Ténicas Criptográficas

José Manuel E. Valença *

18 de Junho de 2007

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

1.Introdução

Criptografia é uma ciência[1] de gerar soluções tecnológicas para os problemas de **Segurança em Sistemas de Informação** [2].

Essas soluções são formadas, normalmente, por várias componentes; nomeadamente é possível identificar componentes que agem em diferentes níveis; por exemplo,

- ao nível das técnicas matemáticas que manipulam os diferentes items de informação; este é o nível das **técnicas criptográficas**,
- ao nível da concretização tecnológica das técnicas criptográficas em dispositivos, programas, etc; este é o nível dos **processos criptográficos**,
- ao nível das regras de conduta e boa utilização dos processos criptográficos; este é o nível que podemos designar por **procedimentos de segurança**.

Este curso vai-se concentrar na primeira destas componentes, reservando as duas restantes para outras disciplinas.

No contexto deste curso a noção de "segurança" é bastante restrita: os seus objectivos exprimem-se em **relações** estabelecidas entre **agentes** e **items de informação**. Essencialmente, essas relações assumem duas formas fundamentais:

- O **conhecimento** que um agente tem sobre a existência, natureza e conteúdo de cada item de informação, e
- A **confiança** que o agente tem nesse conhecimento.

[1] Ou "arte"!

[2] INFOSEC na terminologia dos sistemas estratégicos.

Todos os objectivos de segurança podem ser definidos como garantias sobre o controlo deste tipo de relações **na presença de um ambiente hostil**. Isto é, a segurança tem em conta não só as relações que os **agentes legítimo** têm com determinados items de informação mas também a presença de outros agentes, designados por **agentes hostis** ou **adversários**, que têm o objectivo de violar essas relações.

□

A relação de conhecimento conduz, normalmente, a objectivos de segurança ligados à **confidencialidade** ou **privacidade** da informação. De facto pode-se definir confidencialidade como "controlo do conhecimento" enquanto que privacidade pode ser definida como "controlo do controlo do conhecimento".

Exemplo 1 : O acesso a um determinado sistema de informação pode ser condicionado pelo conhecimento de um determinado **segredo**[3].

Um objectivo de segurança pode ser, por exemplo, garantir que apenas o agente legítimo conhece o conteúdo desse segredo; isto exclui do segredo qualquer agente hostil: a segurança será violada se algum adversário tiver acesso a essa informação.

Este é um exemplo típico de um objectivo de **confidencialidade**.

No entanto, mesmo sem conhecer o conteúdo do segredo, uma agente hostil pode saber classificar o conteúdo informativo do segredo; por exemplo, que tipo de dados é usado, quantos bits de representação exige, etc. O conhecimento da natureza do segredo pode também, em certas circunstâncias, constituir uma violação à confidencialidade.

Também o conhecimento, por um agente hostil, de existência do segredo e da sua funcionalidade (mesmo sem conhecer o seu conteúdo ou a sua natureza) pode constituir uma ameaça à segurança do sistema.

Ocultar a própria existência do segredo é um objectivo de **privacidade**.

□

[3] Que, tradicionalmente, tem nomes como "palavra chave", "código de acesso", etc.

A relação de confiança tem a haver com a **autenticidade** dos items de informação. Um comportamento hostil em relação à autenticidade é, normalmente, designado por **fraude**.

Exemplo 2 : Quando um cliente usa meios electrónicos para aceder a uma conta bancária é necessário que o banco reconheça a sua **identidade**. A informação que define essa identidade tem de ser autenticada; uma fraude é um uso indevido dessa conta por falta de autenticação do utilizador.

□

Quando um médico usa meios electrónicos para efectuar um acto médico (desde pedir um exame até propor um intervenção cirúrgica) é necessário não só autenticar a identidade do médico como obter meios de **prova** que atestem, no futuro, o acto. Portanto tem de existir um documento que explicite o evento identificando o médico, o acto e a data/hora do mesmo.

Um exemplo de um tal documento é uma simples receita médica. Esse documento tem de ser autenticado por todos os intervenientes no acto (pelo médico e por quem dá continuação ao acto).

□

Quando um agente da autoridade examina uma carta de condução, ele aceita como autêntica a informação nela expressa (designadamente, que o titular está habilitado a conduzir um automóvel) baseando-se em dois tipos de **crenças**: ele acredita que o documento tem por **autor** uma autoridade apropriada (neste caso a DGV) e que essa autoridade desenvolveu todos os mecanismos necessários à validação da informação (é fidedigna).

Surgem aqui duas outras noções associadas à confiança/autenticidade: a relação de *autoria* e a relação de *crença*.

A relação de confiança lida com este conjunto de conceitos: autenticidade, fraude, identidade, autoria, prova, crença, etc. Ao longo deste curso veremos como é que estes conceitos de interligam e quais as técnicas criptográficas que lhes estão associadas.

De entre estas noções duas tem um papel primordial e são essenciais a uma representação das noções de autenticidade

entidades que representam agentes; a identidade é, de certa forma, a crença numa determinada entidade.

provas que representam as crenças sobre uma determinada asserção, evento, entidade, etc.

A confiança normalmente exprime-se como um conjunto de crenças de um agente baseado em provas adequadas. Por isso pode-se ver a a confiança como uma aplicação que, a cada entidade, associa um conjunto de provas (as suas "crenças").

Outro modo de de representar essa aplicação será através de uma relação; assim, como primeira aproximação, pode-se escrever:

$$\text{Confiança} \subseteq \text{Entidade} \times \text{Prova}$$

□

1.1.Confiança, Acreditação, Autoridades e Verificação

Todo o utilizador da Criptografia é confrontado com a dúvida constante:

> *será que a técnica que eu estou a usar no meu sistema de informação (e do qual dependem bens, vidas, etc.) é mesmo de confiança? Será que não existe um ataque, que desconheço, e que compromete todo o meu sistema de informação?*

Esta mesma questão pode ser aplicada a qualquer asserção relevante à segurança do sistema de informação. Por exemplo a asserção: *o responsável pelo sistema é X* ou *o sistema tem uma capacidade de carga Y*.

A resposta a esta questão está essencialmente no chamado processo de **acreditação** (por vezes também referido por **certificação**).

No processo de acreditação/certificação uma ou mais entidades, designadas por **autoridades**, fornecem provas (designadas por **credenciais** ou **certificados**) onde se atesta a validade da asserção. O utilizador deve poder **verificar** as credenciais/certificados fornecidas pelas autoridades em que acredita.

Conhecimento/Segredos e Confiança/Autenticidade não são os únicos objectivos das técnicas criptográficas. Veremos, ao longo deste curso, objectivos específicos de cada uma das técnicas que iremos analisar.

Porém todos esses objectivos, mesmo que não sejam classificáveis totalmente em nenhum destes campos, acabam por ter as suas razões essenciais em ambos. Desta forma este curso parte da convicção que qualquer objectivo de segurança realizável por uma técnica criptográfica exprime-se completamente em termos destas duas componentes: conhecimento e confiança.

2.Computações

A confiança numa qualquer técnica criptográfica lida com a crença de que, em circunstâncias normais e sem ter acesso a segredos privilegiados, um atacante não dispõe de capacidade computacional suficiente para violar o objectivo de segurança que a técnica protege.

Nomeadamente na crença que as computações necessárias ao ataque exigem recursos computacionais que estão fora das capacidades "realistas" de qualquer intruso; i.e., computacionalmente o ataque é **intratável**.

Simultaneamente o uso legítimo de uma técnica criptográfica deve exigir poucos recursos computacionais de forma a que possa ser facilmente adoptada nos dispositivos comuns (computadores pessoais, tele-móveis, cartões inteligentes, etc.).

As boas técnicas criptográficas devem estabelecer este equilíbrio: computacionalmente devem ser muitos eficientes no uso legítimo e serem intratáveis para ataques.

Para isso é essencial à Criptografia caracterizar a **complexidade computacional** dos seus algoritmos e dos eventuais ataques. É através do estudo da complexidade que se pode avaliar tanto a eficiência como a intratabilidade.

2.1.Notação Assimptótica

Frequentemente lidamos com funções reais de argumento inteiro positivo $f, g, \ldots : \mathbb{N} \to \mathbb{R}$ [4] cujo comportamento não é conhecido com rigor mas que, ainda assim, pode-se ver contido dentro de determinados limites. Muitas vezes basta conhecer a "ordem de grandeza" dos resultados $f(n), g(n), \ldots$ quando o argumento n percorre todo o domínio $\mathbb{N}$.

Para clarificar esta noção intuitiva de "ordem de grandeza" existe uma notação, designada por **notação-O**, que permite formalizar esta noção através da comparação de funções. A notação oferece várias alternativas, ligeiramente diferentes entre si:

$f(n) = O(g(n))$ (lê-se *f é de ordem g*) quando existe uma constante $C > 0$ tal que, para todo n suficientemente grande,

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

$f(n) = O^{-1}(g(n))$ quando existe uma constante $C > 0$ tal que, para todo n suficientemente grande,

$$|f(n)| \cdot |g(n)| \geq C$$

é apenas uma forma alternativa de escrever $f(n)^{-1} = O(g(n))$

$f(n) = o(g(n))$ quando

$$\lim_{n\to\infty} f(n)/g(n) = 0$$

$f(n) = \Omega(g(n))$ *(definição de Knuth)* quando existe uma constante $c > 0$ tal que, para todo n suficientemente grande,

$$|f(n)| \geq c\,|g(n)|$$

[4] Nomeadamente, neste curso, são muito importantes as funções reais e positivas que caracterizam **probabilidades** de certos acontecimentos ocorrerem ou certas computações terem sucesso.

$f(n) = \Omega(g(n))$ *(definição de Hardy e Littlewood)* quando

$$f(n) \neq o(g(n))$$

$f(n) = \Theta(g(n))$ quando

$$f(n) = O(g(n)) \ \wedge \ f(n) = \Omega(g(n))$$

Notas

1. 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.[5] Por isso $f(n) = o(g(n))$ implica $f(n) = O(g(n))$ mas a implicação inversa não se verifica necessariamente.

2. Na literatura existem duas versões diferentes para a definição da comparação $\Omega(\cdot)$; apresentamos ambas: a primeira é a que é normalmente usada no estudo da Computação enquanto que a segunda é a mais usada em textos na área da Matemática.

   As duas definições não são equivalentes. De facto a segunda definição poderia ser reescrita como

   > *existe* $c > 0$ *tal que* $|f(n) \geq c\,|g(n)|$ *ocorre para um conjunto não limitado de valores de* n.

   Enquanto que a definição de Knuth exige que $|f(n) \geq c\,|g(n)|$ se verifique para todo n suficientemente grande, a definição de Hardy e Likttlewood é mais fraca e exige apenas que esta relação ocorra para um número não limitado de valores de n.

3. Usando a primeira definição de $\Omega(\cdot)$ a definição de $\Theta(.)$ equivale à afirmação 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

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

[5] $f(n) = c + o(1)$ é apenas um modo de escrever $\lim_{n\to\infty} f(n) = c$.

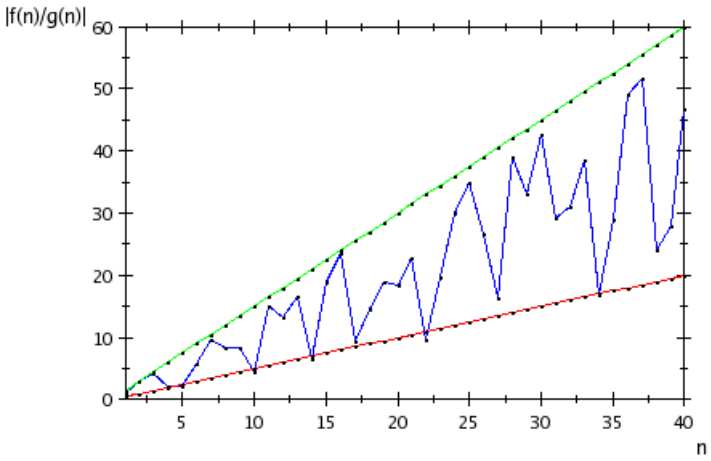
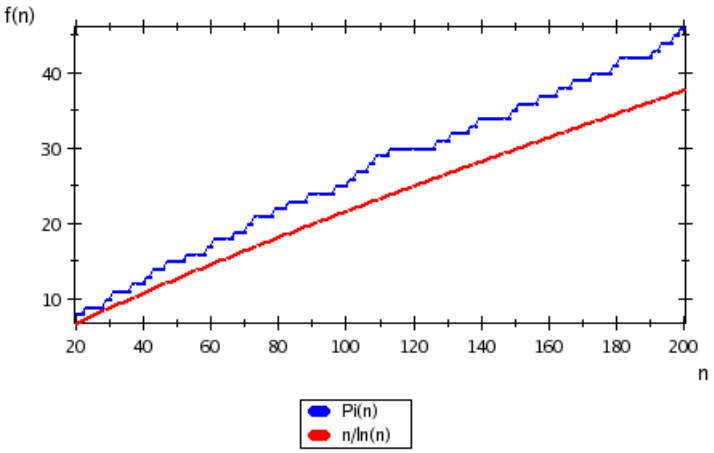


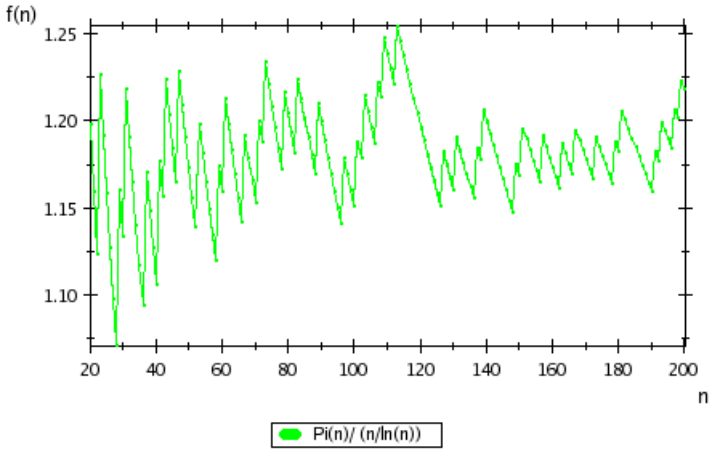
Figura 1: Exemplo: $f(n) = \Theta(n\, g(n))$.

A figura **??** representa um exemplo em que se tem $f(n) = \Theta(n\, g(n))$. Isto é equivalente aos limites $c\, n \leq |f(n)/g(n)| \leq C\, n$.

Exemplo 3 : 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 dois tipos de medida: a **complexidade** e a **probabilidade de sucesso**.

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*. [6]

A probabilidade de sucesso de um algoritmo é também uma medida que se pode caracterizar por uma notação de ordem. Normalmente a complexidade é caracterizada pela forma como as funções crescem enquanto que a probabilidade de sucesso é caracterizada por funções que decrescem

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

Polinomial

A ordem polinomial ocorre quando $\tau(n) = O(p(n))$, para algum polinómio positivo $p(n)$.

Como casos particulares temos as ordens **linear** ($\tau(n) = O(n)$), **quadrática** ($\tau(n) = O(n^2)$), **cúbica** ($\tau(n) = O(n^3)$), etc. . .
Ainda, como caso particular, temo a **ordem constante** em que $\tau(n) = O(1)$.

Polinomial inversa

Ocorre quando $\tau(n) = O^{-1}(p(n)) = O(1/p(n))$, para um polinómio positivo $p(n)$.

Dá origem às versões "inversas" das ordens anteriores: **inversa linear** ($\tau(n) = O(n^{-1})$), **inversa quadrática** ($\tau(n) = O(n^{-2})$), etc. . .

Exponencial

Quando $f(n)$ não é polinomial.

Desprezável

Quando $f(n)$ não é polinomial inversa.

Sub-exponencial

Quando $f(n) = O(e^{o(n)})$

A ordem sub-exponencial é algo intermédio entre a ordem polinomial e a exponencial.

[6] Em alternativa, a complexidade pode contar o número *máximo* ou o número *mínimo* de *slots* temporais ou unidades de memória.

É frequente que a ordem sub-exponencial se possa escrever numa notação específica; para tal define-se

$$L_n[p,c] \;=\; O(2^{c\,n^p\,(\log_2 n)^{1-p}})$$

para $p \in [0,1]$ e $c > 0$.

Por exemplo, a ordem $L_n[0,c]$ é fácil verificar que é equivalente a $O(n^c)$: a ordem polinomial. Por outro lado a ordem $L_n[1,c]$ é equivalente a uma ordem completamente exponencial.

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

Exemplo 4 : 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**.

Uma função do tipo $\delta(n) = 2^{-q(n)}$ (sendo $q(n)$ p.p. de grau > 0) é um exemplo paradigmático de função desprezável. O mesmo se pode dizer da função $\delta(n) = 1/n!$. Em qualquer dos casos verifica-se que, para toda p.p. $p(n)$, a função $\delta(n) \cdot p(n)$ tem um máximo.

Como exemplo, a figura seguinte ilustra o comportamento de $n^{100}/n!$ em escala logarítmica. Vemos que, 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.

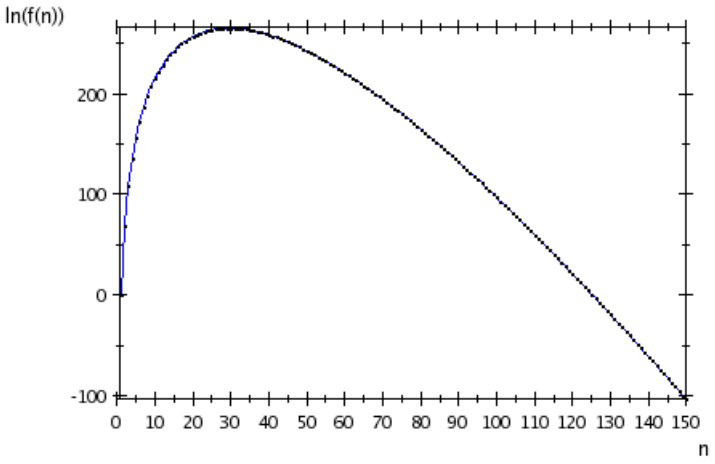


Figura 2: Gráfico de $\frac{n^{100}}{n!}$ em escala logarítmica.

É frequente comparar funções positivas limitadas (probabilidades, por exemplo) através das suas ordens de grandeza.

DEFINIÇÃO 1 *Duas funções* $\delta, \tau : \mathbb{N} \longrightarrow \mathbb{R}_+$ *são* **assimptoticamente similares**, *e escreve-se* $\delta \sim \tau$, *quando* $|\delta(n) - \tau(n)|$ *é desprezável.*

Se $\delta(n)$ e $\tau(n)$ designarem as probabilidades de dois eventos ocorrerem, então verificando-se $\delta \sim \tau$ isso significa que os eventos (sendo, em princípio, distintos) têm uma probabilidade de ocorrer que, consoante n cresce, se vai tornando indistinguível.

2.2.Strings finitas e infinitas de bits

Na sua forma mais simples, a complexidade computacional define-se para funções que transformam *strings* de bits em *strings* de bits. Representamos por $\mathbb{B}, \mathbb{B}^n, \mathbb{B}^*, \mathbb{B}^\infty$, respectivamente, o espaço de bits $\{0,1\}$, o espaço das palavras com exactamente n bits, o espaço de *strings* finitas de bits, o espaço de todas as *strings* infinitas de bits, e o espaço de todas as *strings* finitas ou infinitas.

$$\mathbb{B}^* \;=\; \bigcup_{n=0}^{\infty} \mathbb{B}^n$$

Strings (finitas ou infinitas) de bits codificam informação sob várias formas. Nomeadamente

- $\mathbb{B}^*$ codifica qualquer conjunto **enumerável** (ou **contável**)

  Uma representação normal de $\mathbb{B}^*$ é o conjunto dos números naturais $\mathbb{N}$ que pode ser definida pela função $\mathsf{nat} : \mathbb{B}^* \longrightarrow \mathbb{N}$ dada por

  $$\mathsf{nat} : \langle\,\rangle \;\mapsto\; 0 \qquad , \qquad \mathsf{nat} : \langle b|\omega\rangle \;\mapsto\; b + 2 \cdot \mathsf{nat}(\omega) \tag{1}$$

  Qualquer outro conjunto contável (por exemplo, os racionais $\mathbb{Q}$) são isomórficos com $\mathbb{N}$ e, por esta representação, são isomórficos com $\mathbb{B}^*$.

- $\mathbb{B}^\infty$ codifica os conjuntos **compactos não-enumeráveis**.

  Note-se, em primeiro lugar, que cada $\omega \in \mathbb{B}^\infty$ deve ser visto como uma função booleana $\omega : \mathbb{N} \longrightarrow \mathbb{B}$ que associa cada eventual índice $i \in \mathbb{N}$ um valor em $\{0,1\}$ consoante o *bit* de ordem i é 0 ou é 1.

  Por isso pode-se ver $\mathbb{B}^\infty$ como o conjunto de todas as funções $\mathbb{N} \longrightarrow \mathbb{B}$ ou, equivalentemente, como o conjunto $\mathbf{2}^{\mathbb{N}}$ de todos os conjuntos de números naturais.

  $\mathbb{B}^\infty$ codifica o intervalo real $[0,1] \subset \mathbb{R}$ (que é o paradigma dos compactos não-enumeráveis) através da função $\mathsf{cod} : \mathbb{B}^\infty \longrightarrow \mathbb{R}$ definida por

  $$\mathsf{cod}(\omega) \;\doteq\; \sum_{i=0}^{\infty} \omega_i \, 2^{-i-1} \tag{2}$$

As computações (algoritmos) que vamos usar existem num domínio de probabilidades de tal forma que

Cada computação φ é exclusivamente caracterizada por:

(i) uma **medida de probabilidade** $\{\varphi\}$ de ter sucesso ou falhar.

(ii) uma **medida de complexidade média** $[\![\varphi]\!]$ caso tenha sucesso.

Nesta perspectiva temos de caracterizar:

- Uma **linguagem de algoritmos** que, semanticamente, são caracterizados apenas pelo seu sucesso e pelo esforço em alcançar esse sucesso.
- Um **espaço de probabilidades** que vai servir de base à caracterização dessa semântica.
- Uma **medida de complexidade** que abstrai dos detalhes específicos da máquina que implemente os algoritmos.

2.3.Domínio de Probabilidades

Vamos supor que cada algoritmo têm acesso a uma **fonte de aleatoriedade** determinada por uma *string* infinita de bits $\Omega \in \mathbb{B}^\infty$. Isto significa que cada algoritmo por percorrer Ω sequencialmente recolhendo tantos bits aleatórios quantos precisar. Portanto

O **espaço de possibilidades** que iremos adoptar é o conjunto $\mathbb{B}^\infty$.

Notas

Sob estas *strings* são necessárias algumas operações que convém caracterizar.

Para cada bit $b \in \{0,1\}$ representa-se por b^k a *string* finita que se obtém repetindo k vezes o bit b.

Dada uma string finita $y \in \mathbb{B}^*$ e uma string eventualmente infinita $x \in \mathbb{B}^* \cup \mathbb{B}^\infty$ representa-se por $|x|$ e $|y|$ os respectivos **comprimentos**. Se x for infinita, escreve-se $|x| = \infty$.

Diz-se que y é **prefixo** de x, e escreve-se $y \leq x$, quando existe outra *string* u tal que $x = y\,u$.

Para $x \in \mathbb{B}^\infty$ chama-se **truncatura** de x a n *bits*, e representa-se por $x \rfloor n$, ao maior prefixo de x cujo comprimento é n.

Pode-se generalizar a noção de truncatura em duas direcções:

- Pode-se ver $x \rfloor n$ como um $y \in \mathbb{B}^n$ tal que $y \leq x$; isto sugere que qualquer $\sigma \subseteq \mathbb{B}^*$ pode tomar o lugar de $\mathbb{B}^n$; assim define-se

$$x \rfloor \sigma \doteq \{\, y \in \sigma \mid y \leq x \,\} \tag{3}$$

- Em vez de uma única *string* $x \in \mathbb{B}^\infty$ pode-se considerar um sub-conjunto $\alpha \subseteq \mathbb{B}^\infty$ e estender a truncatura a todos os elementos de α .

$$\alpha \rfloor \sigma \doteq \{\, y \in \sigma \mid \exists x \in \alpha \,.\, y \leq x \,\} \tag{4}$$

Em $\mathbb{B}^*$ a relação de prefixo $\leq$ é uma ordem parcial; ou seja, é uma relação *reflexiva* ($x \leq x$), *transitiva* ($x \leq y \wedge y \leq x \Rightarrow x \leq z$) e *anti-simétrica* ($x \leq y \wedge y \leq x \Rightarrow x = y$).

Para qualquer *string* finita $u \in \mathbb{B}^*$, **conjunto superior** com **pega** u, representado por $\uparrow u$, é o conjunto de todas as *strings* que têm u como prefixo

$$\uparrow u \doteq \{\, \omega \in \mathbb{B}^\infty \mid u \leq \omega \,\} \tag{5}$$

Genericamente, se $U \subseteq \mathbb{B}^*$, for um conjunto *livre de prefixos* (isto é, nenhum elemento de U é prefixo de outro elemento de U), então

$$\uparrow U \doteq \bigcup_{u \in U} \uparrow u \quad = \{\, \omega \in \mathbb{B}^\infty \mid \exists u \in U.\, u \leq \omega \,\} \tag{6}$$

Nestas circunstancias, diz-se que U é **base de prefixos** (ou, simplesmente, **base**) de $\uparrow U$.

Os **eventos** são determinados conjuntos de possibilidades. Assim cada evento será um determinado sub-conjunto $\alpha \subseteq \mathbb{B}^\infty$.

Neste estudo todo o conjunto superior $\uparrow u$, com $u \in \mathbb{B}^*$, será um evento.

Note-se que isto inclui o universo de possibilidade $\mathbb{B}^\infty$ como evento já que coincide com o conjunto superior da *string* nula; isto é, $\uparrow \varepsilon = \mathbb{B}^\infty$.

