* $\mathbf{w}_r(P,Q) = w^l$.

   Pelo lema 271, tem-se $f(D' + (h')) \simeq_r f(D')$ e $g(D + (h)) \simeq_r g(D)$ para quaisquer $h, h' \in K(E)$. Portanto, o valor de $\mathbf{w}_r(P,Q)$ é independente dos divisores D, D' desde que satisfaçam as condições em (1).

Como sabemos, funções racionais distintas com o mesmo divisor são determindas a menos da multiplicação por uma constante; isso não afecta o resultado da sua aplicação a divisores de grau 0. Por isso o valor de $\mathbf{w}_r(P,Q)$ é independente da escolha específica de funções f, g em (2).

As propriedades da função $\mathbf{w}_r$ podem ser sumariadas no seguinte resultado.

272 Teorema (Emparelhamento de Weil)
O algoritmo apresentado na noção 274 define um emparelhamento. Adicionalmente verifica:

1. *(*Invariância de Galois*) Para todo automorfismo* $\sigma \in \mathbb{G}(\overline{K}/K)$ *verifica-se*

$$\mathbf{w}_r(\sigma(P), \sigma(Q)) \;=\; \sigma(\mathbf{w}_r(P,Q))$$

2. *(*Simetria*)* $\mathbf{w}_r(,P,P) = 1$ *e* $\mathbf{w}_r(,P,Q) = \mathbf{w}_r(,Q,P)^{-1}$.

Esboço de prova A prova de que a função é um emparelhamento usa as mesmas técnicas que a prova equivalente no emparelhamento de Tate. A invariância de Galois é também provada do mesmo mdodo. A simetria é óbvia a partir da definição.

Existem semelhanças óbvias entre os emparelhamentos de Tate e de Weil. Para pontos P, Q na intersecção dos respectivos domínios é óbvio, pela definição, que

$$\mathbf{w}_r(P,Q) \quad = \quad \frac{\mathbf{e}_r(P,Q)}{\mathbf{e}_r(Q,P)} \tag{175}$$

Existem, no entanto, alguns detalhes que podem impedir esta relação óbvia. Em primeiro lugar, as curvas E/K são definidas sobre corpos K distintos. Depois os próprios domínios das funções são subgrupos distintos da curva.

Em ambos os casos o corpo de base K_0 é $\mathbb{F}_q$ com característica p; em ambos os casos a ordem r é um primo distinto de p. Porém,

(i) no emparelhamento de Tate, escolhe-se primeiro o corpo K como $\mathbb{F}_q(\boldsymbol{\mu}_r)$ e escolhe-se depois a curva E/K de forma a ter uma ordem que seja múltipla de r.

(ii) no emparelhamento de Weil, escolhe-se primeiro uma curva $E/\overline{\mathbb{F}_q}$ que tenha uma ordem múltipla de r e fixa-se o corpo K como a extensão $\mathbb{F}_q(E[r])$.

Geralmente, o corpo $\mathbb{F}_q(E[r])$ seria muito maior do que o corpo $\mathbb{F}_q(\boldsymbol{\mu}_r)$. No entanto, para muitas situações com interesse criptográfico, os dois coincidem. Isto é resultado do seguinte teorema.

273 Teorema (Balasubramanian e Kobliz)
Seja $E/\mathbb{F}_q$ uma curva elíptica definida dobre um corpo $\mathbb{F}_q$ de característica p. Seja $r \neq p$ um primo que não divide $q-1$ mas divide a ordem $|E/\mathbb{F}_q|$ da curva. Então $E[r] \subset E/\mathbb{F}_{q^k}$ se e só se r divide $q^k - 1$.

Note-se que as condições do teorema exigem que, à partida, a curva sobre o corpo base já contenha pontos suficientes para conter $E[r]$; nomeadamente a sua ordem tem de ser um múltiplo de r. Esta condição é mais forte das que foram colocadas na definição de curva em ambos os emparelhamentos, onde se requeria que apenas a curva sobre a extensão k verificasse uma condição análoga.

Se estas condições forem verificadas o domínio de ambos emparelhamentos é compatível (tendo em atenção que, no emparelhamento de Tate, os dois argumentos têm domínios diferentes) e a igualdade (175) pode ser usada.

□

No cálculo do emparelhamento de Tate, o passo que exige maior esforço computacional é a determinação da função racional f tal que $(f) = r(P) - r(\mathcal{O})$. Dado que o emparelhamento de Weil este passo é repetido para a função g, é natural pensar-se que, com domínios compatíveis, o esforço computacional do emparelhamento de Weil seja, aproximadamente, o dobro do de Tate.

Normalmente r é um número primo com, pelo menos 160 bits de representação e, por isso, a exponenciação directa de uma função não é abordagem razoável. O **algoritmo de Miller** baseia-se na usual estratégia de cálculo eficiente de exponenciais por sucessivas do expoente e cálculo de quadrados.

Para este algoritmo recordemos o lema 258 (página 526), a função racional $\mu(P,Q)$ definida na noção 259 (página 527), e o seu divisor $(\mu(P,Q)) = (P) + (Q) - (P \oplus Q) - (\mathcal{O})$.

Para um ponto P da curva, seja $P_i = [i]P$ e considere-se a sucessão de funções racionais f_i tais que $f_1 = 1$ e

$$(P_i) - (\mathcal{O}) + (f_i) \;=\; i(P) - i(\mathcal{O}) \tag{176}$$

Note-se que, se $P \in E[r]$, $P_r = \mathcal{O}$ e que, portanto, f_r é a função que se pretende calcular. Por outro lado,

$$(\mu(P_i, P_j)) = (P_i) + (P_j) - (P_{i+j}) - (\mathcal{O}) \tag{177}$$

dado que $P_{i+j} = P_i \oplus P_j$.