Os **espaços de eventos** são conjuntos de eventos que contém o universo de possibilidades (designado por **evento certo**) e são fechados por complementos e uniões contáveis. [7]

Genericamente, um conjunto $\Sigma \subseteq \wp(X)$ de sub-conjuntos de um conjunto X que contém o conjunto vazio $\emptyset$ e é fechado por complementos e uniões contáveis, designa-se por σ-**álgebras** de X.

[7] Sendo fechados por complementos e uniões contáveis são, obviamente, também fechados por intersecções contáveis e contém o conjunto vazio (o evento **impossível**).

O espaço de eventos é uma das as seguintes σ-álgebras de $\mathbb{B}^\infty$

- $\mathcal{L}$ – a menor σ-álgebra que contém todos os eventos $\uparrow u$ com $u \in \mathbb{B}^*$
- $\mathcal{L}_n \subseteq \mathcal{L}$ – a menor σ-álgebra que contém todos os eventos $\uparrow u$ com $u \in \mathbb{B}^n$

Notas

Existem diferenças importantes entre estes dois espaços de eventos:

1. Em $\mathcal{L}_n$ todos os eventos se podem escrever como uma união finita de eventos-base $\uparrow u$ em que todos as pegas u têm o mesmo comprimento n. Nomeadamente, é fácil provar que o complemento ou intersecção de uniões finitas $\bigcup_{u \in U} (\uparrow u)$ têm precisamente a mesma forma.

   Vendo, por exemplo, $\uparrow u \in \mathcal{L}_n$ como a caracterização probabilística do sucesso de um algoritmo φ diríamos que esse sucesso depende apenas dos primeiros n bits da fonte de aleatoriedade; o que se segue a seguir aos n bits é irrelevante.
2. Em $\mathcal{L}$ as pegas u dos eventos base $\uparrow u$ têm qualquer comprimento; como consequência já estas propriedades de finitude não se verificam. Nomeadamente o complemento de um evento básico $\uparrow u$ não se pode escrever como uma união de eventos dessa forma mas sim como a intersecção contável (não finita) de eventos base.
   Em contrapartida $\mathcal{L}$ contém muito mais eventos do que qualquer dos $\mathcal{L}_n$; nomeadamente contém eventos $\{\omega\}$ formados por uma única *string* infinita ω. De facto pode-se escrever
   $$\{\omega\} = \bigcap_{u \leq \omega} \uparrow u$$
   que é uma intersecção contável de eventos base em que se toma, como pegas, todos os prefixos finitos de ω.

   Se virmos um evento $\alpha \in \mathcal{L}$ como a caracterização da probabilidade de uma certa computação φ, então não existe garantias que seja possível decidir, olhando para um conjunto finito de bits da fonte de aleatoriedade, se φ tem sucesso ou não.

Dado que um evento em $\mathcal{L}_n$ tem a vantagem de ter uma caracterização "computável" (já que é determinado por bits em número finito e por uniões

finitas de eventos base) enquanto que os eventos em $\mathcal{L}$ são mais expressivos mas menos "computáveis", convém definir uma forma de "aproximação" dos segundos pelos primeiros.

Assim, para cada $\alpha \in \mathcal{L}$ e cada $n \geq 0$, define-se

$$\lceil \alpha \rceil_n \doteq \bigcap_{\sigma \in \mathcal{L}_n \setminus \emptyset} \{ \sigma \mid \sigma \supseteq \alpha \} \tag{7}$$

e designa-se por **fecho** de α em $\mathcal{L}_n$

Exemplo 5 : Considere-se o evento $\{\omega\}$ formada por uma única string $\omega \in \mathbb{B}^\infty$; representando por $u_n = \omega \rfloor n$ a truncatura de ω a n bits, então facilmente se verifica que

$$\lceil \{\omega\} \rceil_n = \uparrow(u_n)$$

Nota

Recorde-se que, genericamente, uma **medida de probabilidade** numa σ-álgebra Σ, sobre um conjunto de possibilidade X, é uma função real $\mu : \Sigma \longrightarrow [0,1]$ que verifica:

(i) $\mu(\mathbb{B}^\infty) = 1$.

(ii) Se α e β são eventos disjuntos então $\mu(\alpha \cup \beta) = \mu(\alpha) + \mu(\beta)$.

(iii) Se $\{\alpha\}_n$ for uma cadeia descendente de eventos ($m \geq n \Rightarrow \alpha_m \subseteq \alpha_n$), então

$$\mu(\bigcap_n \alpha_n) = \lim_n \searrow \mu(\alpha_n)$$

(iv) Se $\{\alpha\}_n$ for uma cadeia ascendente de eventos ($m \geq n \Rightarrow \alpha_m \supseteq \alpha_n$), então

$$\mu(\bigcup_n \alpha_n) = \lim_n \nearrow \mu(\alpha_n)$$

O triplo $\langle X, \Sigma, \mu \rangle$ designa-se por **espaço de probabilidades**.

Neste estudo usaremos uma probabilidade particular definida por

DEFINIÇÃO 2 *A* ***probabilidade de Lebesgue*** *é a medida de probabilidade* $\mu : \mathcal{L} \longrightarrow [0,1]$ *que, para todo* $u \in \mathbb{B}^*$ *, verifica:*

$$\mu(\uparrow u) \;=\; 2^{-|u|}$$

$\mathfrak{B}$ *designa o espaço de probabilidades* $\langle \mathbb{B}^\infty, \mathcal{L}, \mu \rangle$ *e* $\mathfrak{B}_n$ *designa o espaço de probabilidades* $\langle \mathbb{B}^\infty, \mathcal{L}_n, \mu \rangle$ *.*

Notas

Em termos de espaços de probabilidades, neste estudo usaremos sempre $\mathbb{B}^\infty$ como conjunto de possibilidades. A medida de probabilidade também é sempre a mesma: a probabilidade de Lebesgue. O que pode variar é a σ-álgebra: pode-se usar $\mathcal{L}$ ou então uma das suas sub-álgebras $\mathcal{L}_n$.

Estas σ-álgebras têm eventos de probabilidade nula distintos:

1. Em $\mathcal{L}_n$ apenas o evento impossível $\emptyset$ tem probabilidade 0; todos os restantes eventos têm probabilidade > 0.
2. Em $\mathcal{L}$ existe uma infinidade de eventos distintos do evento impossível que têm probabilidade 0.
   Por exemplo, o evento singular $\{\omega\} = \bigcap_{u \leq \omega} \uparrow u$ tem probabilidade 0

   $$\lim_{n \to \infty} \searrow \{\, \mu(\uparrow u) \mid u \leq \omega \wedge |u| = n \,\} \;=\; \lim_{n \to \infty} 2^{-n} \;=\; 0$$

   assim como têm probabilidade zero todos os eventos contáveis; de facto a sua probabilidade será a soma contável da probabilidade de eventos singulares; como todas as parcelas são nulas, a soma será nula.

□

Como temos eventos importantes (os eventos singulares ou contáveis) cuja probabilidade é uniformemente zero, torna-se necessário definir algum outro conceito que consiga captura a caracterização probabilística desses eventos.

Para isso define-se, para cada $\alpha \in \mathcal{L}$ e cada $n \geq 0$,

$$\mathbb{E}^{\mu}[\alpha](n) \doteq \mu(\lceil \alpha \rceil_n) \tag{8}$$

e designa-se por **probabilidade expectável** (ou, simplesmente, **expectativa**) de α em $\mathcal{L}_n$.

Se μ estiver implícito representa-se a expectativa simplesmente por $\mathbb{E}[\alpha](n)$

Note-se que $\mathbb{E}^{\mu}[\alpha] : \mathbb{N} \longrightarrow \mathbb{R}_+$ pode ser vista como uma função real, positiva e decrescente que, no limite, tende para $\mu(\alpha)$.

$$\mu(\alpha) = \lim_{n} \searrow \mathbb{E}^{\mu}[\alpha](n)$$

Exemplo 6 : O evento singular $\{\omega\}$ é caracterizado pela expectativa

$$\mathbb{E}[\{\omega\}](n) = 2^{-n}$$

Com esta noção de expectativa já é possível definir probabilidade condicional mesmo para este tipo de eventos de probabilidade nula.

Recordemos que a teoria elementar das probabilidades define a probabilidade condicional do evento A condicionada ao evento B como

$$P(A|B) \doteq \frac{P(A \cap B)}{P(B)}$$

Quando $P(B) = 0$ a definição não tem sentido.

Infelizmente no espaço de possibilidades $\mathbb{B}^{\infty}$ esta definição é insuficiente porque é necessário considerar frequentemente probabilidades condicionais em que o evento condicionante tem medida de probabilidade nula.

Por exemplo:

1. Considere-se variáveis X, Y reais e aleatórias uniformemente distribuídas no intervalo $[0,1]$ (que, como sabemos, é equivalente ao espaço de possibilidades $\mathbb{B}^\infty$). Vamos supor que as variáveis estão relacionadas pela equação

$$X + Y = 1$$

Em termos de probabilidades clássicas teríamos de afirmar que a probabilidade de ocorrência do evento $(X = 0)$ ou do evento $(Y = 1)$ deve ser nula em ambos os casos: $P(X = 0) = P(Y = 1) = 0$. No entanto parece intuitivo dizer que a probabilidade de ocorrer $(Y = 1)$, condicionada à ocorrência de $(X = 0)$, deve ser 1.
Por isso deve existir uma extensão da noção de probabilidade condicional que capture esta intuição e que resulte em algo da forma

$$P(Y = 1|X = 0) = 1$$

2. Em $\mathbb{B}^\infty$ são também importantes os eventos contáveis (nomeadamente os eventos singulares) que têm todos probabilidade nula. de facto faz todo o sentido colocar questões do tipo: condicionado a uma fonte de aleatoriedade que tem um conjunto de possibilidades contáveis, qual é a probabilidade de sucesso de uma determinada computação φ ?.
A noção clássica de probabilidade condicional não consegue dar-lhe resposta porque, como dissemos, os eventos contáveis têm probabilidade nula.

DEFINIÇÃO 3 *Dados* $\alpha, \beta \in \mathcal{L}$ $(\beta \neq \emptyset)$ *e* $n \geq 0$, *define-se*

$$\mathbb{E}^\mu[\alpha|\beta](n) \doteq \frac{\mu(\lceil\alpha\rceil_n \cap \lceil\beta\rceil_n)}{\mu(\lceil\beta\rceil_n)} \tag{9}$$

e designa-se por **expectativa condicional** *de* α *a* β *em* $\mathcal{L}_n$.

Se μ *estiver implícito,* $\mathbb{E}^\mu[\alpha|\beta](n)$ *escreve-se simplesmente* $\mathbb{E}[\alpha|\beta](n)$.

Se $\mathbb{E}^\mu[\alpha|\beta] \sim \mathbb{E}^\mu[\alpha]$ *então* α, β *dizem-se* ***expectavelmente independentes***.

Exemplo 7 : Considere-se o evento singular $\alpha = \{1^\infty\}$ determinado pela *string* que é uma sequência infinita de bits 1; considere-se também a o evento $\beta = \{0^\infty, 1^\infty\}$ formado por duas strings infinitas: uma só com bits 0 e outra só com bits 1.

Ambos os eventos têm probabilidade nula e por isso, em termos de probabilidade condicional clássica, não faz qualquer sentido $P(\alpha|\beta)$.

Porém a expectativa condicional $\mathbb{E}[\alpha|\beta](n)$ é bem definida. De facto, em $\mathcal{L}_n$ com $n > 0$, o menor σ que contém β é o evento

$$\sigma_n = \uparrow(0^n) \cup \uparrow(1^n)$$

cuja probabilidade é $\mu(\sigma_n) = 2^{-n} + 2^{-n} = 2^{-n+1}$.

Temos também $\alpha \cap \sigma_n = \{1^\infty\} \cap (\uparrow 0^n \cup \uparrow 1^n) = \uparrow 1^n$; a sua expectativa será $\mathbb{E}[\alpha \cap \sigma_n] = \mu(\uparrow 1^n) = 2^{-n}$. Portanto, para $n > 0$,

$$\mathbb{E}[\alpha|\beta](n) = \frac{\mathbb{E}[\alpha \cap \sigma_n]}{\mu(\sigma_n)} = \frac{2^{-n}}{2^{-n+1}} = 1/2$$

Exemplo 8 :
Supúnhamos que tanto α como β são conjuntos singulares: $\alpha = \{\omega\}$ e $\beta = \{\omega'\}$. Qual é a expectativa condicional $\mathbb{E}[\alpha|\beta](n)$?

Para um qualquer n, fazendo $\sigma \doteq \uparrow(\omega \rfloor n)$ e $\sigma' \doteq \uparrow(\omega' \rfloor n)$, obtém-se

$$\mathbb{E}[\alpha|\beta](n) = \mu(\sigma \cap \sigma')/\mu(\sigma')$$

Tem-se $\mu(\sigma) = \mu(\sigma') = 2^{-n}$; para determinar $\mu(\sigma \cap \sigma')$ são possíveis duas situações:

- se for $(\omega \rfloor n) = (\omega' \rfloor n)$, então $\sigma \cap \sigma' = \sigma$, o que implica $\mu(\sigma \cap \sigma') = 2^{-n}$.
- se for $(\omega \rfloor n) \neq (\omega' \rfloor n)$, então $\sigma \cap \sigma' = \emptyset$, o que implica $\mu(\sigma \cap \sigma') = 0$.

Logo,

$$\mathbb{E}[\alpha|\beta](n) = \begin{cases} 1 & \text{se } (\omega \rfloor n) = (\omega' \rfloor n) \\ 0 & \text{se } (\omega \rfloor n) \neq (\omega' \rfloor n) \end{cases}$$

Este exemplo dá uma visão interessante sobre a noção de expectativa condicional: vendo sucessivamente cada vez mais *bits*, enquanto não for possível distinguir as duas *strings* a expectativa é 1; logo que se tem a certeza que as *strings* são diferentes a expectativa passa a ser 0.

As noções de expectativa e expectativa condicional generaliza-se para *variáveis aleatórias*, nomeadamente as variáveis aleatórias inteiras que são indispensáveis ao estudo da complexidade dos algoritmos.

DEFINIÇÃO 4 *Uma função* $\xi : \mathbb{B}^\infty \to \mathbb{N}$ *é uma* **variável aleatória $\mathcal{L}$-expectável** *se, para cada* $k \geq 0$, *o conjunto de possibilidades* $\xi^{-1}(k)$ *é um evento em* $\mathcal{L}$.

Nesse caso define-se a função real positiva $\mathbb{E}[\xi] : \mathbb{N} \to \mathbb{R} \cup \{\infty\}$ *por*

$$\mathbb{E}[\xi](n) = \sum_{k \neq 0} k \cdot \mathbb{E}[\xi^{-1}(k)](n) \qquad \forall\, n \geq 0$$

que se designa por **expectativa** *de* ξ. *De forma análoga,*

Para qualquer evento $\beta \in \mathcal{L}$ *a* **expectativa condicional** *de* ξ *condicionado a* β *é a função* $\mathbb{E}[\xi|\beta] : \mathbb{N} \to \mathbb{R} \cup \{\infty\}$ *dada por*

$$\mathbb{E}[\xi|\beta](n) = \sum_{k \neq 0} k \cdot \mathbb{E}[\xi^{-1}(k)|\beta](n) \qquad \forall\, n \geq 0$$

Nota

De uma forma "grosseira" pode-se afirmar que a expectativa de ξ é a média dos diferentes valores possíveis da variável em que cada valor está pesado com a expectativa da variável ter, de facto, esse valor. O mesmo se pode dizer da expectativa condicional.

Este tipo de variáveis é usado no nosso estudo para medir a complexidade dos algoritmos e contam as unidades de tempo (*slots*) que, para cada valor possível da fonte de aleatoriedade, o algoritmo demora a terminar. Nestas circunstâncias a expectativa mede a complexidade média do algoritmo.

2.4.Funções e Probabilidade

O modelo probabilístico estende-se naturalmente a funções matemáticas que mapeiam *strings* em *strings*.

Neste estudo vamos considerar, principalmente, funções cujo domínio são as *strings* infinitas de bits $(\mathbb{B}^\infty)$ [8] e cujo contradomínio ou são as *strings* finitas $\mathbb{B}^*$ (equivalentemente, os números naturais $\mathbb{N}$, os racionais $\mathbb{Q}$, etc.), como paradigma de um contradomínio enumerável, ou então é o próprio $\mathbb{B}^\infty$ (equivalentemente, um intervalo real fechado, $[0,1]$, $[0,\infty]$, etc.) como paradigma de um contradomínio não enumerável.

As primeiras iremos designar (um pouco impropriamente!) por **funções enumeráveis** e as segundas por **funções não-enumeráveis**.

DEFINIÇÃO 5 *Seja X um qualquer domínio contável. A função enumerável $f : \mathbb{B}^\infty \to X$ é $\mathfrak{B}$-**mensurável** se $f^{-1}(x) \in \mathcal{L}$, para todo $x \in X$.*

*A relação $R \subseteq \mathbb{B}^\infty \times X$ é $\mathfrak{B}$-**mensurável** se, para todo $x \in X$,*

$$R^{-1}(x) \doteq \{\, w \in \mathbb{B}^\infty \mid (w,x) \in R \,\} \tag{10}$$

é um evento em $\mathcal{L}$.

*Se todos $f^{-1}(x)$ (ou, $R^{-1}(x)$) forem não-nulos, a função f (ou, relação R) diz-se **fortemente mensurável**.*

*A função não-enumerável $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$ é $\mathfrak{B}$-**mensurável** se, para todo o evento $\alpha \in \mathcal{L}$, se verifica $h^{-1}(\alpha) \in \mathcal{L}$.*

*A função h é **fortemente mensurável** se, para todo α não-nulo, $h^{-1}(\alpha)$ é não-nulo.*

[8] Como sempre, $\mathbb{B}^\infty$ é visto no contexto de um dos espaços $\mathfrak{B}$ ou $\mathfrak{B}_n$.

Notas

Porque qualquer σ-álgebra é uma topologia, a definição de função não-enumerável mensurável é equivalente, simplesmente, à afirmação de que h é uma função contínua na topologia $\mathcal{L}$.

Para funções enumeráveis pode ser feita uma observação análoga. De facto, equipando X com a σ-álgebra discreta (formada por todos os sub-conjuntos de X), afirmar que $f : \mathbb{B}^\infty \to X$ é mensurável é equivalente à afirmação de que essa função é contínua.

Pode-se interpretar uma função enumerável $f : \mathbb{B}^\infty \to \mathbb{N}$ como uma variável aleatória inteira positiva.

Exemplo 9 : Um exemplo de uma tal variável é o tempo de execução T_φ de uma algoritmo probabilístico φ : para cada possibilidade ω o tempo de execução $T_\varphi(\omega)$ conta o número de *slots* temporais que o algoritmo ocupa.

Só admitindo que T_φ é mensurável, será possível estudar probabilisticamente o tempo de execução. De facto com a hipótese de T_φ ser mensurável (e só com ela) é possível fixar um tempo qualquer t e caracterizar a probabilidade de φ terminar em exactamente t unidades temporais.

Exemplo 10 : Em Criptografia ocorrem, naturalmente, não funções de *strings* infinitas em *strings* finitas mas funções onde o domínio e o contradomínio têm outro tipo de limites. Por exemplo, funções $F : \mathbb{B}^* \to X$ (com X um de $\mathbb{B}^\infty, \mathbb{B}^*, \mathbb{B}^t$) cujo domínio são *strings* finitas são muito importantes. No entanto qualquer função desta forma é sempre representável por uma função mensurável $f : \mathbb{B}^\infty \to X$ da forma seguinte.

Comecemos pela codificação se *strings* finitas em *strings* infinitas. Tome-se uma qualquer $u \in \mathbb{B}^*$ e construa-se, em primeiro lugar, uma sua codificação finita $\tilde{u}$ definida por

$$\tilde{u} \doteq 0^{|u|}\, 1\, u \tag{11}$$

Qualquer z infinito que tenha $\tilde{u}$ como prefixo ($z \geq \tilde{u}$) é uma codificação de u.

Essencialmente a codificação constrói uma sequência $\tilde{u}$ formada inicialmente por tantos 0's quanto o comprimento de u, seguido de um *bit* 1 (para indicar o fim da sequência), junta-lhe finalmente u e continua-a de uma forma arbitrária.

A descodificação da *string* $z \geq \tilde{u}$ começa por contar em z o número de 0's até aparecer o primeiro 1; isso dá-lhe o comprimento da "informação útil" u; lê em seguida tantos *bits* quantos o número de 0's e reconstrói u; o resto de z é desprezado.

Apesar de cada *string* finita ser codificado por infinitas *strings* infinitas (todas as que pertencem a $\uparrow\tilde{u}$), para cada *string* infinita z existe uma única *string* finita u da qual z é uma codificação. Por isso é bem definida a função mensurável $f : \mathbb{B}^\infty \to X$ tal que

$$f(z) \doteq F(u) \qquad \text{com } u \text{ tal que } z \geq \tilde{u} \tag{12}$$

Note-se que esta definição implica que, para cada $\alpha \subseteq X$, se tenha

$$f(z) \in \alpha \quad \Leftrightarrow \quad \exists u.\ z \geq \tilde{u} \ \wedge\ F(u) \in \alpha$$

o que é equivalente a afirmar

$$f^{-1}(\alpha) \ = \ \bigcup \{ \uparrow\tilde{u} \mid u \in F^{-1}(\alpha) \} \tag{13}$$

Dado que $F^{-1}(\alpha)$ está contido em $\mathbb{B}^*$, é sempre enumerável (mesmo que α não o seja); portanto $f^{-1}(\alpha)$ é uma união contável de conjuntos superiores e, por isso, é sempre mensurável.

2.5.Linguagem de Decisões

Sem entrar em pormenores, vamos considerar uma linguagem para exprimir computações orientadas à noção de estado, que têm um comportamento aleatório, comportamento esse que se descreve simplesmente em termos da sua probabilidade de terminarem ou não com sucesso. Note-se que cada computação φ pode terminar com sucesso, terminar com falha ou, simplesmente, nunca terminar.

As componentes da linguagem são:

1. Uma **decisão unitária**, representada por $\mathbf{1}$.

   Este é um algoritmo particular que termina sempre com sucesso num único passo de execução e que não provoca qualquer alteração do estado (veremos adiante o que isto significa).

2. Um conjunto enumerável $\mathcal{A}$ de **decisões atómicas**.

   As decisões atómicas são algoritmos que, num determinado nível de abstracção, não são decomponíveis noutros algoritmos. A linguagem é parametrizada em relação a este conjunto de decisões.
   Por convenção assume-se que as decisões atómicas terminam sempre (com sucesso ou com falha) num único passo de execução.

3. A conectiva unária **negação** ($\neg\varphi$ ou $\bar{\varphi}$) e as conectivas binárias **composição** ($\phi\,\varphi$) e **escolha** ($\phi\|\varphi$).

   A negação é involutiva: i.e., $\neg\neg\varphi = \varphi$. A computação $\neg\varphi$ termina com sucesso quando a computação φ termina com falha e no mesmo número de passos.

   A decisão $\phi\,\varphi$ é a computação que é iniciada com a decisão ϕ que, se tiver sucesso, é continuada com a decisão φ. Assim $\phi\,\varphi$ termina com sucesso se e só ϕ terminar com sucesso e, em seguida, φ terminar com sucesso.

   A decisão $\phi\|\varphi$ escolhe entre as duas decisões de tal forma que $\phi\|\varphi$ termina com sucesso se e só se que uma das computações terminar com sucesso.
   Se ambas as computações tiverem possibilidade de terminar com sucesso não existe qualquer certeza sobre qual delas é executada.

4. A **recursividade** $\mathsf{rec}\,\xi.\,\varphi(\xi)$

A decisão $\phi \doteq \mathsf{rec}\,\xi.\,\varphi(\xi)$ *pode ser construída como o limite de uma sequência de decisões* $\{\phi_k\}$ *construída indutivamente por*

$$\phi_0 = \mathbf{1} \quad , \quad \phi_{k+1} = \varphi(\phi_k)$$

A menor linguagem gerada pelas regras anteriores representa-se por $\mathcal{D}_{\mathcal{A}}$.

Complexidade das Computações

A complexidade para a execução das decisões depende do tipo de máquina que as implementa; uma *máquina determinística* executa, em cada *slot* de tempo, um único passo de execução; uma máquina dita *não-determinística* executa, no mesmo *slot* de tempo, um passo de execução de cada uma das decisões, ϕ ou φ, numa decisão da forma $\phi \| \varphi$.

A complexidade de um decisão φ mede-se contanto o número de *slots* temporais que ocupa para atingir um resultado (sucesso ou falha). Obviamente que, para poder medir a complexidade, é necessário assumir que a computação termina.

Seja

$$T_\varphi \ : \ \mathbb{B}^\infty \ \rightharpoonup \ \mathbb{N}$$

a função parcial que associa a cada possibilidade $\omega \in \mathbb{B}^\infty$ o número de *slots* temporais usados na execução da decisão φ. A função é parcial porque podem existir possibilidades onde a computação φ não termina.

Vamos assumir que T_φ é expectável no sentido em que $T_\varphi^{-1}(k)$ é um evento de $\mathcal{L}$, para todo $k \geq 0$. Note-se que o conjunto

$$\alpha_\varphi \doteq \bigcup_{k\in\mathbb{N}} T_\varphi^{-1}(k) \tag{14}$$

é também um evento (porque é a união contável de eventos) e representa a terminação de φ; isto é, φ termina na possibilidade ω se e só se $\omega \in \alpha_\varphi$.

A complexidade média da decisão φ é agora definida como a expectativa condicional de T_φ condicionada à terminação da computação; isto é, condicionada ao evento α_φ.

Representa-se essa complexidade por $[\![\varphi]\!]$; portanto, para todo $n \geq 0$, temos

$$\begin{aligned} [\![\varphi]\!](n) &\doteq \mathbb{E}[T_\varphi \mid \alpha_\varphi](n) \\ &= \sum_{k \neq 0} k \cdot \mathbb{E}[T_\varphi^{-1}(k) \mid \alpha_\varphi](n) \end{aligned} \tag{15}$$

O calculo de $[\![\varphi]\!](n)$ é, em geral, extremamente complicado. No entanto é quase sempre possível ter alguma ideia da ordem de grandeza dessa função.

Recordando a notação assimptótica (página **??**) é classificar $[\![\varphi]\!]$ em termos de *ordem de grandeza*

- Se $[\![\varphi]\!](n) = O(p(n))$, em que $p(\cdot)$ é um polinómio positivo em n, então diz-se que φ tem **complexidade polinomial**.
  Consoante o tipo de máquina (determinística ou não-determinística) onde a execução decorre e onde foi medido T_φ, diremos que φ é **determinístico polinomial** ou **não-determinístico polinomial**.
- Se $[\![\varphi]\!](n) = O(2^n)$ (respectivamente, $[\![\varphi]\!](n) = O(\log_2(n))$ então diz-se que φ tem **complexidade exponencial** (respectivamente, **complexidade logarítmica**).

2.6.Caracterização Probabilística das Computações

Independentemente da complexidade e do tipo de máquina que executa a decisão, o significado de uma decisão φ dever ser completamente determinado pela sua caracterização probabilística. É o que veremos em seguida.

Uma decisão φ, perante uma possibilidade concreta $\omega \in \mathbb{B}^\infty$ no seu espaço de possibilidades, pode terminar com sucesso, terminar com falha ou nunca terminar. Deste modo, em termos de possibilidades ω, pode-se caracterizar φ por dois eventos:

(i) O menor evento $\sigma \in \mathcal{L}$ tal que φ termina com sucesso para todo $\omega \in \sigma$. Esse evento é representado por $[\varphi]$

(ii) O menor evento $\sigma \in \mathcal{L}$ tal que φ termina com falha (equivalentemente, $\bar{\varphi}$ termina com sucesso) para todo $\omega \in \sigma$. Será representado, naturalmente, por $[\bar{\varphi}]$.

Devido às boas propriedade de separação de $\mathcal{L}$ pode-se provar que $[\varphi]$ e $[\bar{\varphi}]$ são disjuntos; podem, no entanto, não ser complementares. De facto $[\varphi] \cup [\bar{\varphi}]$ é o evento que determina as possibilidades para as quais φ termina; podem existir ainda possibilidades onde φ não termina.

DEFINIÇÃO 6 *Dada uma decisão φ e um $n \geq 0$, define-se*

$$\{\varphi\}(n) \doteq \mathbb{E}[[\varphi]](n) \tag{16}$$

e, dado um qualquer evento $\alpha \in \mathcal{L}$

$$\{\varphi|\alpha\}(n) \doteq \mathbb{E}[[\varphi]|\alpha](n) \tag{17}$$

No caso particular em que o evento α é da forma $[\phi]$, para uma decisão ϕ, então representa-se $\{\varphi|[\phi]\}$ simplesmente por $\{\varphi|\phi\}$.

Portanto $\{\varphi\}(n)$ denota a expectativa (em $\mathcal{L}_n$) de φ terminar com sucesso, enquanto que $\{\varphi|\alpha\}(n)$ denota a expectativa condicional (em $\mathcal{L}_n$) de φ terminar com sucesso condicionado ao evento α.

Indistinguibilidade

A caracterização probabilística de algoritmos conduz a uma "ferramenta" extremamente potente para comparar computações e que designaremos de **indistinguibilidade**.

Esta noção combina a noção de funções assimptoticamente similares (ver definição **??** na página **??**) com a expectativa de sucesso das decisões.

Na sua versão mais simples o conceito assume duas variantes que se podem definir do modo seguinte.

DEFINIÇÃO 7 *Duas decisões φ, ϕ são **indistinguíveis**, e escreve-se $\varphi \simeq \phi$, se as expectativas de sucesso $\{\varphi\}$ e $\{\phi\}$ são assimptoticamente similares.*

*Para eventos $\alpha, \beta \in \mathcal{L}$, os pares decisão-evento $\langle\varphi|\alpha\rangle$ e $\langle\phi|\beta\rangle$ são **condicionalmente indistinguíveis**, e escreve-se*

$$\langle\varphi|\alpha\rangle \simeq \langle\phi|\beta\rangle$$

se as expectativas condicionais de sucesso $\{\varphi|\alpha\}$ e $\{\phi|\beta\}$ são assimptoticamente similares. A relação $\langle\varphi|\alpha\rangle \simeq \langle\mathbf{1}|\beta\rangle$ representa-se, simplesmente, por $\langle\varphi|\alpha\rangle \simeq \mathbf{1}$.

Esta definição pode-se resumir nas seguintes relações

$$\varphi \simeq \phi \quad \text{sse} \quad \{\varphi\} \sim \{\phi\} \tag{18}$$

$$\langle\varphi|\alpha\rangle \simeq \langle\phi|\beta\rangle \quad \text{sse} \quad \{\varphi|\alpha\} \sim \{\phi|\beta\} \tag{19}$$

Assim são indistinguíveis (ou condicionalmente indistinguíveis) as decisões onde a diferença (em valor absoluto) das expectativas de sucesso tenderem para zero mais rapidamente que o inverso de qualquer polinómio.

Um conceito semelhante é o de **sucesso desprezável**: φ tem sucesso desprezável se a função $\{\varphi\}$ for desprezável. Esta situação representa-se por $\varphi \simeq \mathbf{0}$.

□

A noção de indistinguibilidade apresentada na definição anterior é muito "grosseira": afinal apenas compara decisões pela sua expectativa de sucesso. Isso é insuficiente porque esquece que uma decisão com sucesso afecta o sucesso de decisões futuras.

Convém refinar a definição de modo a considerar equivalentes apenas as decisões que, para além de terminarem com graus de sucesso similares, afectem as computações futuras de uma forma também similar.

É usual comparar as decisões tomando por referência uma classe $\mathcal{C}$ de computações que satisfaçam certas propriedades de relacionadas com a complexidade computacional. Nesta perspectiva, é frequente tomar-mos como referência, por exemplo, a classe das computações polinomiais determinísticas.

No entanto podem ser usadas outras classes de computações e outras interpretaçõs que sejam apropriadas à situação concreta que se pretende descrever. Por exemplo, pode-se interpretar a classe $\mathcal{C}$ como um reportório de "ataques" que um agente hostil tem à sua disposição. Nesta interpretação $\mathcal{C}$ determina a *capacidade* do agente hostil e, de certa forma, o próprio agente hostil.

Qualquer que seja a classe $\mathcal{C}$ (e a interpretação que lhe associamos) as definições essenciais são as mesmas.

Vamos considerar uma primeira versão (ainda não completa) da noção de indistinguibilidade.

Definição 8 (Indistinguibilidade - 1ª versão) *Seja* $\mathcal{C} \subseteq \mathcal{D}_{\mathcal{A}}$ *uma classe de decisões que contém* $\mathbf{1}$. *A relação de* $\mathcal{C}$*-indistinguibilidade de decisões condicionais* $\langle \varphi | \alpha \rangle$ *e* $\langle \phi | \beta \rangle$ *é definida por*

$$\langle \varphi | \alpha \rangle \simeq_{\mathcal{C}} \langle \phi | \beta \rangle \quad sse \qquad (20)$$
$$\forall\, \rho \in \mathcal{C}. \quad \{\varphi\, \rho \mid \alpha\} \sim \{\phi\, \rho \mid \beta\}$$

Esta versão particular de indistinguibilidade pode-se classificar como **orientada ao "estado"**.

De facto o que (**??**) afirma é que, para qualquer continuação ρ, as computações ϕ e φ têm (após essa continuação) expectativas de sucesso similares. Isto significa que ϕ e φ atingem, após execução, estados que não podem ser discriminados por qualquer continuação.

Nota

Mesmo sem ser completa, a definição **??** assume várias formas particulares consoante as restrições que são impostas nas sua várias componentes. Alguns exemplos

(1) Se restringirmos a classe de computações $\mathcal{C}$ ao conjunto singular $\{\mathbf{1}\}$ formado pela decisão unitária, então a relação $\simeq_{\mathcal{C}}$ é equivalente à relação $\simeq$ (definição **??**).

(2) Se considerarmos $\varphi = \phi = \mathbf{1}$ então (**??**) toma a forma

$$\langle \mathbf{1} \mid \alpha \rangle \simeq_{\mathcal{C}} \langle \mathbf{1} \mid \beta \rangle \quad \text{sse} \qquad \forall \rho \in \mathcal{C}. \quad \{\rho \mid \alpha\} \sim \{\rho \mid \beta\}$$

Nestas circunstâncias diremos que os eventos α, β são $\mathcal{C}$-**indistinguíveis** e escrevemos, simplesmente,

$$\alpha \simeq_{\mathcal{C}} \beta$$

Esta noção compara dois eventos através da forma como condicionam o sucesso das computações na classe $\mathcal{C}$: se não for possível encontrar uma decisão $\rho \in \mathcal{C}$ que condicionada aos eventos α e β num dos casos tenha sucesso e no outro não, então os eventos não se podem distinguir pelas computações $\mathcal{C}$.

A definição **??** não é suficientemente forte. Frequentemente é necessário usar um conceito de indistinguibilidade que compare duas computações num contexto de uma terceira computação arbitrária.

Para isso precisamos de introduzir a noção de "padrões de computação" que são, essencialmente, computações onde aparecem variáveis.

DEFINIÇÃO 9 *Considere-se a linguagem* $\mathcal{D}_{\mathcal{A}+X}$ *que se obtém de* $\mathcal{D}_{\mathcal{A}}$ *acrescentando às computações primitivas uma variável* X*. Os elementos de* $\mathcal{D}_{\mathcal{A}+X}$ *chamam-se* **padrões** *em* $\mathcal{D}_{\mathcal{A}}$*.*

Para cada padrão $\phi \in \mathcal{D}_{\mathcal{A}+X}$ *e cada* $\varphi \in \mathcal{D}_{\mathcal{A}}$*, a notação* ϕ^{φ} *representa a computação que se obtém substituindo* X *por* φ *em* ϕ*.*

Para uma classe de computações $\mathcal{C}$*, designaremos por* $\mathcal{C}^X$ *a classe de todos os padrões* ϕ *tais que* $\phi^{\mathbf{1}}$ *é um elemento de* $\mathcal{C}$

Com estes conceitos já é possível definir uma noção mais forte de indistinguibilidade.

DEFINIÇÃO 10 (INDISTINGUIBILIDADE - 2^{A} VERSÃO) *A relação de* $\mathcal{C}$-**indistinguibilidade** *entre decisões condicionais* $\langle \rho \mid \alpha \rangle$ *e* $\langle \varphi \mid \beta \rangle$ *define-se por*

$$\langle \rho \mid \alpha \rangle \simeq_{\mathcal{C}} \langle \varphi \mid \beta \rangle \quad sse \qquad (21)$$
$$\forall \phi \in \mathcal{C}^X : \quad \{\phi^{\rho} \mid \alpha\} \sim \{\phi^{\varphi} \mid \beta\}$$

Esta definição diz-se **orientada ao oráculo**. Essencialmente afirma que o uso de ρ ou φ como "oráculos" num mesmo padrão ϕ conduz a decisões indistinguíveis.

É uma noção muito forte que inclui, por exemplo, a indistinguibilidade orientada ao estado. De facto a indistinguinbilidade orientada ao estado

(definição **??**) obtém-se usando padrões da forma $X\,\varrho$, com ϱ arbitrário numa classe $\mathcal{C}$.

2.7.Exemplos

Exemplo 11 (Probabilidade vs. Expectativa):

Muitos algoritmos de procura importantes em Criptografia baseiam-se numa estratégia muito simples que podemos designar por *"força bruta"*.

Imagine-se um predicado que age sobre palavras de 128 *bits*[9] representado por um teste $\mathcal{Q} : \mathbb{B}^{128} \longrightarrow \mathbb{B}$. O objectivo é encontrar uma palavra $x \in \mathbb{B}^{128}$ que verifica $\mathcal{Q}(x)$.

Se não soubermos nada sobre $\mathcal{Q}$ para além de um algoritmo para calcular a função, a única estratégia de procura é a mais simples possível: gerar sucessivamente palavras de 128 *bits* aleatórias e aplicar o teste a cada uma parando só quando o teste tiver sucesso.

Essa estratégia é implementada no algoritmo **??**. Propomos caracterizar a sua complexidade (medindo os ciclos que executa) e a sua probabilidade de terminar com sucesso.

Estado: Ω *contém uma string infinita de bits aleatórios*

repeat
  $x \leftarrow \text{next}(128, \Omega)$ # *retira os próximos* 128 *bits de* Ω
until $\mathcal{Q}(x)$

Algoritmo 1: ataque "força bruta" sobre o teste $\mathcal{Q}$.

Muitas vezes este tipo de problema coloca-se quando se pretende fazer a chamada *criptoanálise* de uma técnica criptográfica qualquer; por exemplo, $\mathcal{Q}$ pode descrever o comportamento de uma cifra.

Nestas circunstâncias é razoável assumir que a solução da equação $\mathcal{Q}(x) = 1$ é única. Existem 2^{128} argumentos diferentes de $\mathcal{Q}(\cdot)$; por isso a probabilidade de sucesso do teste $\mathcal{Q}(x)$, assumindo a aleatoriedade perfeita no valor de Ω, será $1/2^{128} = 2^{-128}$.

Note-se que 2^{128} *é maior do que o número estimado de partículas do universo; por isso* 2^{-128} *é um probabilidade realmente pequena!*

[9] 128 *bits* é o tamanho de chave *standard* para as modernas cifras simétricas.

O número de ciclos que o algoritmo executa depende do valor de Ω. Do mesmo modo a probabilidade de o algoritmo terminar depende também do mesmo valor. Assim, neste algoritmo, Ω é a fonte de aleatoriedade e os seus possíveis valores formam o nosso espaço de possibilidades.

Este algoritmo tem a particularidade de terminar sempre com sucesso desde que realmente termine; logo, a probabilidade de sucesso é, neste caso, igual à probabilidade de terminar.

A primeira questão que se pode colocar será: em que circunstâncias o algoritmo não termina?

A menos que $\mathcal{Q}(x)$ seja válido para todo x (o que seria um caso de muito pouco interesse pratico!), existe sempre a possibilidade do algoritmo não terminar; basta pensar num valor de Ω formado por uma repetição infinita de palavras z tais que $\mathcal{Q}(z) = 0$. Como assumimos que todos os possíveis argumentos x, excepto um, estão nestas circunstâncias, então existem muitas possibilidade de onde o algoritmo não termina.

Paradoxalmente, apesar de existirem todas essas possibilidades de não terminar, a probabilidade do algoritmo não terminar é 0.

Para justificar esta afirmação vamos começar por introduzir alguma notação:

(i) $p = 1 - 2^{-128}$ é a probabilidade do teste $\mathcal{Q}(x)$ falhar assumindo aleatoriedade perfeita no valor de Ω

(ii) φ é a computação descrita pelo algoritmo **??**

(iii) $N(n) = \lfloor n/128 \rfloor$ é o maior inteiro menor do que $n/128$; indica o número de ciclos que é possível formar conhecendo-se apenas os primeiros n *bits* de Ω.

A probabilidade de o algoritmo não terminar ao fim de k ciclos é p^k. A probabilidade de terminar exactamente com k ciclos será

$$p^{k-1} \cdot (1 - p)$$

uma vez que entra com a componente p^{k-1} que mede a probabilidade de ter falhado nos $k - 1$ ciclos anteriores e a componente $(1 - p)$ que dita a probabilidade de ter sucedido exactamente no ciclo de ordem k.

Olhando apenas para os primeiros n *bits* de Ω, temos informação suficiente para $N(n)$ ciclos. Portanto a probabilidade de o algoritmo ter terminado com sucesso com essa

informação é

$$\{\varphi\}(n) \;=\; \sum_{k=1}^{N(n)} p^{k-1}\cdot(1-p) \;=\; 1-p^{N(n)}$$

Apesar de p ser muito próximo de 1, ainda assim é menor do que 1; por isso, quando n tende para infinito, $N(n)$ tende para infinito e $p^{N(n)}$ tende para zero; donde

$$\lim_{n\longrightarrow\infty} \{\varphi\}(n) \;=\; 0$$

Em conclusão (e este é o paradoxo) a probabilidade de φ terminar com sucesso é 1 mas, ainda assim, não se pode afirmar que esta computação termina sempre com sucesso; de facto não termina num número infinito de possibilidades mas, globalmente, essas possibilidade formam um evento de medida nula.

A expectativa $\{\varphi\}(n)$ dá uma informação muito mais rica do que simples probabilidade.

Supúnhamos que o nosso algoritmo conseguia ler 1000 Gigabits ($\sim 2^{40}$) por segundo e corria durante 1 ano ($\sim 2^{25}$ segundos); estar-se-ia a recolher os primeiros 2^{65} bits de Ω.

Com esta informação seriam executados $N(2^{65}) = 2^{65}/128 = 2^{58}$ ciclos. Calculando a expectativa $\{\varphi\}(n) = 1-p^{N(n)}$, com $n = 2^{65}$, obtemos um valor da ordem de 2^{-70}; isto é uma probabilidade extremamente baixa.

Desta forma a probabilidade dá-nos uma mensagem e a expectativa dá-nos a mensagem oposta. A probabilidade diz-nos que, no limite, o algoritmo termina com elevada probabilidade; a expectativa diz-nos que, com a informação que se pode recolher num tempo razoável, a computação tem uma probabilidade de terminar ínfima.

O mesmo tipo de mensagem se pode recolher se calcular-mos a complexidade média. Recorde-se que a probabilidade de o número de ciclos executado ser exactamente k é $p^{k-1}\cdot(1-p)$.

A expectativa para o número de ciclos será

$$\sum_{k=1}^{\infty} k\cdot p^{k-1}\cdot(1-p) \;=\; \frac{1}{1-p} \;=\; 2^{128}$$

o que é um valor enorme.

Exemplo 12 (Indistinguibilidade):

Cifras simétricas são, como veremos, as técnicas criptográficas mais usadas em qualquer situação onde seja crucial a confidencialidade das mensagens; nomeadamente, quase todo o protocolo criptográfico, mesmo que recorra a outro tipo de cifras, acaba por basear a sua segurança na segurança de uma cifra simétrica.

Por isso é essencial ter uma ideia clara do que significa a segurança de uma cifra e, não é de admirar, que este seja um dos problemas mais bem estudados em Criptografia.

Considere-se a seguinte computação parametrizada por um parâmetro k que percorre um conjunto finito de possibilidades. Sem perda de generalidade pode-se supor que k percorre um domínio da forma $\mathbb{B}^r$.

Vamos supor que a implementação a cifra (cujos detalhes não são relevantes neste momento) é representada por uma família de funções $F(k,\cdot) : \mathbb{B}^t \to \mathbb{B}^t$. Vamos supor que existe uma segunda família de funções $G(k,\cdot) : \mathbb{B}^t \to \mathbb{B}^t$ que "decifram" no sentido em que especificaremos em seguida. [10]

Estado: Ω *com valores em* $\mathbb{B}^\infty$ *e* $\mathbf{x}$ *com valores em* $\mathbb{B}^t$

$\mathbf{c}_k()$
$\quad \mathbf{x} \leftarrow \text{next}(t,\Omega)$
$\quad \mathbf{x} \leftarrow F(k,\mathbf{x})$

$\mathbf{d}_k()$
$\quad \mathbf{x} \leftarrow G(k,\mathbf{x})$

$\mathbf{in}()$
$\quad \mathbf{x} \leftarrow \text{next}(t,\Omega)$

Algoritmo 2: Um passo de cifra, um passo de decifra e a cifra nula

No algoritmo **??** estão representadas três computações (realmente, *famílias* de computações):

- a primeira $\mathbf{c}_k()$ executa um passo de cifra recolhendo um valor aleatório em $\mathbf{x}$ e transformando-o, em seguida, com $F(k,\cdot)$.
- a segunda $\mathbf{d}_k()$ actua directamente sobre o valor de $\mathbf{x}$ aplicando-lhe $G(k,\cdot)$.
- a última $\mathbf{in}()$ não faz nada para além de recolher um valor aleatório em $\mathbf{x}$.

[10] Na terminologia das cifras simétricas, k designa-se por **chave** e a ordem r designa-se por **comprimento da chave**. O parâmetro t é designado por **comprimento do bloco**.

Em primeiro lugar é necessário especificar o que se entende por uma **cifra correcta**. Essencialmente isto traduz duas propriedades:

(i) decifrar com uma chave depois de cifrar com a mesma chave deve reproduzir a mensagem inicial, e

(ii) decifrar com uma chave depois de cifrar com uma chave "errada" deve produzir "lixo" se comparado com o resultado expectável (a reprodução da mensagem inicial).

Como é que sabemos que o "resultado" de decifrar é ou não equivalente a reproduzir a mensagem original?

Essencialmente, fazendo *testes*; se aplicando todos os testes (dentro de uma classe escolhida criteriosamente) ao resultado de decifrar e à mensagem original dar origem às mesmas decisões, então pode-se concluir que temos resultados equivalentes.

Nesta perspectiva o primeiro passo deve ser a escolha de uma classe apropriada de testes. Por exemplo, pode-se escolher a classe P de todas as decisões polinomiais determinísticas. Isto é estamos a escolher apenas testes que têm uma implementação computacional eficiente numa máquina determinística.

A primeira condição pode ser escrita

$$\forall k \in \mathbb{B}^r \ , \ \forall \rho \in \mathrm{P}: \quad \{\mathbf{c}_k \, \mathbf{d}_k \, \rho\} \ \sim \ \{\mathbf{in} \, \rho\} \tag{22}$$

que pode ser escrita, simplesmente, como

$$\forall k \in \mathbb{B}^r: \quad \mathbf{c}_k \, \mathbf{d}_k \ \simeq_{\mathrm{P}} \ \mathbf{in}$$

O que é que isto significa?

A computação $\langle \mathbf{c}_k \, \mathbf{d}_k \, \rho \rangle$ indica o seguinte processo: recolhe-se um valor aleatório em $\mathbf{x}$, transforma-se esse valor aplicando-lhe $F(k, \cdot)$, volta-se a transformar aplicando-lhe $G(k, \cdot)$ e, finalmente, aplica-se o teste ρ.

A computação $\langle \mathbf{in} \, \rho \rangle$ recolhe o valor aleatório em $\mathbf{x}$ e, sem o submeter a qualquer transformação prévia, aplica-lhe o teste ρ. A condição (**??**) exige que a expectativa destas duas computações sejam similares (i.e. a diferença entre as expectativas de sucesso é uma função desprezável).

A segunda condição lida com a situação em que se decifra com uma chave distinta da chave usada para cifrar. Quere-se afirmar que isso deve produzir "lixo".

Uma primeira abordagem seria afirmar, simplesmente, que os resultado de cifrar com k e decifrar com k' (com $k \neq k'$) são distinguíveis; isto é

$$\forall k, k' \in \mathbb{B}^r : \quad (k \neq k') \quad \Longrightarrow \quad \mathbf{c}_k \, \mathbf{d}_{k'} \not\simeq_{\mathrm{P}} \mathbf{in} \tag{23}$$

No entanto, ao exigirmos que o resultado, comparado com o resultado expectado, deve ser "lixo", estamos a colocar uma condição mais forte.

Uma estratégia passa por seleccionar, dentro da classe dos testes P, uma sub-classe de **testes raros**; isto é, testes que aplicados a possibilidades completamente aleatórias têm uma expectativa de sucesso desprezável. Isto não impede, como vimos no exemplo anterior, que exista um evento não-vazio (eventualmente infinito) onde o teste tem sucesso.

Vamos então definir a classe dos testes raros como

$$\mathrm{R} \doteq \quad \{ \, \rho \in \mathrm{P} \mid \{\rho\} \sim \mathbf{0} \, \} \tag{24}$$

Esquecendo, por momentos, as cifras e pensando em computações genérica, uma possível definição da noção

> *na esperança de se obter um resultado equivalente ao da computação φ, o resultado da computação ϕ é "lixo"*

será

$$\forall \rho \in \mathrm{R} : \quad \{\phi \, \rho \mid \varphi \, \rho\} \sim \mathbf{0} \tag{25}$$

A definição (**??**) exige algumas explicações.

Essencialmente $\langle \phi \, \rho \rangle$ corresponde em computar ϕ e, sobre os seus resultados, efectuar o teste ρ. Ao condicionarmos a expectativa a $\langle \varphi \, \rho \rangle$ isto significa que estamos a olhar apenas para as possibilidades onde este mesmo processo, mas realizado com φ, teve sucesso.

Portanto estamos a condicionar a observação de ϕ às situações onde o resultado que pretendemos obter (o resultado de φ) foi testado com sucesso. A definição impõe que, mesmo nesses casos, a expectativa de sucesso do teste aplicado ao resultado de ϕ é desprezável. Isto é, os resultados que se obtém de ϕ em nada se comparam com os resultados de φ.

Pode-se comparar os teste raros com uma colecção de "peneiras" finas que só deixam passar os resultados desejados. Um teste raro, que tenha sucesso nos resultados de φ, tem, necessariamente, conhecimento prévio sobre φ. A definição exige que tal teste não deve deixar passar os resultados de ϕ.

Com esta notação, a correcção da cifra pode-se completar exigindo que, para todo par de chaves $k \neq k' \in \mathbb{B}^r$ e para todo o teste raro $\rho \in \mathrm{R}$,

$$\{\mathbf{c}_k\, \mathbf{d}_{k'}\, \rho \mid \mathbf{c}_k\, \mathbf{d}_k\, \rho\} \;\sim\; \mathbf{0} \tag{26}$$

□

Falta analisar a **segurança** das cifras. O que é que se pretende garantir?