274 LEMA *Com a sucessão de funções f_i verificando* (176) *verifica-se, a menos de uma constante multiplicativa,*

$$f_{i+j} \;=\; f_i \cdot f_j \cdot \mu(P_i, P_j) \tag{178}$$

Em particular

$$f_{2\,i} \;=\; f_i^2 \cdot \mu(P_i, P_i) \qquad \textit{e} \qquad f_{i+1} \;=\; f_i \cdot \mu(P_i, P) \tag{179}$$

Prova Dado que $\mu(P_i, P_j)$ verifica (177) tem-se $(f_i \cdot f_j \cdot \mu(P_i, P_j)) \;=\; (f_i) + (f_j) + (\mu(P_i, P_j)) \;=\; (f_{i+j})$. Como as funções racionais são determinadas, a menos de uma constante multiplicativa, pelos seus divisores, f_{i+j} é determinada por (178).

O algoritmo de Miller determina f_r usando sucessivamente as igualdades (179).

Objectivo Determinar $f_r(D)$ sendo $D = (Q \oplus S) - (S)$ para um S apropriado.

1. Fazer $f \leftarrow 1$, $T \leftarrow P$ e $n \leftarrow r$
2. Enquanto $n > 0$ executar repetidamente os seguintes passos
3. $T \leftarrow [2]T \quad ; \quad \lambda \leftarrow \mu(T,T) \quad ; \quad f \leftarrow f^2 \cdot \lambda(Q \oplus S)/\lambda(S)$
4. Se n é impar fazer $T \leftarrow T + P \quad ; \quad \lambda \leftarrow \mu(T,P) \quad ; \quad f \leftarrow f \cdot \lambda(Q \oplus S)/\lambda(S)$
5. $n \leftarrow n/2$.

O valor final de f contém $f_r(D)$; para calcular o emparelhamento de Tate basta agora determinar em K, o valor f^l usando um algoritmo de exponenciação eficiente.

8.7 Curvas Super-singulares e Implementação de Emparelhamentos

As condições para a definição do emparelhamento de Tate exigem uma escolha apropriada do corpo $K = \mathbb{F}_q(\boldsymbol{\mu}_r)$ impondo um garu de embebimento k pequeno, uma ordem r suficientemente grande e uma curva elíptica E/K cuja ordem seja um multiplo de r.

A verificação simultânea de todas estas condições é difícil se as curvas forem escolhidas arbitrariamente. Porém, para alguns tipos de curvas esta escolha é mais simples.

275 Noção
Seja $K = \mathbb{F}_q$ um corpo finito de característica p. Uma curva elíptica E/K é **super-singular** *quando satisfaz uma seguintes condições equivalentes:*

1. *Verifica-se $|E/K| = 1 \mod p$; equivalentemente $|E/K| = q + 1 - t$ para algum múltiplo t de p.*
2. *E/K não tem pontos de ordem p em $\overline{K}$; equivalentemente o grupo de torsão $E/\overline{K}[p]$ reduz-se a $\{\mathcal{O}\}$.*

Curvas que não verificam qualquer destas condições dizem-se **ordinárias**.

Para curvas arbitárias E/K (supersingulares ou ordinárias), o seguinte teorema ajuda a caracterizar o seu grau de embebimento.

276 Teorema (Waterhouse)
Nas condições da definição 278, para cada $a \in \mathbb{N}$, seja

$$T_a \;=\; \left\{\, t \in \mathbb{Z} \;\mid\; |E| = p^a + 1 - t \quad \textit{para alguma curva } E/K \,\right\} \tag{180}$$

Então, para todo $t \in T_a$, verifica-se $\gcd(t,p) \neq 1$, $|t| \leq 2\sqrt{p^q}$ e uma das seguintes condições

1. *a é par e $t = \pm 2\, p^{a/2}$*
2. *a é par, $(p \neq 1 \mod 3)$ e $t = \pm p^{a/2}$*
3. *a é impar, $p = 2, 3$ e $t = \pm p^{(a+1)/2}$*
4. *a é ímpar e $t = 0$*
5. *a é par, $(p \neq 1 \mod 4)$ e $t = 0$.*

Como consequência pode-se construir a seguinte tabela de curvas super-singulares para graus de embebimento $k = 1, 2, 3, 4, 6$. A tabela contém uma coluna q com o número de elementos de $K = \mathbb{F}_q$, a ordem $|E/K|$ da curva e a dimensão n tal que estrutura do grupo $E/\mathbb{F}_{q^k}$ é isomórfico com $\mathbb{Z}_n \times \mathbb{Z}_n$.

k	q	$\lvert E/K\rvert$	n
1	$p^{2\,b}$	$q \pm 2\sqrt{q} + 1$	$\sqrt{q} \pm 1$
2	$*$	$q + 1$	$q + 1$
3	$**$	$q + \sqrt{q} + 1$	$q^{3/2} - 1$
3	$**$	$q - \sqrt{q} + 1$	$q^{3/2} + 1$
4	$2^{2\,b+1}$	$q \pm \sqrt{2\,q} + 1$	$q^2 + 1$
6	$3^{2\,b+1}$	$q \pm \sqrt{3\,q} + 1$	$q^3 + 1$

O caso $(*)$ corresponde às entradas (4) e (5) no teorema 279. Os casos $(**)$ correspondem à entrada (2).