Para falar de segurança temos de ter em atenção que existe um atacante com conhecimento de determinados items informação; as condições de segurança avaliam aquilo que o atacante é capaz de computar com esse conhecimento. Neste caso, o *"conhecimento"* é:

(i) O atacante conhece a família de funções $\mathbf{c}_k$ e $\mathbf{d}_k$, no sentido em que, se conhecer a chave k, consegue executar qualquer uma destas computações.

(ii) O atacante conhece também uma computação particular $\mathbf{c}$ que é indistinguível de uma das $\mathbf{c}_k$ mas desconhece qual.

A *"capacidade de computar"* é avaliada em termos das chamadas **condições de segurança**; dessas destacam-se:

(i) O atacante, que tenha acesso a um resultado de $\mathbf{c}$, não deve ser capaz de reconstruir o texto que originou esse resultado (*não-invertibilidade do criptograma*)[11].

(ii) O atacante, que tenha acesso ao resultado de $\mathbf{c}$, não deve ser capaz de distinguir esse resultado de uma *string* aleatória (*segurança perfeita*).

(iii) O atacante, mesmo com acesso aos textos claros, não deve ser capaz de computar a chave k para a qual $\mathbf{c}_k$ é indistinguível de $\mathbf{c}$ (*não-invertibilidade da chave*).

Se um atacante for capaz de inverter a chave (computar k tal que $\mathbf{c} \simeq_P \mathbf{c}_k$) então pode usar $\mathbf{d}_k$ para inverter o criptograma. Também o resultado da cifragem não pode

[11] Na terminologia das cifras, o argumento da função de cifragem chama-se **mensagem** ou **texto claro** (abreviado para **texto**). O resultado da cifragem chama-se **criptograma**.

ser considerado aleatório. Portanto a invertibilidade da chave implica a invertibilidade do criptograma.

O inverso já não se verifica: mesmo que seja possível descobrir o texto que deu origem a um criptograma conhecido, nada indica que esta informação permita descobrir a chave que foi usada para cifrar.

Pode-se tentar exprimir não-invertibilidade em termos de testes computacionalmente eficientes.

Suponhamos, por exemplo, que era possível descobrir o valor do 2^{a} *bit* do texto fazendo apenas cálculos simples com o criptograma; de certeza que tal cifra não seria segura.

Podemos generalizar esta condição para qualquer propriedade do texto representando essa propriedade por um computação $\phi \in \mathrm{P}$ e escrever:

$$\forall \phi \in \mathrm{P},\ \exists \rho \in \mathrm{P}\ : \qquad \{\mathbf{c}\,\rho\} \ \sim\ \{\phi\}$$

Esta não é uma condição de segurança; pelo contrário é uma condição de forte insegurança já que representa a invertibilidade perfeita: qualquer decisão tratável que se possa pensar fazer com o texto, tem uma contrapartida nas decisões que se podem fazer no criptograma; observando o criptograma com ρ determina-se como é que o texto se comporta com ϕ.

Uma análise mais realista parte do princípio que existe uma colecção de computações críticas $\mathcal{S}$ cujo sucesso compromete a segurança do texto: a segurança é comprometida logo que exista um $\phi \in \mathcal{S}$ cujo sucesso seja previsível; se for previsível observando o criptograma temos uma quebra de segurança.

Pode-se, agora, escrever a condição de não-invertibilidade do criptograma como:

$$\forall \phi \in \mathcal{S}, \forall \rho \in \mathrm{P}\ : \qquad \{\phi\} \ \not\sim\ \{\mathbf{c}\,\rho\} \tag{27}$$

Esta condição diz-nos que, para todos os testes críticos ϕ, não pode existir nenhum ρ que, no criptograma, possa prever o resultado de ϕ no texto claro.

Uma forma alternativa, que evita negações nos testes de indistinguibilidade e traduz melhor o papel das computações críticas, recorre a testes raros. Entende-se que, qualquer teste raro que eventualmente tenha sucesso, fornece informação preciosa sobre o conjunto de possibilidades onde esse sucesso ocorreu. Por isso pode-se assumir que todo o teste raro é uma computação crítica.

Considere-se a seguinte condição

$$\forall \phi \in \mathrm{R}, \forall \rho \in \mathrm{P} : \qquad \{\phi \mid \mathbf{c}\, \rho\} \ \sim \ \mathbf{0} \tag{28}$$

Isto diz-nos que para um teste raro ϕ ter sucesso no *input* da cifra (o texto claro) é irrelevante saber-se o resultado de qualquer teste ρ efectuado sobre o criptograma; para efeitos do conhecimento do texto, o criptograma não se distingue de uma *string* aleatória.

Esta é a formalização da condição de segurança perfeita.

Falta formalizar a condição de não-invertibilidade da chave.

Uma forma semelhante à usada para definir a invertibilidade do criptograma consiste em definir uma "cifra generalizada" da forma seguinte:

Estado: Ω *com valores em* $\mathbb{B}^\infty$

$\mathbf{C}(n)$
  $\mathbf{k} \leftarrow \mathsf{next}(r, \Omega)$
  foreach $i \in 1..n$
    $\mathbf{x}_i \leftarrow \mathsf{next}(t, \Omega)$
    $\mathbf{y}_i \leftarrow F(\mathbf{k}, \mathbf{x})$
  $\mathbf{k} \leftarrow \mathsf{next}(r, \Omega)$

Algoritmo 3:

Esta computação gera primeiro uma chave aleatória e depois recolhe n amostras de pares (x_i, y_i) (texto e respectivo criptograma); antes de terminar "esconde" a chave usada como um novo valor aleatório. Qualquer computação que continue $\mathbf{C}$ pode dispor dos N valores $(\mathbf{x}_i, \mathbf{y}_i)$ mas não tem acesso à chave usada para construir esses pares.

A invertibilidade perfeita da chave seria testada por uma condição do tipo

$$\forall \phi \in \mathrm{P} \ , \ \exists \rho \in \mathrm{P} : \qquad \{\mathbf{C}(n)\, \rho\} \ \sim \ \{\phi\} \tag{29}$$

A não-invertibilidade usa, tal como em (**??**), uma classe de testes críticos $\mathcal{S}$.

$$\forall \phi \in \mathcal{S} \ , \ \forall \rho \in \mathrm{P} : \qquad \{\mathbf{C}(n)\, \rho\} \ \not\sim \ \{\phi\} \tag{30}$$

Este processo pode ser melhor definido comparando o comportamento de apenas duas instâncias, com chaves distintas, da cifra $\mathbf{c}_k$

Vamos considerar um esquema de decisão ϕ gerada pelo conjunto de decisões atómica que inclui uma única variável X.

A notação ϕ^ρ denota a computação que se obtém de ϕ substituindo X por ρ. A notação $\phi \in \mathrm{P}^X$ significa

$$\textit{para toda a decisão } \rho \in \mathrm{P}, \textit{ a computação } \phi^\rho \textit{ também está em } \mathrm{P}$$

Uma condição forte quanto à não-invertibilidade da chave será agora

$$\forall \phi \in \mathrm{P}^X : \qquad \{\phi^{\mathbf{c}_k}\} \sim \{\phi^{(\mathbf{c}_k \| \mathbf{c}_{k'})}\} \tag{31}$$

para todo o par de chaves distintas $k \neq k' \in \mathbb{B}^r$.

Apesar da aparente complexidade esta condição tem uma interpretação simples:

1. tem-se à disposição duas "caixas pretas" $\mathbf{c}_k$ e $\mathbf{c}_{k'}$ correspondentes à execução da cifra com duas chaves, k e k', eventualmente distintas mas desconhecidas,
2. possui-se uma colecção de testes eficientes ϕ que contêm um parâmetro X que pode ser substituído por qualquer computação supostamente eficiente,
3. a computação que substituir X pode ser usada várias vezes dentro de ϕ; em cada uma das suas ocorrências pode usar resultados de outras computações incluindo resultados de ocorrências prévias dela própria,
4. a substituição é feita por uma de duas computações:
   (i) por $(\mathbf{c}_k \| \mathbf{c}_{k'})$ que significa usar a cifra com a chave k ou com a chave k' sem se saber exactamente qual,
   (ii) ou então por $\mathbf{c}_k$ que significa usar sempre a cifra com a chave k
5. a condição (**??**) diz-nos que ϕ não consegue distinguir entre estas duas situações; isto é, pelo resultado de ϕ não se sabe qual foi a chave usada.

□

É possível estabelecer outras condições de segurança para cifras que são variantes destas condições.

Muitas vezes basta-nos as duas condições ((**??**) e (**??**)) para caracterizar com bastante detalhe essa segurança. Por vezes, porém, estas condições são excessivamente fortes e necessitam de ser adaptadas ao problema particular onde se quer incluir a segurança das cifras.

3.Princípios Nucleares

O desenvolvimento de quaisquer técnicas criptográficas pressupõe um conjunto de princípios e notações que são essenciais a quase todas elas.

Noções como **computação**, **algoritmo**, **verificação** e **aleatoriedade** podem parecer razoavelmente familiares para não necessitar um estudo específico.

Já noções como **entropia**, **unidirecionalidade**, **conhecimento zero** e **função de "hash"** parecem ser específicas da área da Criptografia e, por isso, necessitam de alguma análise prévia.

Facilmente se verifica, porém, que mesmo o primeiro grupo de noções, quando usadas nas técnicas criptográficas, apresenta particularidades que exigem uma análise específica.

Neste capítulo iremos apresentar (de forma sucinta) estas noções fundamentais que são transversais a todo o estudo da Criptografia.

3.1.Incerteza, Compressão e Codificação

A probabilidade de um evento mede, de alguma forma, o grau de "certeza" que temos sobre a ocorrência de um evento. Um ponto de vista complementar é baseado na noção de **incerteza**. De forma heurística

> *A* ***incerteza*** *de um evento mede a quantidade de informação que é preciso possuir para forçar que o evento ocorra.*

Formalmente a incerteza de um evento α, representado por $\vartheta(\alpha)$, é

$$\vartheta(\alpha) \doteq -\log_2 \mu(\alpha) \tag{32}$$

que garante a relação $\mu(\alpha) = 2^{-\vartheta(\alpha)}$.

Exemplo 13 : Para um evento $\uparrow u$ e incerteza é

$$\vartheta(\uparrow u) = -\log_2 2^{-|u|} = |u|$$

Isto é, a incerteza num evento da forma $\uparrow u$ é, simplesmente, o comprimento da pega u.

Esta noção estende-se a variáveis aleatórias $X : \mathbb{B}^\infty \longrightarrow \mathbb{N}$ expectáveis; isto é, variáveis para as quais $X_k \doteq X^{-1}(k)$ é sempre um evento.

Recordemos que X_k denota o evento *"a variável X tem o valor k"* e que a colecção $\{X_k\}$ de todos estes eventos caracteriza, em termos de medida, a variável X.

Pode-se calcular todas as incertezas $\vartheta(X_k)$ e, em seguida, calcular a sua média (pesada com a respectiva probabilidade).

$$H(X) \doteq \sum_{k=0}^{\infty} \mu(X_k) \cdot \vartheta(X_k) \tag{33}$$

Esta "incerteza média" designa-se por **entropia** da variável X.

Exemplo 14 : Considere-se o algoritmo "força bruta" estudado no exemplo **??** (página **??**).

Seja X, neste exemplo, a variável aleatória que mede o número de ciclos que o algoritmo executa antes de terminar.

Recordemos que $p = 1 - 2^{-128}$ era a probabilidade de, num único teste, o algoritmo não terminar. A probabilidade de terminar ao fim de k testes era

$$\mu(X_k) \;=\; p^{k-1} \cdot (1-p)$$

A incerteza do evento X_k é

$$\vartheta(X_k) \;=\; -(k-1) \cdot \log_2 p - \log_2(1-p)$$

A entropia $H(X)$ é, expandindo (**??**),

$$H(X) \;=\; -\frac{p}{1-p} \cdot \log_2 p - \log_2(1-p) \;\sim\; 129.44$$

É curioso observar a entropia do número de ciclos de teste é uma valor ligeiramente superior ao logaritmo da complexidade média do algoritmo (que, neste caso, é 2^{128}).

É importante definir também *incerteza condicional* e *entropia condicional*. Para isso já não é possível lidar directamente com probabilidades mas as definições têm de passar pelas expectativas.

Para eventos definem-se duas noções fortemente relacionadas: a noção de "incerteza condicional" e a noção de "conhecimento".

(i) a **incerteza condicional** do evento α condicionado a β define-se

$$\vartheta(\alpha \mid \beta) \;\doteq\; \lim_{n\to\infty} -\log_2(\mathbb{E}^{\mu}[\alpha \mid \beta](n)) \qquad (34)$$

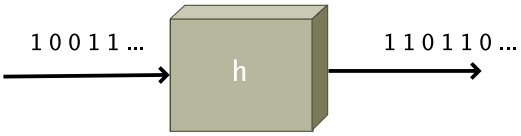
(ii) o **conhecimento** que β possui acerca de α define-se

$$\mathcal{I}(\beta : \alpha) \;\doteq\; \vartheta(\alpha) - \vartheta(\alpha \mid \beta) \qquad (35)$$

Compressão

Uma função não-enumerável $h : \mathbb{B}^{\infty} \to \mathbb{B}^{\infty}$ pode ser vista como uma variável aleatória com valores reais no intervalo $[0, 1]$.

Pode também ser vista como uma "caixa preta" com dois *ports*: uma entrada e uma saída. Em ambos observam-se strings infinitas de *bits*.



A visão probabilística destas "caixas" assume que o conteúdo de qualquer um dos *ports* é visto como um espaço de possibilidades e a caracterização da função h é feita em termos dos eventos que se formam com estas possibilidades.

Assume também assume que não existe nenhum controlo particular sobre qualquer dos *ports*. De facto, em princípio, o observador não sabe sequer qual é o *port* que é *input* e qual é o que é *output*. Ele apenas pode observar o que ocorre em cada um deles, para isso dispondo da capacidade de detectar se certas *propriedades* são verificadas ou se certos *eventos* ocorrem.

Para que um observador sem qualquer informação prévia consiga distinguir os *ports* (identificando qual deles é o *input* e qual é o *output*) através de eventos (ou propriedades) é necessário que seja capaz de estabelecer uma relação de causa e efeito entre tais eventos. Se for capaz de de afirmar que um evento que ocorre num dos *ports* é o efeito de um outro evento (a causa) que ocorre no outro *port*, então o primeiro *port* será o *output* e o segundo será o *input*.

A noção de função mensurável estabelece o grau de assimetria necessário para possibilitar a descriminação dos *ports*.

O facto de h ser mensurável indica-nos que, para qulquer evento α observável no *output*, existe um evento correspondente $h^{-1}(\alpha)$ observável no *input*. O evento $h^{-1}(\alpha)$ é formado por todas as possibilidades de *input* que produzem possibilidades de *output* em α e, por conseguinte, é identificável como a *causa* que produz o *efeito* α.

□

Esta propriedade (a mensurabilidade) é necessária para que a relação se causalidade exista; não é, porém, suficiente para que seja tratável estabelecer tal relação.

Para se ver a razão desta observação, sopunhamos que se escolhe um evento α que seja observável lendo n bits; por exemplo, quando α é da forma $\uparrow u$ e $|u| = n$. Suponhamos também que se tinha $h^{-1}(\alpha) = \{\omega\}$, com $\omega \in \mathbb{B}^\infty$, um evento singular.

As exigências de mensurabilidade de h são satisfeitas. No entanto não é possível estabelecer uma relação de causa-efeito já que, para ser possível constatar que a causa $h^{-1}(\alpha)$ ocorre, é necessário ler um número infinito de bits; enquanto o efeito tem uma incerteza finita de n bits, a potencial causa tem incerteza infinita.

Este exemplo sugere que, para ser viável estabelecer a relação entre a causa e o efeito, é necessário que ambos os eventos sejam observáveis de forma computacionalmente tratável.

Mais precisamente, a incerteza de ambos os eventos deve ser limitada de alguma forma; idealmente, a incerteza na causa deverá ser sempre menos do que a incerteza no efeito. Quando isto ocorre diremos que se está em presença de uma *compressão*.

□

As características da função h estabelecem uma relação entre a incerteza do efeito $\vartheta(\alpha)$ e a incerteza da causa $\vartheta(h^{-1}(\alpha))$. Alguns exemplos

Exemplo 15 :

1. Considere-se o evento $\alpha = \uparrow 1$ formado por todas as strings que têm 1 como primeiro *bit*. O seu complemento $\bar{\alpha}$ é, obviamente, o evento $\uparrow 0$. Tem-se, em cada caso,

$$\vartheta(\alpha) = \vartheta(\bar{\alpha}) = -\log_2(1/2) = 1$$

Ambos eventos podem ser detectados lendo apenas 1 bit.
Suponhamos que a função h era tal que todos os seus possíveis resultados iniciavam-se como o *bit* 1. Isto significa um efeito que se observa a lendo 0 *bits*.
De facto $h^{-1}(\alpha) = \mathbb{B}^\infty$ e $h^{-1}(\bar{\alpha}) = \emptyset$ e, portanto,

$$\vartheta(h^{-1}(\alpha)) \;=\; 0 \qquad \text{e} \qquad \vartheta(h^{-1}(\bar{\alpha})) \;=\; \infty$$

2. Se α for um evento de baixa probabilidade (ou, equivalentemente, elevada incerteza) é natural que $h^{-1}(\alpha)$ também seja um evento muito incerto. Nem sempre isso acontece, porém.

   Suponha-se a função h mapeia qualquer argumento em 1^∞ (uma sequência infinita de bits 1) ou 0^∞ consoante o primeiro bit do argumento é 1 ou 0. Isto equivale a

   $$h^{-1}(\{1^\infty\}) \;=\uparrow 1 \qquad , \qquad h^{-1}(\{0^\infty\}) \;=\uparrow 0$$

   e

   $$\vartheta(h^{-1}(\{1^\infty\})) \;=\; \vartheta(h^{-1}(\{0^\infty\})) \;=\; 1$$

   Considere-se uma família $\alpha_n =\uparrow(1^n)$ de eventos observados no *output*; o evento α_n ocorre se e só se são observados n sucessivos 1's no resultado da função. A sua incerteza é

   $$\vartheta(\alpha_n) \;=\; -\log_2 2^{-|1^n|} \;=\; |1^n| \;=\; n$$

   Quanto aos eventos $h^{-1}(\alpha_n)$ correspondentes, dado 1^∞ está contido em todo α_n e 0^∞ em nenhum deles, tem-se $h^{-1}(\alpha_n) =\uparrow 1$. Portanto

   $$\vartheta(h^{-1}(\alpha_n)) \;=\; 1$$

   Neste exemplo a incerteza dos eventos α_n cresce com n; no entanto a incerteza dos correspondentes $\vartheta(h^{-1}(\alpha_n))$ é constante e igual a 1.

□

Vimos que para ser possível estabelecer uma relação entre causa e e efeito (ou entre *input* e *output*) era necessário limitar as incertezas nos eventos observáveis. A relação entre a incerteza no *output* α e a respectiva incerteza no *input* $h^{-1}(\alpha_n)$ fornece essa informação essencial sobre as características da função h. A diferença

$$\vartheta(\alpha) - \vartheta(h^{-1}(\alpha)) \tag{36}$$

mede o "ganho de incerteza" no efeito em relação à causa.

Se este valor for ≥ 0 isto significa que o efeito é mais incerto que a causa. Inversamente, se esta diferença for negativa, a incerteza na causa é maior do que no efeito e isso pode dificultar o estabelecimento de uma relação causa-efeito computacionalmente tratável.

Exemplo 16 : Voltando aos casos estudados no exemplo **??**, temos

1. Neste caso tinhamos $\vartheta(\alpha) = \vartheta(\bar{\alpha}) = 1$, $\vartheta(h^{-1}(\alpha)) = 0$ e $\vartheta(h^{-1}(\bar{\alpha})) = \infty$. As diferenças são

$$\vartheta(\alpha) - \vartheta(h^{-1}(\alpha)) = 1 \quad , \quad \vartheta(\bar{\alpha}) - \vartheta(h^{-1}(\bar{\alpha})) = -\infty$$

2. Neste caso $\vartheta(\alpha_n) = n$ e $\vartheta(h^{-1}(\alpha_n)) = 1$. A diferença é

$$\vartheta(\alpha) - \vartheta(h^{-1}(\alpha)) = 1 - n$$

Com excepção de $\bar{\alpha}$, no primeiro caso, as diferenças são sempre positivas o que nos indica que os *outputs* são mais incertos que os *inputs* respectivos.

Estas observações motivam a seguinte definição:

DEFINIÇÃO 11 *Seja $\mathcal{E} \subseteq \mathcal{L}$ uma classe de eventos não nulos ($\mu(\alpha) > 0$ para todo $\alpha \in \mathcal{E}$). Uma função mensurável $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$ é uma* **compressão** *de $\mathcal{E}$ se existe uma constante $c > 0$ tal que, para todo o evento $\alpha \in \mathcal{E}$,*

$$\vartheta(\alpha) - \vartheta(h^{-1}(\alpha)) \ \geq \ c \tag{37}$$

A função h é uma compressão de $\xi \subseteq \mathbb{B}^\infty$, não-vazio, se **(??)** *for válido para todo $\alpha \supseteq \xi$.*

Exemplo 17 : Tome-se, como a classe dos eventos, a família de todos os $\uparrow u$ quando u percorre todas as strings finitas.

$$\mathcal{E} \ = \ \{ \uparrow u \mid u \in \mathbb{B}^* \}$$

Vamos supor, também, que existe uma função h tal que todos os eventos $h^{-1}(\alpha)$, com $\alpha = \uparrow u$, são também eventos na mesma classe $\mathcal{E}$. Neste exemplo h^{-1} mapeia eventos na classe $\mathcal{E}$ em eventos na mesma classe. Desa forma. para toda a string finita u, existe uma string finita $\nu(u)$ tal que

$$h^{-1}(\uparrow u) = \uparrow \nu(u)$$

Com esta hipótese, as propriedades probabilísticas de uma função sobre strings infinitas $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$ ficam determinadas por uma função $\nu : \mathbb{B}^* \to \mathbb{B}^*$ sobre strings finitas. Para eventos na classe $\mathcal{E}$ tem-se

$$\mu(\uparrow u) = 2^{-|u|} \quad \text{e} \quad \mu(h^{-1}(\uparrow u)) = 2^{-|\nu(u)|}$$

Note-se que

$$\vartheta(\uparrow u) = |u| \quad \text{e} \quad \vartheta(h^{-1}(\uparrow u)) = |\nu(u)|$$

A função h será uma compressão da classe de eventos $\mathcal{E}$ se $\vartheta((h^{-1}(\uparrow u)) \leq \vartheta(\uparrow u) - c$, para algum $c > 0$ e todo $u \in \mathbb{B}^*$. Ou seja, quando for,

$$|\nu(u)| \leq |u| - c$$

Isto é, a função $\nu(\cdot)$ comprime qualquer argumento u em, pelo menos, c bits.

É esta a noção que a definição **??** extende ao caso mais geral de uma classe de eventos arbitrária e uma função $h(\cdot)$ que se exige apenas que seja mensurável.

A outra versão de compressão diz respeito a eventos. Tome-se, por exemplo, o evento formado por uma única string infinita ω

$$\xi = \{\omega\}$$

Será possível que uma função h, como vimos atrás, seja uma compressão desse evento?

Os eventos $\uparrow u$ que contêm ξ são aqueles em que $u \leq \omega$ se verifica; portanto h é uma compressão de ξ se:

(i) $h^{-1}(\uparrow u) = \uparrow \nu(u)$ para algum $\nu : \mathbb{B}^* \to \mathbb{B}^*$
esta condição exige que dois argumentos de $h(\cdot)$ que coincidam no prefixo $\nu(u)$ dêm origem a resultados que coincidem no prefixo u,

(ii) $|v(u)| \leq |u| - c$
pode-se ver $\nu(u)$ como o "programa", para a máquina $h(\cdot)$, que produz o resultado u; esta condição exige que o "programa" seja mais curto do que o *output* em, pelo menos, c *bits*.

□

Normalmente os problemas em Criptografia exprimem-se por funções sobre domínios finitas (mesmo que grandes) ou, quanto muito, contáveis. Genericamente lidamos com funções da forma $f : \mathbb{B}^n \longrightarrow \mathbb{B}^m$ ou da forma $f : \mathbb{B}^* \longrightarrow \mathbb{B}^*$.

Pode-se reduzir qualquer uma destas hipóteses a uma função "inteira" $f : \mathbb{N} \longrightarrow \mathbb{N}$.

Como será possível descrever estas funções probabilisticamente e, nomeadamente, como será possível representar um tal f por uma função $h : \mathbb{B}^\infty \longrightarrow \mathbb{B}^\infty$?

Em primeiro lugar é necessário analisar a forma de representar probabilisticamente (por strings infinitas) os naturais. Para isso usa-se o conceito de *codificacao*

DEFINIÇÃO 12 *Uma* **codificação** *de* $\mathbb{N}$, $X = \{ X_n \in \mathcal{L} \mid n \in \mathbb{N} \}$, *é uma partição contável de* $\mathbb{B}^\infty$ *por eventos não-nulos.*

Notas

1. Em primeiro lugar X é uma partição; isto significa que é uma família de conjuntos disjuntos que cobrem todo o universo de possibilidades $\mathbb{B}^\infty$.
   Em seguida deve-se ter em atenção que todos os elementos da partição são eventos. Nomeadamente estão definidas as várias probabilidades $\mu(X_n)$.
2. A interpretação de X como código assenta na ideia que cada X_n é o evento formado por todas as possibiliades ω que codificam o inteiro n; isto é,

   $$\omega \text{ codifica } n \quad \text{sse} \quad w \in X_n$$

   Assim:
   (a) Como os eventos X_n são disjuntos não pode existir nenhuma possibilidade ω que codifique dois naturais diferentes,
   (b) Como os X_n são não-nulos, cada natural n é codifcado por um número não contável de strings,

(c) Como os X_n cobrem $\mathbb{B}^\infty$, cada possibilide $\omega \in \mathbb{B}^\infty$ codifica um e só um número natural n

Faz sentido falar da **entropia da codificação** X determinada da forma definida em (**??**), isto é

$$H(X) \;=\; \sum_n \mu(X_n)\,\vartheta(X_n) \tag{38}$$

Nota

Qualquer codificação dos naturais $X = \{\, X_n \in \mathcal{L} \mid n \in \mathbb{N} \,\}$ determina uma função $\xi : \mathbb{B}^\infty \longrightarrow \mathbb{N}$ tal que

$$X_n \;=\; \xi^{-1}(n) \tag{39}$$

A função ξ está bem definida porque os X_n são disjuntos e cobrem todo o domínio $\mathbb{B}^\infty$. Assim, $\xi(\omega)$ tem como resultado o único inteiro n que contém ω.

Desta forma pode-se ver ξ como uma representação da codificação. Note-se que a função é fortemente mensurável porque todos os $\xi^{-1}(x)$ são, por hipótese, eventos não-nulos.

Inversamente, dada uma qualquer função ξ, a família $\{\xi^{-1}(n)\}_{n\in\mathbb{N}}$ é uma partição de $\mathbb{B}^\infty$. Se se exigir que todos os $\xi^{-1}(n)$ sejam não nulos, obtém-se uma codificação de $\mathbb{N}$.

Isso é equivalente à condição de que ξ é fortemente mensurável (ver definição **??**).

Por isso faz sentido afirmar o seguinte facto.

FACTO 1 *Cada codificação dos naturais* $X = \{\, X_n \in \mathcal{L} \mid n \in \mathbb{N} \,\}$ *é univocamente determinada pela função fortemente mensurável* $\xi : \mathbb{B}^\infty \longrightarrow \mathbb{N}$ *que verifica* (**??**).

Inversamente cada função $\xi : \mathbb{B}^\infty \longrightarrow \mathbb{N}$ *fortemente mensurável determina uma codificação dos naturais.*

Este resultado indica-nos que a noção que função enumerável fortemente mensurável é equivalente à noção de codificação.

□

A representação das funções inteiras pode ser feita de duas formas: directa e inversa

DEFINIÇÃO 13 *Seja* $f : \mathbb{N} \to \mathbb{N}$ *uma função inteira e tomemos por referência uma codificação dos naturais* $\{X_n\}_{n\in\mathbb{N}}$.

A função $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$ **descreve** f *se, para todo* $n \in \mathbb{N}$,

$$X_n \subseteq h^{-1}(X_{f(n)}) \tag{40}$$

e diz-se que h **inverte** f *se, para todo* $n \in \mathbb{N}$,

$$h^{-1}(X_n) \subseteq X_{f(n)} \tag{41}$$

Notas

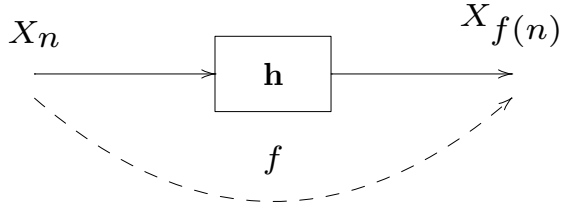
A codificação dos naturais X_n pode também ser vista como uma sequência de propriedades que se observam nos *ports* de h.

O facto de serem disjuntas diz-nos que nunca podem ocorrer, simultaneamente e no mesmo *port*, duas propriedades distintas. Observando o *port* e identificando a propriedade X_n que aí ocorre, pode-se concluir que esse port está a representar o inteiro n.

Quando h descreve f observa-se no *input* de h o evento X_n, que codifica o argumento n da função, e observa-se no *output* $X_{f(n)}$ o evento que codifica o resultado respectivo $f(n)$.

A condição (**??**) traduz a correcção dessa descrição

$$X_n \subseteq h^{-1}(X_{f(n)})$$



e diz-nos essencialmente que toda a possibilidade ω deve verificar

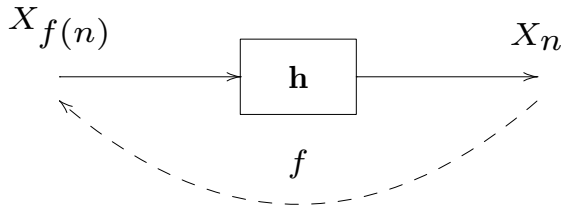
$$\omega \text{ codifica } n \implies h(\omega) \text{ codifica } f(n)$$

□

Quando h inverte f observa-se no *input* de h propriedades do resultado $f(n)$ através da codificação $X_{f(n)}$; deve-se observar no *output* de h a propriedade X_n que codifica o argumento n que deu origem a esse resultado.

Agora é a condição (**??**) que traduz a correcção desta implementação

$$h^{-1}(X_n) \subseteq X_{f(n)}$$



O que esta condição traduz é que toda a possibilidade ω deve verificar o seguinte

$$h(w) \text{ codifica } n \implies \omega \text{ codifica } f(n)$$

A noção de codificação dos naturais estende-se naturalmente para qualquer conjunto enumarável X (nomeadamente o conjunto das strings finitas $\mathbb{B}^*$).

DEFINIÇÃO 14 *Seja X um qualquer domínio contável. Uma família $\{\alpha_x\}_{x\in X}$ de eventos não-nulos é uma* **codificação** *de X se for uma partição do universo das possibilidades $\mathbb{B}^\infty$.*

Notas

1. Note-se que todos os α_x são eventos em $\mathcal{L}$ não-nulos: i.e. $\mu(\alpha_x) > 0$.
2. Porque a família é uma partição de $\mathcal{L}$ deve ser $\alpha_x \cap \alpha_y = \emptyset$, para $x \neq y$, e
$$\bigcup_{x \in X} \alpha_x = \mathbb{B}^\infty$$
3. Como consequência tem-se $\mu(\alpha_x \mid \alpha_y) = 0$, para todo $x \neq y$, e
$$\sum_{x \in X} \alpha_x = 1$$

As noções de descrição e inversão de funções, apresentadas na definição **??**, estendem-se naturalmente a funções $f : X \to Y$, em que X, Y são domínios contáveis.

DEFINIÇÃO 15 *Sejam X, Y domínios contáveis e sejam $\{\alpha_x\}_{x \in X}$ e $\{\beta_y\}_{y \in Y}$ codificações de cada um desses domínios.*

A função $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$ ***descreve*** *a função $f : X \to Y$ se, para todo $x \in X$,*
$$\alpha_x \subseteq h^{-1}(\beta_{f(x)})$$
e ***inverte*** *f se, para todo $x \in X$,*
$$h^{-1}(\alpha_x) \subseteq \beta_{f(x)}$$

3.2.Enumeração e Implementação

Para relacionar computações com funções mensuráveis (enumeráveis ou não) vamos começar por relacionar computações com eventos tentando ver se existe alguma forma de caracterizar um evento através do sucesso (ou insucesso) de determinadas computações.

Vamos tomar por referência uma classe de computações $\mathcal{C}$ que contém $\mathbf{1}$ e é fechada por composições (isto é, $\rho, \lambda \in \mathcal{C}$ implica $\lambda\,\rho \in \mathcal{C}$).

DEFINIÇÃO 16 *Um evento α é $\mathcal{C}$-**enumerável** se existe uma computação $\rho \in \mathcal{C}$ tal que*

$$\langle \rho \mid \alpha \rangle \quad \simeq \quad \langle \bar{\rho} \mid \bar{\alpha} \rangle \quad \simeq \quad \mathbf{1} \tag{42}$$

*A todo ρ que verifica (**??**) chamaremos uma $\mathcal{C}$-**enumeração** de α .*

Notas

Suponhamos que $\mathcal{C}$ é uma classe de computações caracterizada por certas condições de "boa" eficiência computacional. Por exemplo poderíamos supor que $\mathcal{C}$ é a classe das computações determinísticas polinomiais.

A condição $\langle \rho \mid \alpha \rangle \simeq \mathbf{1}$ pode ser interpretada da seguinte forma: sempre que ρ executa numa possibilidade que cai dentro de α então é indistinguível da decisão unitária $\mathbf{1}$; isto é, termina e com sucesso. A condição $\langle \bar{\rho} \mid \bar{\alpha} \rangle \simeq \mathbf{1}$ é interpretada da forma dual: sempre que ρ executa numa possibilidade que não cai dentro de α então é indistinguível da computação que termina mas falha. [12]

Portanto, se não soubermos qual é a possibilidade onde ρ está a executar, temos a certeza que ρ tem grande probabilidade de terminar e, se tiver sucesso, é porque a possibilidade estava em α; se falhar é porque a possibilidade não estava em α.

Vendo α como uma forma de caracterizar uma qualquer propriedade no espaço de possibilidades, o facto de α ter uma enumeração ρ diz-nos que esta computação "implementa"

[12] Recordemos que $\bar{\rho}$ termina com sucesso quando ρ termina com falha.

computacionalmente α: termina com sucesso ou com falha consoante a possibilidade satisfaz ou não a propriedade.

Exemplo 18 (testes de propriedades de naturais): Considere-se a codificação apresentada no exemplo **??** aplicada, agora, aos números naturais.

Assim, dado um qualquer número natural $m \in \mathbb{N}$, constrói-se em primeiro lugar a sua representação binária (também representada por m) e depois a string

$$\tilde{m} \doteq 0^{|m|}\, 1\, m$$

Com esta codificação cada sub-conjunto dos números naturais $X \subseteq \mathbb{N}$ pode ser codificado num evento $\tilde{X}$ definido por

$$\tilde{X} \doteq \bigcup_{m \in X} \uparrow \tilde{m} \tag{43}$$

Se $\tilde{X}$ for $\mathcal{C}$-enumerável isto significa que existe uma computação ρ em $\mathcal{C}$ capaz de reconhecer um qualquer elemento de X.

Numa qualquer possibilidade $\omega \in \tilde{X}$, o teste ρ começa por extrair o único inteiro m tal que $\omega \geq \tilde{m}$ e usa o resto da string ω como fonte de aleatoriedade.

Nomeadamente se tomarmos X como o conjunto dos números primos, então uma eventual enumeração de $\tilde{X}$ é um **teste de primalidade**.

□

Implementação

As observações anteriores conduzem naturalmente a uma noção de implementação de uma função:

DEFINIÇÃO 17 *Uma função mensurável* $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$ *é* $\mathcal{C}$-***implementável*** *quando, para todo o evento* $\mathcal{C}$*-enumerável* α*, o evento* $h^{-1}(\alpha)$ *também é* $\mathcal{C}$*-enumerável.*

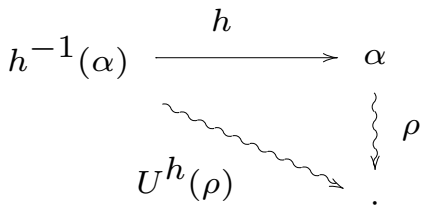
Um operador $U^h : \mathcal{C} \to \mathcal{C}$ *que mapeie uma enumeração* $\rho \in \mathcal{C}$ *de* α *numa enumeração de* $h^{-1}(\alpha)$ *designa-se por* $\mathcal{C}$-***implementação*** *de* h*.*

É obvio, pela definição, que h é $\mathcal{C}$-**implementável** se e só se existir uma $\mathcal{C}$-**implementação** de h.

Comentários

Supúnhamos que α tem a enumeração ρ.

A definição diz-nos que $h^{-1}(\alpha)$ tem também uma enumeração (que pode não ser única). Deste modo permite-nos definir uma implementação U^h (não necessariamente única) que mapeia cada enumeração ρ de um evento arbitrário α numa enumeração de $h^{-1}(\alpha)$.



Note-se que, para uma qualquer possibilidade ω,

$$\omega \in h^{-1}(\alpha) \quad \text{sse} \quad h(\omega) \in \alpha$$

Por outro lado, pela definição de enumeração,

- $\omega \in h^{-1}(\alpha)$ quando, com elevada probabilidade, $U^h(\rho)$ sucede em ω
- $h(\omega) \in \alpha$ quando, com elevada probabilidade, ρ sucede em $h(\omega)$

Portanto a relação entre as enumerações ρ e $U^h(\rho)$ pode ser expressa da seguinte forma

> *para toda a possibilidade ω, com elevada probabilidade, ρ termina com sucesso em $h(\omega)$ se e só se $U^h(\rho)$ termina com sucesso em ω.*

As computações $\mathcal{C}$ podem ser interpretadas como *testes computacionais*; enquanto que ρ é um teste que actua no *output* de h, a enumeração $U^h(\rho)$ é um teste no *input* dessa função. A definição diz que, qualquer que seja o teste ρ com que se pretenda analisar o *output*, é possível construir um outro teste que actua no *input* e que dá os "mesmos" resultados.

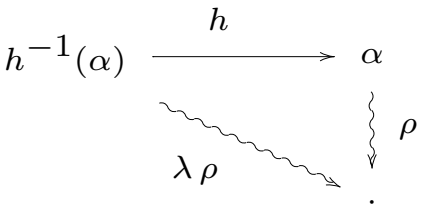
Por exemplo: suponhamos que o teste ρ consiste em determinar se os primeiros k bits do *output* $h(\omega)$ representam um número primo; então o teste $U^h(\rho)$ consegue dar a mesma resposta olhando apenas para ω.

O operador $U^h : \mathcal{C} \to \mathcal{C}$ representa, deste modo, uma implementação computacional da função h no sentido em que "dispensa" a aplicação dessa função: basta testar o *input* com $U^h(\rho)$ para se ter o mesmo resultado que se teria construindo o *output* $h(\omega)$ e testando $h(\omega)$ com ρ.

Obviamente que podem existir várias destas implementações dado que, para cada enumeração de α, podem existir várias enumerações de $h^{-1}(\alpha)$. Cada estratégia de mapeamento entre as primeiras e as segundas define um operador diferente.

Uma forma particular de operador em $\mathcal{C}$ é definido fixando uma computação $\lambda \in \mathcal{C}$ e usando o mapeamento $\rho \mapsto \lambda\,\rho$.

Se for $U^h(\rho) = \lambda\,\rho$, então a implementação é particularmente simples porque fica completamente determinada por uma única computação λ.



Só nessas circunstâncias é possível fazer a afirmação:

o algoritmo λ implementa a função h.

□

A definição **??** adapta-se facilmente a funções e relações enumeráveis.

DEFINIÇÃO 18 *Seja $f : \mathbb{B}^\infty \to \mathbb{N}$ uma função enumerável mensurável. f é $\mathcal{C}$-**implementável** quando, para todo $y \in \mathbb{N}$, o evento $f^{-1}(y)$ é $\mathcal{C}$-enumerável.*

*Uma relação mensurável $R \subseteq \mathbb{B}^\infty \times \mathbb{N}$ é $\mathcal{C}$-**implementável** quando, para todo $y \in \mathbb{N}$, o evento $R^{-1}(y)$ é $\mathcal{C}$-enumerável.*

*Um operador $U^f : \mathbb{N} \to \mathcal{C}$, que mapeia qualquer $y \in \mathbb{N}$ numa enumeração de $f^{-1}(y)$, é uma $\mathcal{C}$-**implementação** de f.*

Notas

Pela definição, uma implementação U^f de uma função $f : \mathbb{B}^\infty \to \mathbb{N}$ dá origem sequência infinita de enumerações $\phi_i = U^f(i)$ em que, cada ϕ_i enumera $f^{-1}(i)$.

Dento desta classe são ainda importantes as funções de domínio e contra-domínio contáveis.

DEFINIÇÃO 19 *Seja* $X = \{X_n\}$ *uma codificação de naturais e seja* $R \subseteq \mathbb{N} \times \mathbb{N}$ *uma relação em* $\mathbb{N}$.

Define-se, para cada $x \in \mathbb{N}$,

$$R_X^{-1}(x) \doteq \bigcup_{(n,x)\in R} X_n \tag{44}$$

Uma $\mathcal{C}$-**implementação** *de* R *é uma qualquer função* $U^R : \mathbb{N} \to \mathcal{C}$ *que associe um* $x \in \mathbb{N}$ *arbitrário a uma enumeração de* $R_X^{-1}(x)$.

3.3. Unidirecionalidade

Considere-se uma função implementável $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$. O operador U^h pode também ser usado "ao contrário".

Suponhamos que, em vez de se fixar uma propriedade no *output* de h, se fixava um teste ρ no *input* e se fazia a seguinte pergunta:

> *existe algum teste ρ' que possa ser feito no resultado $h(\omega)$ que seja "equivalente" ao teste ρ em ω ?*

Se a "equivalência" for indistinguibilidade, então a pergunta é:

> *dado um teste ρ arbitrário, existe algum teste ρ' tal que*

$$\{U^h(\rho')\} \sim \{\rho\} \tag{45}$$

Se for sempre possível encontrar um tal ρ', para um ρ arbitrário, então é possível definir uma aplicação

$$V : \mathcal{C} \to \mathcal{C} \qquad \text{com} \qquad V : \rho \mapsto \rho'$$

que, dado (**??**), verifica

$$\forall \rho \in \mathcal{C} : \quad \{U^h(V(\rho))\} \sim \{\rho\} \tag{46}$$

Nota: A aplicação $\rho \mapsto U^h(V(\rho))$ é, funcionalmente, a composição $(U^h \cdot V)$ das aplicações U^h e V. A condição (**??**) estabelece uma forma de "invertibilidade" no sentido em que asserta que essa composição é indistinguível da aplicação identidade. Isso justifica a seguinte definição.

DEFINIÇÃO 20 *Uma transformação $V(\cdot)$ que satisfaça* (**??**) *designa-se por $\mathcal{C}$-**inversa** da implementação U^h.*

*Uma função $\mathcal{C}$-implementável $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$ diz-se $\mathcal{C}$-**invertível** que alguma das suas $\mathcal{C}$-implementações tem $\mathcal{C}$-inversa.*

□

A segurança de muitas técnicas criptográficas depende, crucialmente, da existência de certas funções matemáticas $f : X \longrightarrow Y$ 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 19 : 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 $\exp_{p,a} : \mathbb{Z}_{p-1} \to \mathbb{Z}_p^*$ [13]

$$\exp_{p,a} : x \mapsto a^x \pmod{p}$$

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.

Estes exemplos apontam para funções matemáticas invertíveis da forma $F : \mathbb{N} \to \mathbb{N}$. A noção de invertibilidade, que apresentamos na definição **??** (pag. **??**), refere-se a implementações computacionais de funções da forma $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$. São, portanto, duas noções distintas.

No entanto, como vimos no exemplo **??** e no exemplo **??**, é sempre possível codificar funções inteiras em funções sobre strings infinitas e

[13] $\mathbb{Z}_n$ 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 $[1, p-1]$.

ver a "unidirecionalidade" de F como a não-existência de uma inversa computacional de qualquer das possíveis implementações de qualquer uma das possíveis codificações de F.

A noção de unidirecionalidade aplica-se também a funções $f : \mathbb{B}^{\infty} \longrightarrow \mathbb{N}$ não-invertíveis[14].

Formalmente é possível definir função unidireccional, simplesmente, como uma função implementável que não é invertível. Isto significa que não é possível computar o argumento da função a partir do conhecimento do seu resultado.

Porém, normalmente, as técnicas criptográficas cuja segurança é baseada na unidirecionalidade exigem algo mais forte: exigem que, para efeitos da observação de "propriedades reveladoras" do argumento, o resultado da função dever ser indistinguível de uma sequência aleatória.

A caracterização do que são essas "propriedades reveladoras" conduz à noção de **teste raro**; vamos conceber um teste raro como uma computação que, sem informação adicional, tem uma expectativa de sucesso muito baixa (de facto, uma expectativa de sucesso desprezável); no entanto, essa expectativa não é nula e, se o teste for condicionada a eventos apropriados, a expectativa pode ser claramente não-desprezável e até, eventualmente, perto de 1.

Tomemos como referência uma classe qualquer $\mathcal{C}$ de computações que assumimos como sendo "tratáveis". Os **testes raros** de $\mathcal{C}$ são as computações ϱ nessa classe que verificam $\{\varrho\} \sim \mathbf{0}$. Representamos por $\mathcal{C}_r$ a sub-classe de $\mathcal{C}$ formada por todos os seus testes raros.

Dependente da situação, nem todos os testes raros são obrigatoriamente "propriedades reveladoras". Por isso faz sentido especificar, para cada situação, qual é a classe dos testes raros que se consideram "propriedades reveladoras".

[14] O exemplo mais importante são as funções de "hash" discutidas numa das secções seguintes.

DEFINIÇÃO 21 *Seja* $h : \mathbb{B}^\infty \to \mathbb{B}^\infty$ *uma função implementável.*

(i) *Uma* $\mathcal{C}$*-implementação* U^h *de* h *é* **fracamente** $\mathcal{C}$**-unidireccional** *se não é* $\mathcal{C}$*-invertível.*

(ii) *Seja* $\mathcal{S} \subseteq \mathcal{C}_r$ *uma classe de testes raros. Uma* $\mathcal{C}$*-implementação* U^h *de* h *é* **fortemente** $\mathcal{C}$**-unidireccional** *para a classe* $\mathcal{S}$*, se, para qualquer teste raro* $\varrho \in \mathcal{S}$ *e qualquer outro teste* $\phi \in \mathcal{C}$*, verifica-se*

$$\{\varrho \mid U^h(\phi)\} \sim \mathbf{0} \tag{47}$$

h *é* **fracamente** *(resp.* **fortemente***)* $\mathcal{C}$**-unidireccional** *quando toda a sua implementação é fracamente (resp. fortemente)* $\mathcal{C}$*-unidireccional.*

Notas

1. É óbvio que existem testes ϱ que verificam $\{\varrho \mid U^h(\phi)\} \not\sim \mathbf{0}$. Nomeadamente, fazendo $\varrho = U^h(\phi)$ tem-se $\{\varrho \mid U^h(\phi)\} \sim \mathbf{1}$.
   Porém um tal ϱ não é, normalmente, nenhuma das "propriedades reveladoras". Se for então, claramente, a implementação U^h não é unidireccional.
2. Por vezes não é necessário que h seja "totalmente unidireccional" mas o seja apenas no sentido em que, para efeitos de avaliar os resultados de uma outra função f, os resultados de h se devem comportar como strings aleatórios. Uma tal condição obtém-se através de uma (aparente!) "generalização" de (**??**).
   Considere-se duas funções implementáveis h, f e duas implementações U^h , U^f destas funções. Se, para todo o teste raro $\varrho \in \mathcal{C}_r$ e todo o teste $\phi \in \mathcal{C}$, se verifica

   $$\{U^f(\varrho) \mid U^h(\phi)\} \sim \mathbf{0} \tag{48}$$

   então U^h é $\mathcal{C}$-unidireccional relativo a U^f. Se (**??**) se verifica para todas as implementações U^h e U^f então h é fortemente unidireccional relativo a f.
   Essencialmente o que se está estabelecer, nesta definição, é uma classe $\mathcal{S}$ de propriedades reveladoras da forma

   $$\mathcal{S} \doteq \{\, U^f(\varrho) \mid \varrho \in \mathcal{C}_r \,\}$$

   Neste ponto de vista, e com um tal $\mathcal{S}$, (**??**) é, apenas, uma instância particular da definição mais geral (**??**).

□

A noção de unidirecionalidade em funções ou relações enumeráveis exprime-se de forma algo diferente. Para um domínio X contável, dado

$$f : \mathbb{B}^\infty \to X \qquad \text{ou} \qquad f \in \mathbb{B}^\infty \times X$$

procura-se que a propriedade traduza o facto de o conhecimento de um qualquer x não deve fornecer qualquer informação útil sobre $f^{-1}(x)$. Para todos os testes computacionais, $f^{-1}(x)$ deve-se comportar como um evento aleatório[15].

Tomamos por referência um comjunto de computações $\mathcal{C}$ e o respectivo conjunto de testes raros $\mathcal{C}_r$

DEFINIÇÃO 22 *Seja $f : \mathbb{B}^\infty \to X$ (ou, $f \in \mathbb{B}^\infty \times X$) uma função (respectivamente, relação) $\mathcal{C}$-implementável.*

*Uma implementação $U^f : X \to \mathcal{C}$ de f é $\mathcal{C}$-**unidireccional** se, para todo $\varrho \in \mathcal{C}_r$ e todo $x \in X$,*

$$\{\varrho \mid U^f(x)\} \sim \mathbf{0} \tag{49}$$

*A função f (respectivamente, relação) é $\mathcal{C}$-**unidireccional** quando toda a sua implementação é $\mathcal{C}$-unidireccional.*

Sopunhamos que $f : \mathbb{B}^\infty \to X$ é $\mathcal{C}$-unidireccional. Se f for fortemente mensurável, então f determina uma codificação de X. Uma tal codificação de X designa-se, obviamente, por **codificação unidireccional**.

DEFINIÇÃO 23 *Um subconjunto $Y \subseteq X$ é $\mathcal{C}$-**difícil** se, para qualquer codificação unidireccional $f : \mathbb{B}^\infty \to X$ e qualquer $\mathcal{C}$-implementação U^f, toda a eventual enumeração de $\bigcup_{x \in Y} U^f(x)$ é um teste raro.*

[15] *Veremos adiante o que "aleatório" significa.*

Notas

A codificação unidireccional procura descrever a ideia de que uma possibiliade arbitrária ω codifica um e só um elemento $x \in X$ mas, a partir do resultado x, não é possível recuperar nenhuma possibilidade ω que lhe tenha dado origem.

Os subconjuntos Y difíceis em X descrevem problemas de procura típicos; se Y for o conjunto característico de um qualquer predicado $p : X \longrightarrow \mathbb{B}$, então Y representa o problema

$$gerar\ x \in X\ tal\ que\ p(x)$$

Sendo Y difícil está-se a afirmar que as possibilidades codificáveis em elemetos x que verifiquem $p(x)$, se for enumerável, é-o por um teste raro; isto é, uma computação com expectativa de susesso desprezável.

3.4.Permutações e Cifras

Tomemos uma vez mais, como referência, uma classe $\mathcal{C}$ de computações assumidamente tratáveis.

Uma noção fundamental, para caracterizar funções da forma $f : \mathbb{B}^* \to \mathbb{B}^*$ reside na sua capacidade de **preservação da informação**: informalmente, f obedece a esta propriedade quando o nível de incerteza no argumento é preservado no resultado.

A preservação de informação é indispensável à noção de *cifra*.

Alguma notação prévia:

1. Seja $F = \{f_i\}_{i\in\mathbb{N}}$ uma família de permutações unidireccionais.
2. Dadas $\mathcal{C}$-implementações arbitrárias U^{f_i} dos f_i e dado $\rho \in \mathcal{C}$, seja

$$U^F(\rho) \doteq \{U^{f_i}(\rho)\}_{i\in\mathbb{N}}$$

   Diremos que U^F é uma $\mathcal{C}$-**implementação** de F.

DEFINIÇÃO 24 *F é uma $\mathcal{C}$-**cifra** se, para toda a implementação U^F e teste raro $\rho \in \mathcal{C}_r$*

$$\varphi_i \simeq_{\mathcal{C}} \varphi_i \,\|\, \varphi_j \qquad \forall\, i,j \in \mathbb{N} \tag{50}$$

sendo definido $\varphi_i \doteq U^F(\rho)_i$.

Notas

1. Em primeiro lugar a cifra F é constituída por permutações unidireccionais; isto é, cada uma delas preserva informação e nenhuma informação pode ser extraída, do resultado da função, sobre o seu argumento.
   Na linguagem comum das cifras isto é equivalente à condição "não-invertibilidade do criptograma".

2. Um outro aspecto é que cada elemento de F é identificado por um índice i, índice esse que se designa normalmente por **chave**. A condição (**??**) traduz aquilo que, em linguagem comum, se designa por "não-invertibilidade da chave".

   Essencialmente a condição fixa uma implementação arbitrária U^F da cifra (a condição deve ser válida para todas) e um teste raro arbitrário ρ que actua sobre os *outputs* dos f_i. As computações $\varphi_i = U^F(\rho)_i$ é a computação que implementa a função f_i e, em seguida, efectua o teste ρ sobre o resultado.

   Em seguida comparam-se duas computações: a computação φ_i (por si só) e a escolha $\varphi_i \| \varphi_j$ entre essa computação e uma outra φ_j dentro da mesma família.
   De acordo com a definição de indistinguibilidade (definição **??** na página **??**)) estas computações são inseridas num mesmo padrão arbitrário e as duas computações resultantes têm expectativas de sucesso indistinguíveis.

   Isto sigifica que não há nenhuma computação que, recebendo φ_i como oráculo ou em alternativa $\varphi_i \| \varphi_j$, consiga distinguir as duas situações.

3.5.Geradores Aleatórios e Pseudo-Aleatórios

Se existisse um concurso para eleger o conceito matemático mais "fugidio" a uma análise substantiva, então a noção de **aleatoriedade** estaria entre os primeiros candidatos.

Nada parece ser tão simples de intuir e, simultaneamente, tão complicado de descrever com rigor.

Aparentemente nada poderia parecer mais simples: a aleatoriedade pode ser perfeitamente descrita pela noção matemática de probabilidade. Porém, logo que se pensou usar computadores para gerar artificialmente comportamentos que simulassem características dos sistemas ditos "naturais", rapidamente foi claro que a necessidade para rigor era premente.

Exemplo 20 : Um programa que faça previsões meteorológicas tem de simular o comportamento natural de inúmeras quantidades físicas que afectam a meteorologia do planeta.

A construção de um tal programa tem, essencialmente, objectivos contraditórios: por um lado as quantidades do mundo físico que afectam a meteorologia (temperaturas, ventos, correntes, etc. . .) apresentam um elevado grau de incerteza; por outro lado o programa deve funcionar, geralmente sem incerteza.

Esta "dupla face" (certeza *vs* aparente incerteza) para que um tal programa exige uma descrição bastante precisa destes conceitos. Não basta dizer, simplesmente, que o programa apresenta um comportamento aleatório ou não-aleatório.

Algoritmos que, como programa do exemplo anterior, simulam a incerteza tomam o nome genérico de **geradores pseudo-aleatórios** (GPA's) ou PRG's (da designação *pseudo-random generators*). PRG's são amplamente utilizados não só em programas de simulação (como nesse exemplo) mas também em problemas de optimização e (e isso interessa-nos particularmente) em criptografia.

A aleatoriedade é uma componente essencial dos sistemas de informação. De facto toda a Teoria da Informação de Shannon lida com a informação como "quebra de incerteza".

Em Criptografia a aleatoriedade tem também esta interpretação: a forma como uma comunicação se protege de "ataques de adversários" escondendo os seus segredos no meio da incerteza. Por isso a **segurança** de muitos sistemas criptográficos depende crucialmente do modo como lidam com a aleatoriedade.

A Teoria da Informação só por si não é suficiente: os nosso programas exigem que se analise também aos aspectos de computabilidade e complexidade.

No capítulo 1 reconhece-se o papel crucial da aleatoriedade na definição de computação e no estudo da complexidade: toda a nossa análise assenta no conceito de computação aleatória onde significado e comparação assentam em noções onde a aleatoriedade é fulcral.

Todos os algoritmos que necessitamos assentam nestas duas componentes: por um lado os aspectos de Teoria da Informação; por outro os aspectos ligados à Computabilidade e à Complexidade. Ambos necessitam de uma visão rigorosa da aleatoriedade.

O estudo da aleatoriedade baseia-se em dois problemas essenciais:

Teste de aleatoriedade:
quão "aleatória" é uma sequência finita de bits $\omega \in \mathbb{B}^*$ *ou uma sequência infinita de bits* $\Omega \in \mathbb{B}^\infty$ *?*

Geração de sequências aleatórias de bits:
definir um algoritmo determinístico e computacionalmente eficiente para gerar sequências "aleatórias" $\omega \in \mathbb{B}^*$ *de comprimento pré-determinado.*

Na era em que as Ciências da Computação adquiriram credibilidade científica, várias abordagens têm sido tentadas para obter um conceito rigoroso de aleatoriedade que conduzem a "soluções" para estes problemas:

Aleatoriedade e Testes estocásticos

Nesta abordagem caracteriza-se sequências aleatórias de *bits* através da forma como verificam (ou não) certas propriedades estocásticas.

Martin-Löf, Solovay e outros desenvolveram uma abordagem à aleatoriedade assente na noção de propriedade computável e *rara*: isto é, propriedades que só se verificam num número muito pequeno de possibilidades (a probabilidade de se verificar numa string aleatória deve ser desprezável).

Segundo Martin-Löf toda a string que verifique alguma destas propriedades, não pode ser aleatória. Dito de outra forma, é aleatória uma string que não verifica nenhuma propriedade computável e rara.

Aleatoriedade vis. Imprevisibilidade

Nesta abordagem, assente na Teoria da Informação de Shannon, aleatoriedade é associada à ausência de informação: as sequências aleatórias de *bits* são vistas como mensagens em que qualquer receptor, que só tenha conhecimento de parte da mensagem, não consegue reconstruir a sua totalidade.

Na abordagem de Shannon a incerteza sobre o conteúdo da mensagem aleatória não desaparece se só for recebida parte dessa mensagem.

Aleatoriedade vis Incompressibilidade

Esta abordagem é baseada na noção de **complexidade de Kolmogorov** que mede a quantidade de incerteza numa string pelo tamanho do menor programa que é capaz de gerar essa string numa máquina de referência.

Nessa perspectiva não pode ser considerada aleatória uma string que possa ser gerada por um programa mais curto do que ela própria. Desta forma são aleatórias as strings que não podem ser comprimidas.

Pseudo-Aleatoriedade

Nas aplicações correntes é mais importante saber gerar sequências de *bits* que "aparentem ser aleatórias" do que testar a aleatoriedade dessas sequências.

Os geradores pseudo-aleatórios "alongam" a incerteza de uma qualquer fonte aleatória: a partir de uma curta sequência, que se assume aleatória e que se chama **"seed"** (*semente*), é construída uma sequência bastante mais longa que tem um comportamento indistinguível de uma *sequência*

verdadeiramente aleatória.

□

O nosso objectivo é, agora, tentar formalizar estas noções de aleatoriedade.

Aleatoriedade e testes raros

Historicamente procurar a aleatoriedade no sucesso ou falha de um certo número de testes foi a abordagem mais comum ao tratamento empírico da aleatoriedade.

Por exemplo, é natural pensar-se que uma sequência suficientemente grande deve ter, aproximadamente, tantos 0's quantos 1's. Este exemplo pode ser estendido e é usual verificar-se empiricamente a aleatoriedade de uma sequência finita de *bits* através de um conjunto de testes estatísticos que medem o grau de proximidade dessa sequência a uma sequência ideal[16].

Neste estudo vamos caracterizar as propriedades estocásticas por uma determinada classe de computações que actuam num espaço de possibilidades cujos elementos se quer analisar em termos de aleatoriedade.

Concretamente o nosso objectivo é analisar a aleatoriedade de eventos $\alpha \in \mathcal{L}$ através de uma família de computações determinadas por tais eventos. Seguindo a abordagem de Martin-Löf consideraremos aleatório qualquer evento cuja ocorrência não afecte o comportamento de propriedades "raras".

Como foi atrás referido, só faz sentido analisar a aleatoriedade de eventos raros (de probabilidade nula).

□

[16] Em *The Art of Computer Programming* (Vol.2) DONAL KNUTH apresenta um extenso estudo sobre o teste e geração de sequências aleatórias avaliadas por testes estatísticos. Estes testes medem de que forma uma sequência de bits, que pode ou não ser aleatória, se "aproxima" de uma distribuição probabilística ideal de bits.

Para lidar com a noção de "computação tratável" vamos tomar por referência uma classe $\mathcal{C}$ de computações recursivas geradas a partir de um conjunto de computações atómicas $\mathcal{A}$ (que se assume implicitamente).

Neste contexto,

DEFINIÇÃO 25 *Um evento ξ diz-se $\mathcal{C}$-ML-**aleatório** se, para todo o teste raro $\varrho \in \mathcal{C}$ (i.e. $\{\varrho\} \sim \mathbf{0}$), se verificar ainda $\{\varrho \mid \xi\} \sim \mathbf{0}$.*

Essencialmente a definição exige que todo o teste raro em $\mathcal{C}$ continue a ter sucesso desprezável mesmo que esteja condicionado ao evento ξ.

Notas

1. Modelamos "propriedades computáveis" parametrizando a definição por uma classe $\mathcal{C}$ de computações. A escolha apropriada desta classe determina diferentes definições de aleatoriedade.
   É usual considerar que $\mathcal{C}$ são as computações determinísticas polinomiais. Isto significa os testes estão limitados aos algoritmos recursivos determinísticos a partir das computações atómicas em $\mathcal{A}$ com complexidade polinomial. No entanto nada nos impede de estender (por exemplo, para computações não determinísticas ou não polinomiais) ou restringir a classe dos testes e, desta forma, adaptar a definição a várias situações concretas.
2. A definição **??** categoriza como aleatórios os eventos α que, para toda a propriedade rara ρ, verificam também $\{\rho \mid \xi\} \sim \mathbf{0}$.
   Esta condição força a que a expectativa condicional $\{\rho \mid \xi\}(n)$ também decresça com n mais rapidamente que qualquer função racional.
   Note-se que, se fosse possível construir uma propriedade $\rho \in \mathcal{C}$ que tivesse sucesso em todas as possibilidades $\omega \in \xi$ teríamos $\{\rho \mid \xi\} \sim \mathbf{1}$.
   Para qualquer computação, a relação

$$\{\rho \mid \xi\} \sim \{\rho\}$$

   indica a independência entre o sucesso da computação ρ e a condição α.
   Por isso a ideia $\mathcal{C}$-aleatoriedade determinada pela definição **??** assenta no grau de independência de ξ em relação ao sucesso de todas as computações raras $\rho \in \mathcal{C}$.

Para formalizar a aleatoriedade de uma string infinita $\omega \in \mathbb{B}^{\infty}$ usa-se a definição **??** com o conjunto singular formada só por essa string; isto é, $\xi = \{\omega\}$.

Para formalizar a aleatoriedade de uma string finita $u \in \mathbb{B}^{*}$ usa-se o respectivo conjunto superior: $\xi = \uparrow u$.

Aleatoriedade e Imprevisibilidade

Seja $\mathcal{C}$ uma classe de computações de referência.

Pretende-se formalizar a ideia de que qualquer teste tratável sobre uma string aleatória não pode "ser antecipado" com um outro teste sobre a mesma string que use menos informação.

DEFINIÇÃO 26 *Seja $l > 0$ um inteiro; o evento ξ é C-l-**previsível** se, para todo o teste $\phi \in \mathcal{C}$, existe um teste $\rho \in \mathcal{C}$ tal que a função de n*

$$n \quad \mapsto \quad |\{\phi \mid \xi\}(n) - \{\rho \mid \xi\}(n-l)| \tag{51}$$

é desprezável.

*O evento ξ é C-l-**aleatório** se não é C-l-**previsível**.*

Notas

À medida que se obtém mais informação (i.e quando n cresce) o grau de confiança sobre o grau de sucesso da computação ϕ vai aumentando. O que a definição nos diz é que não existe diferença assinalável, que seja detectável no sucesso de ϕ conhecendo ξ com n *bits* ou no sucesso de ρ conhecendo o mesmo evento com $n - l$ *bits*. Isto significa que ρ consegue antecipar em l *bits* o grau de sucesso de ϕ.

Aleatoriedade e Incompressibilidade

Como é usual vamos tomar uma classe de computações $\mathcal{C}$ como referência e, nesse contexto, vamos tentar formalizar a aleatoriedade de um evento ξ.

Começamos por definir a classe de todos os eventos de probabilidade não nula que cobrem ξ

$$\mathcal{E}_\xi \doteq \{\, \alpha \mid \xi \subseteq \alpha \ \wedge \ \alpha \in \mathcal{L}_n \text{ para algum } n \,\} \qquad (52)$$

DEFINIÇÃO 27 *ξ é $\mathcal{C}$-**compressível** se existe uma compressão de $\mathcal{E}_\xi$ que é $\mathcal{C}$-implementável.*

*ξ é $\mathcal{C}$-K-**aleatório** se não for $\mathcal{C}$-compressível.*

Notas

A designação K-aleatoriedade vem da ideia original baseada na complexidade de **K**olmogorov.

A intuição baseia-se na ideia de um string compressível pode ser determinada por um "programa" mais curto do que a própria string. Se a string for aleatória não deve existir nenhuma forma de a gerar para além de a reproduzir completamente.

Por exemplo, a string infinita 1 0 1 0 1 0 1 0 . . . . pode ser gerada pelo "programa" 1 0 por repetição infinita. Claramente que não pode ser considerada

Pseudo-Aleatoriedade

Um gerador pseudo-aleatório (GPA) é, antes de mais, uma computação tratável. Por isso faz sentido especificar qual é a classe das funções que são consideradas tratáveis.

Vamos, assim, tomar por referência uma classe de computações $\mathcal{C}$.

Um GPA é caracterizado pela forma como extende, ou alonga, a natureza aleatória da "semente" para strings de maiores comprimentos. Para formalizar este aspecto vamos considerar uma função inteira $h : \mathbb{N} \longrightarrow \mathbb{N}$ que é *fortemente crescente* no sentido em que o resultado $h(n)$ é bastante maior do que o argumento n ($h(n) \gg n$ para todo $n \in \mathbb{N}$).

Nestas circunstâncias

DEFINIÇÃO 28 *Um gerador* $\mathcal{C}$-**pseudo-aleatório** *para o* **função de alongamento** l *é uma função* $\lambda : \mathbb{B}^* \longrightarrow \mathbb{B}^*$ *tal que, para todo* $\rho \in \mathcal{C}$, *a função*

$$|\{U^{\lambda}(\rho)\}(n) - \{\rho\}(l(n))| \tag{53}$$

é uma função de n *desprezável, para toda a implementação* U^{λ} *de* λ.

Notas

A definição pretende traduzir a seguinte ideia: uma implementação de λ avaliada num teste ρ, vendo apenas n *bits*, deve ser equivalente a testar ρ directamente sobre a fonte de aleatoriedade mas conhecendo um comprimento $l(n)$ (que é muito maior do que n).

O valor n representa, nesta definição, o tamanho da semente enquanto que $l(n)$ representa o tamanho da string gerada.

Existe uma relação estreita entre geradores pseudo-aleatórios e funções unidireccionais que pode ser observada no exemplo seguinte que representa numa implementação comum de geradores pseudo-aleatórios.

Exemplo 21 : Seja $f : \mathbb{B}^k \longrightarrow \mathbb{B}^k$ uma função unidireccional que actua num espaço de estados de dimensão k; seja $u : \mathbb{B}^k \longrightarrow \mathbb{B}$ um **teste justo**: isto é, uma função booleana com a propriedade de dar, com igual expectativa, um resultado 1 ou 0. [17]

Estado: $\mathbf{s}$ *com valores em* $\mathbb{B}^k$

```
RANDOM(x)
   s ← x                          # "seed" x inicializa o estado s
   for t times                    #          parâmetros
      s ← f(s)                    #  k – dimensão do espaço de estados
      write u(s)                  #  t – tamanho do output em bits
```

Algoritmo 4: De funções unidireccionais para GPA's

[17] No sentido em que, se s está uniformemente distribuído no espaço $\mathbb{B}^k$, então $u^{-1}(1)$ e $u^{-1}(0)$ têm ambos probabilidade $1/2$.

3.6.Funções de Hash

Um dos mais versáteis conceitos criptográficos é do **função *hash*** que são, essencialmente, funções $H : \mathbb{B}^* \longrightarrow \mathbb{B}^t$ que transformam strings de *bits* de comprimento arbitrário em strings de *bits* de comprimento fixo, que têm uma implementação computacional eficiente e que satisfazem certos requisitos de segurança que veremos em seguida.

As funções de *hash* modernas usam um comprimento de *output* de 160 ou 256 *bits*[18] O texto, como dissemos, tem comprimento arbitrário e é usual lidar-se com texto com dezenas de MegaBytes.

O uso típico dessas funções é o de construir um **representante** $H(x)$ para um texto arbitrário x, que tenha comprimento fixo e que, de alguma forma, identifique x.

Dado que o domínio da função tem uma cardinalidade muito superior ao contra-domínio[19], a função $H(\cdot)$ não pode ser injectiva: existem sempre pares de textos distintos $(x \neq x')$ com o mesmo representante $(H(x) = H(x'))$.

A existência desses pares não significa que seja computacionalmente viável detectá-los. Quando tal for possível, diz-se que (x, x') forma uma **colisão**. Parece óbvio que, para que uma função de *hash* seja útil na função de construir "representantes", não podem existir colisões.

Desta forma as **condições de segurança** das funções de *hash* procuram caracterizar a inviabilidade de colisões. Elas podem assumir várias formas; nomeadamente

1. **unidirecionalidade**
   Dado um qualquer valor de *hash*, $z \in \mathbb{B}^t$, deve ser intratável computar um qualquer texto $x \in \mathbb{B}^*$ tal que $z = H(x)$.

[18] Ainda restam alguns usos para as funções de *hash* de 128 bits.

[19] Por muito grande que seja t, o contra-domínio $\mathbb{B}^t$ é finito; enquanto que o domínio $\mathbb{B}^*$ é infinito contável.

2. **inexistência do 2º representado**
   Dado um qualquer texto x, deve ser intratável computar um outro texto y, com $x \neq y$, tal que $H(y) = H(x)$.
3. **inexistência de colisões**
   Deve ser intratável encontrar um par de textos distintos $y \neq x$ tais que $H(y) = H(x)$.

Notas

1. A diferença essencial entre **1** e **2** reside no facto de, em **2**, as imagens da função estarem restritos a *hashs* de textos conhecidos, enquanto que em **1** a imagem é livre. A *unidirecionalidade* é, assim, uma condição mais forte do que a *inexistência do 2º representado* dado que, se a primeira não for violada, necessariamente a segunda também não o será.

   A diferença essencial entre as condições **2** e **3** reside no facto de, em **2** um dos textos é fixo enquanto que ambos os textos são livres em **3**. A *"inexistência de colisões"* é uma condição mais fraca do que a *"inexistência do 2º representado"* dado que uma violação de **2** implica necessariamente uma violação de **3** mas o inverso não é verdade.

   Portanto, dentro destas três condições de segurança, a ordem com que ocorrem reflecte a sua exigência relativa.

2. A autenticação de textos é uma decisão (uma *prova de autenticidade*) que, normalmente, não incide directamente sobre o texto x mas sim sobre o seu *hash* $H(x)$.

   Um atacante que, dada uma prova feita inicialmente para um texto x, a queira reutilizar num texto fraudulento, basta-lhe encontrar y que tenha o mesmo *hash* que x. Esta fraude baseia-se numa violação da *"inexistência do 2º representante"*.

   O atacante pode querer, simplesmente, desacreditar a prova de autenticação. Nesse caso pode escolher um 1º texto de acordo com as suas conveniências. Assim basta-lhe computar uma colisão qualquer que ela seja; por isso, basta-lhe violar a terceira regra de segurança sem que tal exija uma violação da segunda regra.

   A necessidade da condição mais forte de todas (a *unidirecionalidade*) surge da existência de alguns ataques mais subtis que se baseiam nas propriedades algébricas dos processos de construção de provas de autenticidade; a tempo voltaremos a fazer referência a esta situação.

Funções de *hash* e funções unidireccionais

Existe uma ligação óbvia entre funções de *hash* e funções unidireccionais que pode ser ilustrada no seguinte algoritmo que constrói uma função de *hash* $H_x : \mathbb{B}^* \longrightarrow \mathbb{B}^t$ parametrizada por um qualquer $x \in \mathbb{B}^k$.

Vamos considerar:

(i) Um "espaço de estados" $\mathbb{B}^k$, com $k > t$, e uma função unidireccional $f : \mathbb{B}^k \longrightarrow \mathbb{B}^k$ usada para sucessivas transformações do estado.

(ii) Uma função "justa" $g : \mathbb{B}^k \longrightarrow \mathbb{B}^t$ que "comprime" os k *bits* do estado em t *bits* de *output*.

Entende-se que g é "justa" no sentido em que, se s está uniformemente distribuído em $\mathbb{B}^k$, $g(s)$ estará uniformemente distribuído em $\mathbb{B}^t$.

Estado: $\mathbf{s}$ *toma valores sobre* $\mathbb{B}^k$

$H_x(u)$
  $\mathbf{s} \leftarrow x$ # *a "chave" x inicializa o estado s*
  while $u \neq \epsilon$
    $r \leftarrow \mathsf{next}(k, u)$
    $\mathbf{s} \leftarrow f(\mathbf{s} \oplus r)$
  write $g(\mathbf{s})$

Algoritmo 5: De funções unidireccionais para funções de *hash*.

Funções de *hash* e GPA's

Existe uma ligação simples entre funções de *hash* e geradores pseudo-aleatórios usando a propriedade de unidirecionalidade.

Seja $h : \mathbb{B}^* \longrightarrow \mathbb{B}^t$ uma função de *hash* e seja k o tamanho da "semente".

Estado: $\mathbf{s}$ *valores em* $\mathbb{B}^k$

RANDOM(x)
  $s \leftarrow x$ # *"seed"*
  forever
    $s \leftarrow h(s)$
    write s

Algoritmo 6: De funções de *hash* para GPA's

3.7.Assimetria

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[20], como um comportamento assimétrico que conduz à assimetria de chaves.

Nesta secção vamos tomar por referência

(i) uma classe $\mathcal{C}$ de computações tratáveis

(ii) uma classe contável $\mathcal{U}$ de funções unidireccionais $u : \mathbb{B}^* \longrightarrow \mathbb{B}^*$ fechada por composição de funções,

(iii) uma função de *hash* (unidireccional e resistente a colisões) $h \in \mathcal{U}$.

(iv) Uma codificação $\{\alpha_s\}_{s\in\mathbb{B}^*}$ das strings finitas (ver definições **??** e **??**).

Apesar de os elementos de $\mathcal{U}$ serem funções unidireccionais e, consequentemente, não serem invertíveis, pode acontecer que aceitem uma forma mais fraca de "inversa" definida do modo seguinte:

DEFINIÇÃO 29 *Uma função $f \in \mathcal{U}$ é h-**fracamente invertível** se existe uma função $\mathcal{C}$-implementável t tal que, para toda a implementação*

[20] Como sabemos, não existe prova de que realmente existam funções unidireccionais; existe apenas uma boa possibilidade de que certas classes de funções exibam esse comportamento.

U^h e toda a implementação U^f, existe uma implementação U^t tal que

$$\forall \rho \in \mathcal{C}: \quad \{U^f(U^t(\rho))\} \sim \{U^h(\rho)\} \tag{54}$$

Um t que verifique esta condição **abre** *f em relação a h; e a relação entre estas funções representa-se por*

$$t \triangleright_h f$$

ou, se h está implícito, simplesmente por $t \triangleright f$.

Comentários

1. A relação $t \triangleright_h f$ está definida em termos das implementações mas, essencialmente, a condição representada por (**??**) é uma formalização de uma noção muito simples: o de que são computacionalmente indistinguíveis as funções de $t(f(x))$ e $h(x)$. Abusando um pouco da notação pode-se escrever

   $$t \cdot f \simeq h \tag{55}$$

   Comparemos com a definição usual de invertibilidade

   $$t \cdot f = f \cdot t = \mathrm{id} \tag{56}$$

   Note-se que, em (**??**), a comparação entre funções é feita computacionalmente recorrendo às suas implementações; em contraste, em (**??**), a comparação é feita directamente nas funções de forma extensional (duas funções são iguais se produzem resultados iguais para argumentos iguais).

   Outra diferença, reside no facto que (**??**) usar como "fulcro" uma função de *hash* h e não a função identidade usada em (**??**). A última diferença vem do facto de a invertibilidade proposta ser apenas uma "invertibilidade à esquerda".

Através da relação de não-invertibilidade fraca chegamos à noção de *trapdoor*.

DEFINIÇÃO 30 *Um conjunto* $\mathcal{T}_f \subseteq \mathcal{U}$ *é um* **trapdoor set** *de* f *se contém todos* t *que abrem* f *(isto é,* $t \triangleright_h f$ *implica* $t \in \mathcal{T}_f$*) e se existe pelo menos um tal* t*.*

Escreve-se $f, g \blacktriangleright_h u$ *se, para todos* $a, b \in \mathcal{U}$*, verificando-se* $(a \triangleright_h f)$ *e* $(b \triangleright_h g)$ *também se verifica* $(a \triangleright_g u)$ *e* $(b \triangleright_f u)$*.*

Os triplos $\langle f \ , \ g \ , \ u \rangle$*, na relação* $f, g \blacktriangleright_h u$*, designam-se por* **triplos trapdoor***.*

Comentários

1. Um *trapdoor set* de f forma-se quando existe pelo menos um t que abre f e o conjunto contém todos esses t. Pode também conter funções que não abrem f e, de certa forma, uma medida do grau de informação que contém, está na probabilidade de um elemento aleatório desse conjunto ser ou não uma *trapdoor* de f.
2. Verifica-se a relação $(f, g \blacktriangleright_h u)$ quando se verifica a inferência

$$\frac{(a \triangleright_h f) \qquad (b \triangleright_h g)}{(a \triangleright_g u) \ \wedge \ (b \triangleright_f u)} \tag{57}$$

   É importante notar-se que, dados $f, g \in \mathcal{U}$ arbitrários, não está garantida a existência de um u que verifique $f, g \blacktriangleright_h u$ nem que, caso exista, esse u seja único.

LEMA 1 *Para todo* $f, g \in \mathcal{F}$*,*

(1) $f, h \blacktriangleright_h f$ *verifica-se sempre,*

(2) Se $(a \triangleright_h f)$*,* $(b \triangleright_h g)$ *e* $(f, g \blacktriangleright_h u)$*, então verifica-se*

$$(a \cdot b) \triangleright_h u \quad \wedge \quad (b \cdot a) \triangleright_h u$$

onde $(\cdot)$ *denota a composição de funções.*

Nota

A prova deste resultado é consequência directa da definição **??**.

□

Vamos chamar **permutações** às funções $f : \mathbb{B}^* \longrightarrow \mathbb{B}^*$ que preservam, no resultado, a incerteza contida no argumento. Um exemplo de permutações são as funções que preservam comprimentos das strings; isto é,

$$|f(x)| \;=\; |x| \qquad \forall s \in \mathbb{B}^*$$

No entanto outras formas mais flexíveis de "preservação de informação" podem ser usadas.

DEFINIÇÃO 31 *Uma* ***"weak-trapdoor permutation" (wtp)*** *um par* $(F\ ,\ \mathcal{T})$ *em que*

(1) $F \subseteq \mathcal{U}$ *é uma cifra e* $\mathcal{T} \subseteq \mathcal{U} \times F$ *é uma relação unidirecional,*

(2) Para cada $f \in F$, *as o conjunto de funções* $\mathcal{T}^{-1}(f) \doteq \{\, t \in \mathcal{U} \mid \langle t\ ,\ f\rangle \in \mathcal{T}\,\}$ *forma um* ***trapdoor set*** *de* f.

Num par $\langle t, f\rangle \in \mathcal{T}$ *em que* $t \triangleright_h f$, *o elemento* t *designa-se por* ***chave privada*** *e o elemento* f *designa-se por* ***chave pública***.

Notas

1. Uma wtp é formada por dois objectos essenciais: uma cifra, F e uma relação unidrecional $\mathcal{T}$.

   Porque F é uma cifra os seus elementos verificam as condições que estamos à espera: não-invertibilidade do criptograma e não-invertibilidade da chave.

   Porque $\mathcal{T}$ é unidireccional isto significa que, para todo $f \in F$, é intratável encontrar um $t \in T$ tal que $\langle t, f\rangle$ estão na relação $\mathcal{T}$.
2. A relação $\mathcal{T}$ determina, para cada f, um *trapdoor set* de f; isto significa que contém todos t (*trapdoors* de f) que abrem f e que é não vazio.

3. Seria possível "liberalizar" a definição não impondo que h seja uma função de *hash* mas apenas uma qualquer função unidireccional $h \in \mathcal{U}$; isto é equivalente a não forçar h a ser resistente a colisões.
   É elementar converter um tal triplo $\langle f, t, h\rangle$ num outro triplo $\langle f', t', h'\rangle$ que satisfaz a nossa definição. Basta escolher uma qualquer função de *hash* H e definir
   $$h' = h \cdot H \quad , \quad f' = f \cdot H \quad , \quad f' = f$$

4. Nesta definição não está explícito o modo como, numa situação concreta, se pode escolher uma chave pública f e determinar a respectiva *trapdoor* t. Exige-se, apenas, que seja computacionalmente intratável inferir a informação *trapdoor* a partir da informação pública (chave pública e função de *hash*).
   Para poder realizar essa escolha é necessário impor um mecanismo gerador que analisaremos em seguida (ver definição **??**).

Exemplo 22 :

A definição de "weak-trapdoor" é suficientemente forte para permitir a construção de várias técnicas criptográficas abstractas seguras (como veremos em seguida). Convém, por isso, ver algumas instâncias concretas destas *wtp* que conduzem a implementações concretas dessas técnicas abstractas.

1. Um dos exemplos de estruturas algébricas que aparentemente[21] conduzem a permutações *trapdoor* assenta na exponenciação num domínio de inteiros.
   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
   $$h(x) = g^x \pmod p$$
   percorrem todos os inteiros $\mathbb{Z}_p^*$; tais g chamam-se *elementos primitivos* ou *geradores primitivos*.
   Acredita-se que a função $x \mapsto h(x)$ seja unidireccional; isto é, é injectiva, computacionalmente tratável mas com inversa (conhecida por *logaritmo discreto*) que é supostamente intratável.

[21] 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.

Para cada $a \in \mathbb{Z}_{p-1}$ tal que $h(a)$ é também um elemento primitivo, definem-se duas funções adicionais

$$f_a\colon\ x \mapsto h(a^{-1})^x \pmod p \qquad , \qquad t_a\colon\ x \mapsto x^a \pmod p$$

Facilmente se verifica que a classe $\{f_a\}$ é uma "trapdoor permutation" para o *hash* h e que t_a é a chave privada correspondente à chave pública f_a. De facto:

(i) A função $h : x \mapsto g^x \pmod p$ é unidireccional mas, em geral, <u>não é</u> 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 $x' = x \pmod p$.
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.

(ii) Cada $f_a : x \mapsto \beta^x \pmod p$, com $\beta \doteq h(a^{-1})$, é unidireccional (é, de facto, injectiva). São indistinguíveis no sentido em que nenhuma decisão tratável consegue distinguir entre o uso de uma função f_a e o uso de uma escolha $f_a \| f_b$.

(iii) Verifica-se, tomando congruências módulo p,

$$t_a(f_a(x)) = (\beta^x)^a = g^{a^{-1}\,x\,a} = g^x = h(x)$$

(iv) É intratável, dado f_a (ou, equivalentemente, dado $\beta = h(a^{-1})$) determinar t_a (ou, equivalentemente, determinar a) dado que tal implica calcular o logaritmo discreto de β.

2. Se $p > q$ são dois primos grandes e sejam $m \doteq p\,q$ e $n \doteq (p-1)\,(q-1)$; determinar n conhecendo m é equivalente a conhecer-se a factorização de m 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

$$a = b \pmod n \implies x^a = x^b \pmod m$$

Neste contexto, escolhe-se uma função de *hash* arbitrária h e define-se uma permutação *trapdoor* da seguinte forma:

Para todo $k \in \mathbb{Z}_n$, *tal que* $\gcd(k, n) = 1$, *define-se*

* $f_k : x \mapsto h(x)^r \pmod m$ *sendo* $r \doteq k^{-1} \pmod n$.
* $t_k : x \mapsto x^k \pmod m$

A família $\{f_k\}$ é (a menos de um eventual estratégia para factorizar m) uma permutação *trapdoor* porque:

(i) Todos os f_k são unidireccionais (não só devido à função de hash $h(\cdot)$ mas também a exponenciação $x \mapsto x^r$). As funções são também indistinguíveis no sentido referido no exemplo anterior.

(ii) Tem-se, com as igualdades referindo congruências módulo m,

$$t_k(f_k(x)) \;=\; (h(x)^r)^k \;=\; h(x)^{r\,k} \;=\; h(x)$$

porque, pela definição, $r\,k = 1 \pmod{n}$ e, como consequência do teorema RSA, tem-se $h(x)^{r\,k} = h(x)^1 \pmod{m}$.

(iii) Determinar t_k a partir de f_k equivale a recuperar k a partir de $k^{-1} \pmod{n}$ e isto exige o conhecimento de n que é equivalente à factorização de m.

Determinadas técnicas criptográficas podem ser definidas directamente a partir da noção de *weak-trapdoor pernutation* abstraindo completamente em relação aos detalhes das funções usadas.

No que se segue consideramos definida uma permutação *trapdoor* $\mathcal{F}$ e está seleccionada uma função de *hash* de referencia $h \in \mathcal{U}$

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 $\langle y, \sigma \rangle$ em que $y \in \mathbb{B}^*$ é a *imagem* do texto e tem o mesmo comprimento que o texto, enquanto que $\sigma \in \mathbb{B}^\infty$ é infinita e designa-se por *redundância*.

Protocolo 3.7.1

cifrar: *objectivo: dado o texto x construir o criptograma $\langle y, \sigma \rangle$*

(c.1) escolhe um $f \in \mathcal{F}$ e publicita esse valor.
(c.2) toma uma string aleatória ω e calcula $\sigma \leftarrow f(\omega)$ e $k \leftarrow h(\omega)$
(c.3) calcula $y \leftarrow x \oplus k$ e publicita o criptograma $\langle y \ , \ \sigma \rangle$

decifrar: *objectivo: reconstruir x a partir do criptograma $\langle y \ , \ \sigma \rangle$*

(d.1) recolhe a *trapdoor* t correspondente a f
(d.2) recupera k calculando $k \leftarrow t(\sigma)$.
(d.3) recupera x calculando $x \leftarrow y \oplus k$.

Notas

Os passos críticos, tanto para cifrar como decifrar, são os primeiros. O passo (c.**??**) assume que, quem produz o criptograma, pode escolher uma qualquer chave pública f que determina quem vai poder decifrar esse criptograma. Os restantes passos, no passo "cifrar", podem ser executados por qualquer agente que tenha capacidade de executar o primeiro. O passo (d.**??**) assume que quem decifra tem acesso à informação *trapdoor* t correspondente à chave pública f.

Se um intruso tem, de alguma forma, acesso a x descobre facilmente a chave k. No entanto isto não é uma quebra de "futura segurança" já que a chave k é uma *chave de sessão* (é usada apenas uma vez); de facto, qualquer futuro uso da cifra gera um novo ω e, portanto, uma nova chave $h(\omega)$.

Assinatura de 1 *bit*

É possível definir uma classe muito geral de assinaturas digitais sobre mensagens de tamanho 1 *bit*[22]. O problema a resolver é

> *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 3.7.2

Inicialização — *B determina o agente A de onde surgirá a mensagem*

(o.1) *B* escolhe uma chave pública f e publicita-a.
(o.2) são geradas duas strings aleatórias ω_0 e ω_1 e, para $i \in \{0,1\}$, calculados

$$\sigma_i \leftarrow f(\omega_i) \qquad \text{e} \qquad k_i \leftarrow h(\omega_i)$$

(o.3) *B* destrói $\langle \omega_0, \omega_1 \rangle$, torna pública $\langle \sigma_0, \sigma_1 \rangle$ e mantém secreto $\langle k_0, k_1 \rangle$.

Assinatura — *A gera a assinatura s para uma mensagem b com um único bit.*

(s.1) *A* recolhe a informação *trapdoor* t correspondente a f
(s.2) se $b = 0$ calcula $s \leftarrow t(\sigma_0)$; se for $b = 1$ calcula $s \leftarrow t(\sigma_1)$,
(s.3) torna público o par $\langle b, s \rangle$.

Verificação — *B verifica se s é a assinatura correcta para b*

(v.1) *B* aceita a mensagem b sse for válido

$$(s = k_0) \wedge (b = 0) \ \vee \ (s = k_1) \wedge (b = 1)$$

[22] 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 f e o segredo representado pela informação *trapdoor* correspondente. Isto significa que tem de ser pública a associação ente cada agente e a prova que lhe está associada. Para isso vamos supor que existe uma função $C : \mathbb{N} \longrightarrow \mathcal{F}$ que a cada agente, identificado um inteiro n (o seu código, ou *login*), associa uma chave pública f.

Protocolo 3.7.3

Intensão — *A manifesta a intensão de ser identificado*

(o.1) publicita o seu código n

Desafio — *B gera um desafio σ aleatório*

(d.1) escolhe a chave pública $f = C(n)$
(d.2) gera uma string aleatória ω, calcula $\sigma \leftarrow f(\omega)$ e $k \leftarrow h(\omega)$,
(d.3) destrói ω, publicita σ e guarda, como segredo, k.

Resposta — *A gera a resposta adequada ao desafio*

(r.1) recolhe a *trapdoor* t
(r.2) calcula $r \leftarrow t(\sigma)$ e publicita este valor

Verificação — *B verifica a correcção da resposta*

(v.1) decide se $r = k$

Acordo de chaves com autenticação mútua

Dois agentes A e B, que possuem (cada um) um segredo na forma de trapdoor, *acordam num segredo comum k e, simultaneamente, certificam-se que estão a comunicar com o interlocutor legítimo.*

Protocolo 3.7.4

segredo e desafio — *A gera o segredo comum k*

(s.1) *A* obtém a chave pública f_B
(s.2) gera ω aleatório, calcula $\sigma_1 \leftarrow f_B(\omega)$ e $k_A \leftarrow h(\omega)$
(s.3) *assume* $k = k_A$
(s.4) destrói ω, guarda k_A e publicita σ_1

resposta e desafio — *B recolhe o segredo e responde ao desafio*

(r.1) *B* recorre a informação *trapdoor* t_B e a chave pública f_A
(r.2) calcula $k_B \leftarrow t_B(\sigma_1)$ e guarda-o,
(r.3) *assume* $k = K_B$
▷ se a execução for correcta $k_B = t_B(f_B(\omega)) = h(\omega) = k_A$; logo, ambos agentes assumem a mesma chave k
(r.4) calcula $\sigma_2 \leftarrow f_A(k_B)$ e publicita-o.

resposta — *A reconhece o interlocutor e responde ao desafio*

(rr.1) *A* recolhe a informação *trapdoor* t_A
(rr.2) calcula $k_1 \leftarrow t_A(\sigma_2)$
(rr.3) *aceita prosseguir se* $k_1 = h(k)$
▷ se o protocolo for bem executado será $k_1 = t_A(f_A(k_B)) = h(k_B) = h(k_A) = h(k)$
(rr.4) se o passo anterior teve sucesso, calcula $\sigma_3 \leftarrow f_B(k_1)$ e publicita-o.

verificação — *B reconhece o interlocutor*

(v.1) calcula $k_2 \leftarrow B(\sigma_3)$
(v.2) *aceita prosseguir se* $k_2 = h(h(k))$
▷ se o protocolo for bem executado, será $k_2 = t_B(f_B(k_1)) = h(k_1) = h(h(k_B)) = h(h(k))$.
(v.3) termina com sucesso

Este tipo de protocolo evita, por exemplo, que um dos agentes (p.e. *A*)

julgue que está partilhar um segredo com B e, de facto, está a partilhá-lo com um intruso.

O protocolo tem 3 passos: no primeiro o segredo é gerado; nos dois restantes (de forma semelhante ao protocolo de identificação) cada um dos agentes certifica-se que está a interagir com o interlocutor certo.

□

No contexto das wtr $\langle \mathcal{F} , \mathcal{T} \rangle$ convém especificar vários tipos de problemas que permitem classificar as diferentes estruturas matemáticas onde eles têm solução. Seja $\mathcal{H}$ o subconjunto de $\mathcal{F}$ formado pelas funções de *hash* (resistentes a colisões).

Decisão *trapdoor* (D-tr)
: Dados $h \in \mathcal{H}$ e $(t, f) \in \mathcal{T}$ determinar se estão em relação $t \triangleright_h f$.

Computação *trapdoor* (C-tr)
: Dado $(f, h) \in \mathcal{F} \times \mathcal{H}$, determinar um $t \in \mathcal{U}$ que verifique $t \triangleright_h f$.

A definição de wtr impõe que o problema C-tr não tenha uma implementação tratável. Considera-se, porém, que o problema D-tr é sempre simples no sentido em que existe uma enumeração do subconjunto de $\mathcal{T}$ formado pelos pares (t, f) que verificam a relação $t \triangleright_h f$.

O uso de uma permutação *trapdoor* presupõe que exista um mecanismo para gerar o par (t, f) de forma que seja computacionalmente tratável.

Para isso é preciso, em primeiro lugar, uma função $\mathcal{C}$-implementável $G : \mathbb{B}^\infty \to \mathcal{T}$ que gere um par $(t, f) \in \mathcal{T}$ a partir de uma possibilidade aleatória.

Como um atacante que conheça f não deve ser capaz de reconstruir a possibilidade que lhe deu origem, deve-se exigir que G seje unidireccional e, idealmente, livre de colisões. Isto é, G deve ser uma função de *hash*.

Isto só não basta: de facto um par arbitrário $(t, f) \in \mathcal{T}$ pode não verificar a relação $t \triangleright_h f$. Para isso precisamos, pelo menos, de uma solução para o problema D-tr

Nestas circunstâncias:

DEFINIÇÃO 32 *Um* **gerador** *para a permutação* $\langle \mathcal{F} , \mathcal{T} \rangle$ *é um par* $\langle G , \phi \rangle$ *em que:*

(i) $G : \mathbb{B}^\infty \to \mathcal{T}$ *é uma codificação unidireccional de* $\mathcal{T}$

(ii) ϕ *é uma* $\mathcal{C}$*-enumeração do sub-conjunto de* $\mathcal{T}$ *formado pelos pares* (t, f) *que estão na relação* $(t \triangleright_h f)$.

□

Associados a uma *weak trapdoor* $\langle \mathcal{F} , \mathcal{T} \rangle$ e à relação $(f, g \blacktriangleright_h u)$ podem ser definidos vários problemas:

Geração DH-*trapdoor* (GDH-tr)
: Dados $h \in \mathcal{H}$ e pares $(a, f) \in \mathcal{T}$ e $(b, g) \in \mathcal{T}$ gerar $u \in \mathcal{F}$ tal que $(f, g \blacktriangleright_h u)$.

Decisão DH-*trapdoor* (DDH-tr)
: Dados $(f, g, u) \in \mathcal{F}^3$ e $h \in \mathcal{H}$, verificar se estão na relação $(f, g \blacktriangleright_h u)$

Computação DH *trapdoor* (CDH-tr)
: Dados $(f, g) \in \mathcal{F}^2$ e $h \in \mathcal{H}$, calcular $u \in \mathcal{F}$ tal que $(f, g \blacktriangleright_h u)$

Esquema de Computação DH *trapdoor* (ECDH-tr)
: Dados $(f, g) \in \mathcal{F}^2$ e $h \in \mathcal{H}$, calcular $u \in \mathcal{F}$ tal que $f, g \blacktriangleright_h u$ usando, como oráculo, uma solução de DDH-tr.

Presupõe-se que o problema GDH-tr é simples e que os restantes problemas são intratáveis. Neste contexto,

DEFINIÇÃO 33 *Uma permutação* trapdoor $\mathfrak{T} = \langle \mathcal{F} \ , \ \mathcal{T} \rangle$ *é uma* **weak Diffie-Hellman trapdoor (wDH-tr)** *se, para todos* $f, g \in \mathcal{F}$,

$$\mathsf{dh}(f, g) \doteq \{ u \in \mathcal{F} \mid f, g \blacktriangleright_h u \} \tag{58}$$

é não-vazio e $\mathcal{C}$*-difícil em* $\mathcal{F}$.

É essencial ter-se em conta que qualquer computação que enumere as codificações de $\mathsf{dh}(f, g)$ é um teste de sucesso desprezável. Por isso qualquer enumeração do conjunto de tais codificações pode ser vista como informação *trapdoor* que permite resolver o problema CDH-tr.

Exemplo 23 : Usando um wDH-tr arbitrário é possível definir um **protocolo de distribuição de chaves** da seguinte forma:

Contexto:

i. Existem dois agentes A e B com acesso a chaves privadas a e b respectivamente,
ii. Existe um agente TA (*trust agent*) que tem acesso a informação *trapdoor* (p.ex. a ambas as chaves privadas) que lhe permite resolver o problema CDH-tr.
iii. Pretende-se que TA distribua pelos agentes A e B um segredo k.

Inicialização

(1) TA escolhe uma função de *hash* h aleatória e resolve e gera (f, g, u) tais que $(a \triangleright_h f)$, $(b \triangleright_h g)$ e $(f, g \blacktriangleright_h u)$.
Destrói f e g e mantém secretos h e u.

Protocolo:

(p1) TA gera um ω aleatório, calcula $k \leftarrow h(\omega)$ e $\sigma \leftarrow u(\omega)$.
Publicita σ e mantém k secreto.
(p2) Simultânemamente A calcula $r_a \leftarrow a(\sigma)$ e B calcula $r_b \leftarrow b(\sigma)$.
Ambos valores são publicitados.
(p3) Simultâneamente A calcula $k_a \leftarrow a(r_b)$ e B calcula $k_b \leftarrow b(r_a)$.

No final tem-se $k \sim k_a \sim k_b$ porque

$$k_a = a(b(\sigma)) = a(b(u(\omega))) \sim a(f(\omega)) \sim h(\omega) = k$$

$$k_b = b(a(\sigma)) = a(b(u(\omega))) \sim b(g(\omega)) \sim h(\omega) = k$$

□

Neste protocolo as funções h, f e g não devem ser vistas como chaves "públicas" uma vez que o seu conhecimento permite uma forma de ataque conhecido por **homem-no-meio**:

Contexto

i. Um intruso I conhece h, f e g tais que $(a \triangleright_h f)$ e $(b \triangleright_h g)$

Ataque

(q1) I gera ω' e calcula $k' \leftarrow h(\omega')$, $r'_a \leftarrow g(\omega')$ e $r'_b \leftarrow f(\omega')$.

(q2) Após o passo (p**??**) o atacante intercepta r_a e r_b substituindo-os por r'_a e r'_b.

(q3) A e B executam normalmente o passo (p**??**):
A calcula $k_a \leftarrow a(r'_b)$ e B calcula $k_b \leftarrow b(r'_a)$.

No final tem-se $k_a \sim k_b \sim k'$ que é uma chave conhecida do intruso I.

Note-se que se o intruso dispusesse da possibilidade de resolver o problema CDH-tr poderia assumir completamente a personalidade do TA e, desta forma, também ter acesso à chave.

Quando $\mathrm{dh}(f, g)$ é um conjunto singular (existe um único u que verifica $f, g \blacktriangleright_h u$) as wDH-tr tomam uma forma particular.

Nesses casos define-se uma operação binária $\star : \mathcal{F}^2 \to \mathcal{F}$ que associa cada $(f, g) \in \mathcal{F}^2$ ao único elemento de $\mathrm{dh}(f, g)$.

$$u = f \star g \qquad \text{sse} \qquad (f, g \blacktriangleright_h u) \tag{59}$$

LEMA 2 *A função $\star : \mathcal{F}^2 \to \mathcal{F}$ determina em $\mathcal{F}$ a estrutura de um semigrupo comutativo que tem h por elemento neutro.*

Prova: Pelo lema **??**, claramente que $\star$ é comutativa, associativa, e tem h por elemento neutro.

DEFINIÇÃO 34 *Uma* ***weak DH trapdoor (wDH-tr)*** *é wDH-tr em que a operação* $(\star)$ *determina um grupo.*

Um wDH-trg tem a propriedade de que todo $f \in \mathcal{F}$ tem um inverso f^{-1} tal que $f \star f^{-1} = h$. Esta propriedade permite sugerir um protocolo de assinatura digital como se indica em seguida.

Exemplo 24 :

Contexto

(i) Estão definidos um $h \in \mathcal{H}$ e um wDH-trg $\langle \mathcal{F} \ , \ \mathcal{T} \rangle$.

(ii) Para um $m \in \mathbb{B}$ e um $x \in \mathbb{B}^*$ arbitrários define-se

$$[m](x) \doteq h^m(x) = \begin{cases} x & \text{se} \quad m = 0 \\ h(x) & \text{se} \quad m = 1 \end{cases}$$

(iii) A pretende assinar digitalmente uma mensagem $m \in \mathbb{B}$.

(iv) É pública uma chave privada $f \in \mathcal{F}$

(v) A dispõe de informação *trapdoor* que lhe dá capacidade para gerar $(\tau, r) \in \mathcal{T}$, com $(\tau \triangleright_h r)$, e calcular u tal que $(f, r \blacktriangleright_h u)$.

Assinatura

O agente A

1. Gera $(\tau, r) \in \mathcal{T}$, com $\tau \triangleright_h r$, e um $\omega \in \mathbb{B}^\infty$ aleatórios.
2. Gera u tal que $(f, r \blacktriangleright_h u)$.
3. Calcula $z \leftarrow h(\omega)$ e $s \leftarrow u([m](z))$.
4. Destrói ω, u e publicita a assinatura (τ, s, z).

Verificação

Um agente B com acesso à chave pública f, à mensagem m e à assinatura (τ, s, z), verifica-a do seguinte modo

1. Calcula $\nu \leftarrow f([m](z))$.
2. Aceita a assinatura se e só se $\tau(s) = \nu$.

Justificação

Correcção

Como $(f, r \blacktriangleright_h u)$ tem-se $\tau(u(y)) = f(y)$ para todo y. Com $y = [m](x)$ é $s = u(y)$ e $\nu = f(y)$.

Segurança

4.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.

4.1.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 25 : $\mathbb{Z}_5$

$+$	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 $a \times b = 1 \pmod{n}$.

FACTO 2 *$a \in \mathbb{Z}_n$ 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). [23]

FACTO 3 *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 26 :

$\mathbb{Z}_9^*$ 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

[23] 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**.

4.2.Resultados Fundamentais da Divisibilidade

TEOREMA 1 (FUNDAMENTAL DA ARITMÉTICA) *Cada* $m > 1$ *admite uma única* **factorização**

$$m = p_1^{e1} \times p_2^{e2} \times \cdots \times p_k^{ek}$$

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

TEOREMA 2 (PEQUENO DE FERMAT) *Se* p *é primo então, para todo* $a \geq 0$, *verifica-se* $a^p = a \pmod p$.

COROL‖ARIO 1 *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 $a^{-k} = a \cdot a^{-k-1} = a^p \cdot a^{-k-1} = a^{p-k-1}$.

COROL‖ARIO 2 *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, $x = y$.

DEFINIÇÃO 35 *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*)

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

- $m = \sum_{a|m} \phi(a)$ ($a|m$ 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 27 :

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

DEFINIÇÃO 36 *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* $\mathbb{Z}_m^*$

FACTO 4

(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 $\phi(21) = 12$).

4.3.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.

DEFINIÇÃO 37 *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).

FACTO 5 *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[24]:

- *Se $(p + 1)$ é divisível por 4 então $x = \pm a^{(p+1)/4} \pmod{p}$.*

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

- *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.*

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

FACTO 6 *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. . .

4.4.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.

4.5.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 $\upsilon > 0$ tal que $\overbrace{1+1+\cdots+1}^{\upsilon \text{ vezes}} = 0$.

$\mathbb{Z}_p$, para p primo, é um **corpo finito** de característica p; note-se que, neste caso, $\mathbb{Z}_p \setminus \{0\} \equiv \mathbb{Z}_p^*$

FACTO 7 *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***[25] *$\mathbf{c} \in \mathbb{Z}_p[X]$ de grau n tal que*

$$\mathbb{F}_m \cong \mathbb{Z}_p[X]/\mathbf{c}\mathbb{Z}_p[X]$$

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

[25] 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 $k < m$.

todas as raízes desse polinómio[26]*; seja* β *uma raíz de* $\mathbf{c}$ *em* K *que não seja raíz de nenhum polinómio* $(X^k - 1)$ *com* $k < m - 1$ [27]*. 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* $n =$ grau$(\mathbf{c})$*, o elemento genérico desse corpo tem a forma*

$$a_0 + a_1\,\beta + a_2\,\beta^2 + \cdots + a_{n-1}\,\beta^{n-1} \quad com \quad a_i \in \mathbb{Z}_p \tag{60}$$

(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

(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.

[26] Diz-se que K é um corpo que **fractura** o polinómio $\mathbf{c}[X]$ sobre $\mathbb{F}_p$.

[27] Estas raízes designam-se por **raízes primitivas**.

(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 (**??**), 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 (**??**).

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 28 : 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.

$$(X^2+1) \qquad (X^2+X-1) \qquad (X^2-X-1)$$

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[28]; a partir daqui existem duas representações possíveis para os elementos de $\mathbb{F}_9$

1. Tomemos uma qualquer raiz β deste polinómio numa extensão K de $\mathbb{Z}_3$ (um qualquer outro corpo do qual $\mathbb{Z}_3$ seja um sub-corpo) que contenha todas as raízes de $X^2 - X - 1$. Em K o elemento β verifica a equação $\beta^2 = \beta + 1$.
   O sub-anel $\mathbb{Z}_3(\beta)$ de K, formado por todos os elementos da forma $a + b\,\beta$, é um corpo que verifica as seguintes igualdades de adição e multiplicação:

$$\begin{aligned}(a + b\,\beta) + (c + d\,\beta) &= (a + c) + (b + d)\,\beta \\ (a + b\,\beta) \cdot (c + d\,\beta) &= (a\,c + b\,d) + (a\,d + b\,c + b\,d)\,\beta\end{aligned}$$

   A menos de um isomorfismo o corpo $\mathbb{Z}_3(\beta)$ é único e os seus elementos são $\{0\ ,\ -1\ ,\ 1\ ,\ \beta\ ,\ -\beta\ ,\ 1+\beta\ ,\ 1-\beta\ ,\ -1+\beta\ ,\ -1-\beta\}$. A outra raiz do polinómio característico pode ser calculada como $\beta^3 = 1 - \beta$.
   Nesta representação, cada $x \in \mathbb{F}_9$ é descrito por um elemento de $\mathbb{Z}_3(\beta)$. 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

$\beta^0 = 1$	$\beta^1 = \beta$	$\beta^2 = 1 + \beta$	$\beta^3 = 1 - \beta$
$\beta^4 = -1$	$\beta^5 = -\beta$	$\beta^6 = -1 - \beta$	$\beta^7 = -1 + \beta$

   Para calcular o inverso de um elemento pode-se usar esta tabela; por exemplo $(1+\beta)^{-1} = (\beta^2)^{-1} = \beta^6 = -1 - \beta$; note-se que $6 = -2 \pmod 8$.
   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

   Para calcular, por exemplo, $\beta^5 + \beta^3$ calculamos $\beta^{\tau(5-3)+3} = \beta^2$ (note-se que os expoentes são vistos como elementos de $\mathbb{Z}_8$). Também $\beta^6 + \beta^2 = 0$ porque $\tau(6-2) = \perp$.

[28] 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.

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.

Exemplo 29 : 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[29] 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 $\underbrace{\sigma \cdot \sigma \cdots \sigma}_{k \text{ vezes}}$.

[29] O morfismo σ fixa x quando $\sigma(x) = x$.

FACTO 8 *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.

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 $\mathbb{F}_p$.
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 (**??**), 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

FACTO 9 *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 $\mathsf{Gal}(L/K)$ 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 $\mathsf{Gal}(L/K)$ 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 $x \mapsto x^p$).

DEFINIÇÃO 38 *Sejam L e K corpos finitos como no facto* **??**. *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.

FACTO 10 *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^*.*

Este resultado pretende afirmar que o traço $\mathsf{tr}(x)$ 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 $t = \mathsf{tr}(x)$; 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 $\mathsf{tr}(\cdot)$ 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 $\mathsf{tr}(y) \neq 0$; 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 $x \doteq (c/\mathsf{tr}(y))\,y$, temos um elemento com traço c.

4.6.Corpos de Galois

São os corpos finitos de característica 2.

FACTO 11 *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 30 : O corpo finito $\mathbf{GF}(2^4)$ determinado pelo polinómio característico $\mathbf{c}[X] = (X^4 + X + 1)$ está representado na tabela seguinte 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.

Usando essa tabela pode-se ver alguns exemplos de codificação de $\mathbb{Z}_{16}$ em $\mathrm{GF}(2^4)$

$$7 + 11 = 0111 + 1011 = 1100 = 12$$

$$7 * 11 = 0111 * 1011 = X^{10} * X^7 = X^{17} = X^2 = 0100 = 4$$

$$(9)^2 = (1001)^2 = (X^{14})^2 = X^{28} = X^{13} = X^3 + X^2 + 1 = 1101 = 13$$

$$9^{-1} = (X^{14})^{-1} = X^{-14} = X^1 = 0010 = 2$$

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

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 **??** temos, neste caso, característica $p = 2$ e extensão em cause é $[\mathrm{GF}(2^n)/\mathrm{GF}(2)]$, temos $s = n$ e o morfismo de Frobenius é $\sigma : x \longrightarrow x^2$. Portanto

$$\mathrm{tr}(x) = \sum_{k=0}^{n-1} x^{2^k} \quad , \quad \mathrm{nr}(x) = \prod_{k=0}^{n-1} x^{2^k}$$

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 (**??**) no facto **??**

$$x_0 + x_1 \cdot \beta + \cdots + x_{n-1} \cdot \beta^{n-1} \qquad x_i \in \mathrm{GF}(2) \tag{61}$$

indica que o conjunto

$$\mathcal{B}_{\beta,0} = \{1 \ , \ \beta \ , \ \beta^2 \ , \ \cdots \ , \ \beta^{n-1}\} \tag{62}$$

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 (**??**).

> *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 **??** 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

DEFINIÇÃO 39 *A* ***matriz companheira*** *do polinómio* $\mathbf{c}[X]$ *é uma matriz* $\mathbf{A} \in \mathrm{GF}(2)^{n\times n}$ *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 **??** 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

FACTO 12 *Para quaisquer* $x, y \in \mathsf{GF}(2^n)$, *verifica-se* $(\beta \cdot x)^{\tilde{}} = \mathbf{A}\, x^{\tilde{}}$ *e, genericamente,*

$$(y \cdot x)^{\tilde{}} \;=\; \bigoplus_{y_i=1} \mathbf{A}^i\, x^{\tilde{}} \tag{63}$$

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 (**??**), 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

$$\mathcal{B}_{\beta,1} \;=\; \{\beta \;,\; \beta^2 \;,\; \cdots \;,\; \beta^{n-1} \;,\; \beta^n\} \tag{64}$$

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

$$\mathcal{N}_\rho = \{\rho\ ,\ \rho^2\ ,\ \cdots\ ,\ \rho^{2^k}\ ,\ \cdots\ ,\ \rho^{2^{n-1}}\}$$

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

$$x = x_0 \cdot \rho + x_1 \cdot \rho^2 + x_2 \cdot \rho^4 + \cdots + x_{n-1} \cdot \rho^{2^{n-1}} \qquad (65)$$

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 (**??**), 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

$$x \cdot y = \sum_{x_i=1} \sum_{y_j=1} \rho^{2^i} \cdot \rho^{2^j} \tag{66}$$

é bastante mais complicada e, genericamente, mais complexa nas bases normais do que nas bases polinomiais. A dificuldade reside no facto de (**??**) 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**.

FACTO 13 *Se* $p = n + 1$ *é primo e* 2 *é gerador do grupo cíclico* $\mathbb{Z}_p^*$ *então existe* ρ *que verifica* $\mathsf{tr}(\rho) = 1$ *e em que o conjunto*

$$\{1\ ,\ \rho\ ,\ \rho^2\ ,\ \rho^4\ ,\ \cdots\ ,\ \rho^{2^{n-1}}\}$$

determina um sub-grupo cíclico de $\mathsf{GF}^*(2^n)$ *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 $\mathsf{GF}^*(2^n)$, concluímos que $\mathsf{GF}^*(2^n)$ contém um sub-grupo cíclico $G \subseteq \mathsf{GF}^*(2^n)$ 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

$$G = \quad \{1\} \cup \{\, \rho^{2^k} \mid k \in \mathbb{Z}_n \,\} \quad = \quad \{1\ ,\ \rho\ ,\ \rho^2\ ,\ \cdots\ ,\ \rho^{2^{n-1}}\}$$

Resta provar que $\mathsf{tr}(\rho) = 1$; tem-se

$$\mathsf{tr}(\rho) = \sum_{k=0}^{n-1} \rho^{2^k} = \sum_{s=1}^{p-1} \rho^s = \quad (\rho^p + \rho) \cdot (1 + \rho)^{-1} = \ 1$$

porque, dada a ordem de G, $\rho^p = 1$.

FACTO 14 *Nas condições do facto* **??** *tem-se:*

(i) Se $2^i + 2^j = 0 \pmod p$ *então*

$$\rho^{2^i} \cdot p^{2^j} = \quad \rho + \rho^2 + \rho^4 + \cdots + \rho^{2^{n-1}}$$

(ii) Se $2^i + 2^j \neq 0 \pmod p$ *então*

$$\rho^{2^i} \cdot p^{2^j} = \rho^{2^{\lambda_{ij}}} \quad com \quad \lambda_{ij} \doteq i + \tau(j-i) \pmod n$$

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 (**??**): 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: Tomemos de novo 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 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 (**??**) 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 $(x \cdot y)\tilde{} = (0,1,1,1)$

ou seja $x \cdot y = \rho^2 + \rho^4 + \rho^8$

4.7.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 31 : (**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.*.

4.8.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 32 : 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

$$x_i = x_{i-1}^2 + 1 \pmod m \qquad y_i = (y_{i-1}^2 + 1)^2 + 1 \pmod m \tag{†}$$

$$\rho_i = x_i - y_i \pmod m$$

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

FACTO 15 *Se se verificar*

$$x^2 = y^2 \pmod{m} \qquad \& \qquad x \neq \pm y \pmod{m} \qquad (\dagger)$$

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* $(\dagger)$.

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 $a = p_1^{e_1} \times \ldots \times p_k^{e_k}$, 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.

4.9.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 criptograficamente 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

DEFINIÇÃO 40 *Dado um primo p ímpar, um algoritmo resolve* 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[30] 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*

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

$$(\alpha^a)^r = (\alpha^r)^{a_q} \pmod p \qquad \text{ou seja} \qquad \beta^r = (\gamma)^{a_q} \pmod p \qquad (\ddagger)$$

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 $x_1, x_2, \ldots x_e$.

[30] Muitas técnicas criptográficas usam primos da forma $p = q\,2^t + 1$, com q primo.

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.[31]
2. Define-se a sequência

$$x_0 = 1 \qquad x_{i+1} = \begin{cases} \beta\, x_i \pmod{p} & \text{se } x_i \in S_0 \\ x_i^2 \pmod{p} & \text{se } x_i \in S_1 \\ \alpha\, x_i \pmod{p} & \text{se } x_i \in S_2 \end{cases}$$

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

$$x_k = x_{2k} \pmod{p}$$

[31] Por exemplo, $S_i \equiv \{\, x \mid x = i \pmod{3} \,\} \quad i = 0, 1, 2$.

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* $\sqrt{q}$.

5.Funções Booleanas

O projecto e segurança de muitas técnicas criptográficas estão ligadas ao estudo das funções da forma $f : \mathbb{B}^n \rightarrow \mathbb{B}^n$ ou $f : \mathbb{B}^n \rightarrow \mathbb{B}$ 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

$$f : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2)$$

5.1.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 $\mathrm{sel} : \mathbb{U}_n \times \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$ generaliza-se facilmente

$$\mathrm{sel}(u, x) = \Pi_{i \in u}\, x_i \qquad , \qquad \mathrm{sel}(\emptyset, x) = 1 \tag{67}$$

DEFINIÇÃO 41 *Para $u \in \mathbb{U}_n$ designa-se por x_u o monómio* $\mathrm{sel}(u, x)$. *Designa-se por $[x]_u$ o polinómio* $\mathrm{sel}(u, x) \cdot \Pi_{i \notin u}(1 + x_i)$.

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 $\mathrm{GF}(2)^n$ pode ser escrita como um polinómio a n variáveis $x_0, x_1, \ldots, x_{n-1}$ dado por

$$f(x) \;=\; \sum_{u \in \mathbb{U}_n} a(u)\, x_u \tag{68}$$

para uma função $a : \mathbb{U}_n \longrightarrow \mathrm{GF}(2)$.

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 [32].

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 (**??**) escreve-se

$$f(x) \;=\; \sum_{u \in \mathcal{A}} x_u \tag{69}$$

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, $\emptyset \notin \mathcal{A}$.

[32] Veremos adiante que é importante definir um outra forma de espectro de funções booleanas: o espectro de *Walsh-Hadamard*.

Exemplo 33 : Considere-se as funções booleanas $f : \mathbb{B}^4 \to \mathbb{B}$ 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 $\delta : \mathsf{GF}(2)^n \to \mathsf{GF}(2)$ que verifica $\delta(x) = 1$ se e só se $x = (0, 0, \cdots, 0)$.

FACTO 16 *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

$$X \circledast Y \;\doteq\; \biguplus_{x \in X} \{\, x \cup y \mid y \in Y \,\}$$

FACTO 17 *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* $z \in \mathsf{GF}(2)^n$, *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, $\mathrm{wt}(f) = 2^{n-1}$.

FACTO 18 *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*

$$f = \sum_{z \in \mathrm{supp}(f)} \delta^{(z)} \tag{70}$$

(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 $z \in \mathrm{GF}(2)^n$ 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 $x \in \text{supp}(f)$ e $x \notin \text{supp}(f)$. Tem muita importância computacional quando o suporte da função é tem baixa cardinalidade (f tem pouco peso) e pode ser calculado facilmente. Nestas circunstâncias (**??**) é uma forma conveniente de representar a função.

O resultado (iii) é importante nomeadamente porque sugere um ataque a uma função boolena $\mathcal{Q} : \mathbb{B}^n \to \mathbb{B}$ 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 $\text{Sp}(\mathcal{Q})$ de $\mathcal{Q}$.

O resultado anterior diz-nos que o espectro $\text{Sp}(\mathcal{Q})$ é $\uparrow\bar{k}$. Portanto é um conjunto que tem um limite inferior $\bar{k}$ que pode ser calculado como

$$\bar{k} = \bigcap_{u \in \text{Sp}(\mathcal{Q})} u$$

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, $f \equiv \delta^{(z)}$.

TEOREMA 3 *O espectro* $\mathsf{Sp}(f)$ *de uma função booleana* f *verifica*

$$\mathsf{Sp}(f) = \biguplus_{z \in \mathsf{supp}(f)} \uparrow \bar{z} \tag{71}$$

Seja $f^{(z)}$ *a função definida por* $f^{(z)}(x) = f(z \oplus x)$ *. Então*

$$\mathsf{Sp}(f^{(z)}) = \biguplus_{y \in \mathsf{supp}(f)} \uparrow (\bar{z} \uplus \bar{y}) \tag{72}$$

A prova do primeiro resultado é consequência imediata de (**??**) e do facto **??**.

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 34 : Consider-se a função booleana em $\mathsf{GF}(2)^4$

$$f(x_0, x_1, x_2, x_3) = 1 + x_0 \cdot x_1 + x_0 \cdot x_2 + x_1 \cdot x_2 \cdot x_3$$

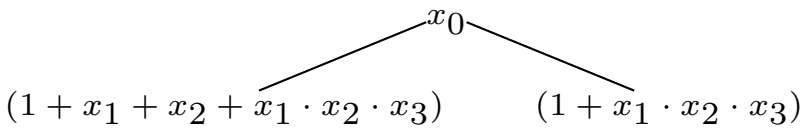
Pode-se sempre escrever

$$f(x_0, x_1, x_2, x_3) = x_0 \cdot f(1, x_1, x_2, x_3) + (1 + x_0) \cdot f(0, x_1, x_2, x_3)$$

ou seja, com alguma manipulação

$$= x_0 \cdot (1 + x_1 + x_2 + x_1 \cdot x_2 \cdot x_3) + (1 + x_0) \cdot (1 + x_1 \cdot x_2 \cdot x_3)$$

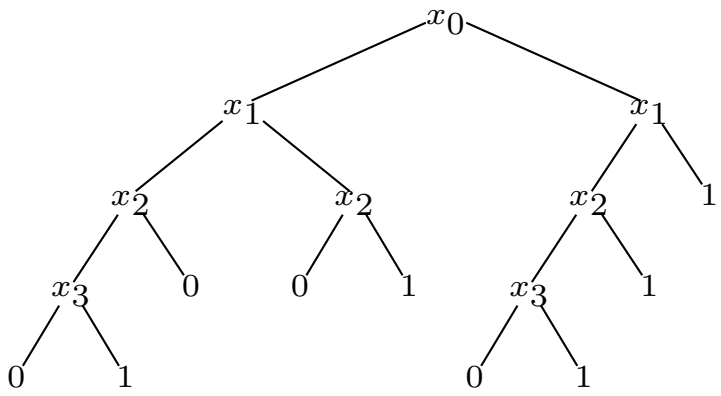
O que sugere uma representação arbórea



Repetindo o processo para as restantes variáveis, vemos que

$$(1 + x_1 + x_2 + x_1 \cdot x_2 \cdot x_3) = x_1 \cdot x_2 \cdot (1 + x_3) + (1 + x_1) \cdot (1 + x_2)$$
$$(1 + x_1 \cdot x_2 \cdot x_3) = x_1 \cdot (x_2 \cdot (1 + x_3) + (1 + x_2) \cdot 1) + (1 + x_1) \cdot 1$$

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".

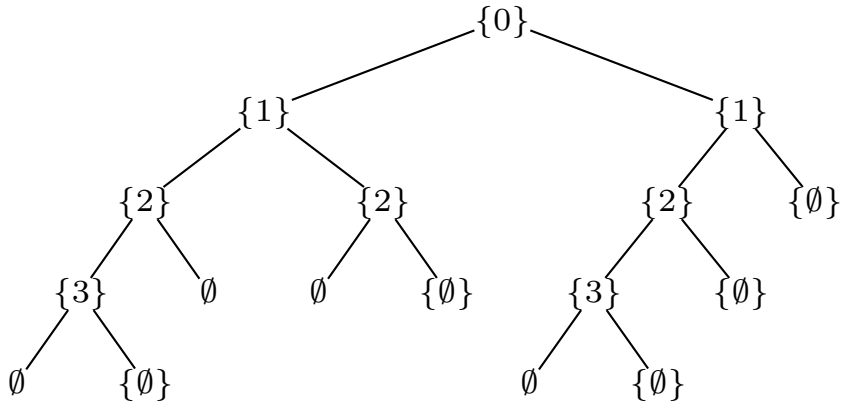
Nesta árvore, por exemplo, os seguintes percursos terminam no valor 0

$$\{x_0 = 1 \ , \ x_1 = 0 \ , \ x_2 = 1 \ , \ x_3 = ?\} \quad , \quad \{x_0 = 1 \ , \ x_1 = 1 \ , \ x_2 = 0 \ , \ x_3 = ?\}$$
$$\{x_0 = 1 \ , \ x_1 = 1 \ , \ x_2 = 1 \ , \ x_3 = 1\} \quad , \quad \{x_0 = 0 \ , \ x_1 = 1 \ , \ x_2 = 1 \ , \ x_3 = 1\}$$

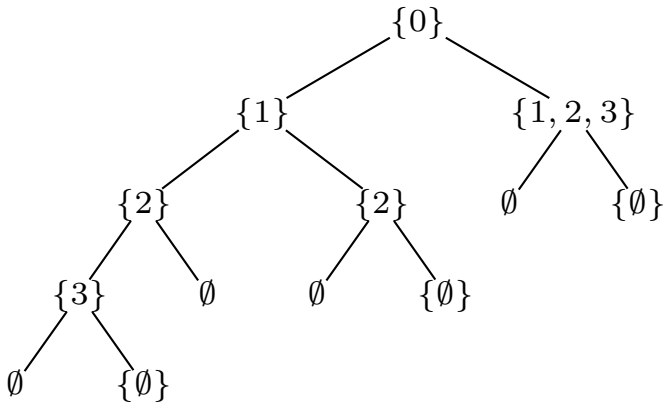
Todos os restantes percursos terminam no valor 1. Isto diz-nos que

$$f(1,0,1,0) = f(1,0,1,1) = f(1,1,0,0) =$$
$$= f(1,1,0,1) = f(1,1,1,1) = f(0,1,1,1) = 0$$

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

$$\mathcal{A} \ = \ \mathcal{A}^- \uplus \{\sigma\} \uplus (\mathcal{A}^+ \uplus \mathcal{A}^-)$$

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 $k \leq i < n$.

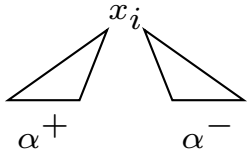
DEFINIÇÃO 42 [33] *Dado um polinómio reduzido (sem monómios repetidos)* $f = \sum_{u \in \mathcal{A}} x_u$ *com* $\mathcal{A} \subseteq \mathbb{U}_{k,n}$ *o* **fraccionamento** *de* f *em* $\mathbb{U}_{k,n}$ *é:*

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 $\langle x_i \ , \ \alpha^+ \ , \ \alpha^- \rangle$ (com $i \geq k$) formados por uma variável e dois outros fraccionamentos.



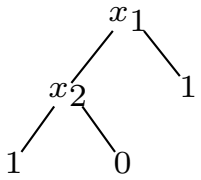
[33] Baseada na noção de fraccionamento ("*split*")de Davis-Putnam.

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.



TEOREMA 4 *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*

$$\mathcal{A}^- = \{ u \mid u \in \mathcal{A} \ \& \ k \notin u \}$$

$$\mathcal{A}^+ = \{ u \setminus \{k\} \mid u \in \mathcal{A} \}$$

$$\mathcal{A} = \mathcal{A}^- \uplus (\mathcal{A}^+ \uplus \mathcal{A}^-) \oplus \{\{k\}\}$$

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 $(f+g)$, $(f\cdot g)$, $f^{(z)}$.

FACTO 19 *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*

$$\alpha^{(z)} = \begin{cases} \langle x_k\ ,\ (\alpha^+)^{(z)}\ ,\ (\alpha^-)^{(z)}\rangle & se\quad z_k=0 \\ \langle x_k\ ,\ (\alpha^-)^{(z)}\ ,\ (\alpha^+)^{(z)}\rangle & se\quad z_k=1 \end{cases}$$

□

Frequentemente as funções boolenas aparecem agrupadas em vectores. Uma função booleana vectorial $\mathbf{S}:\mathrm{GF}(2)^n\to\mathrm{GF}(2)^n$ pode ser descrita por um vector de n funções booleanas escalares

$$\mathbf{S}(x_1,x_2,\cdots,x_n) = \begin{bmatrix} h_1(x_1,x_2,\cdots,x_n) \\ h_2(x_1,x_2,\cdots,x_n) \\ \cdots \\ h_n(x_1,x_2,\cdots,x_n) \end{bmatrix}$$

Na terminologia criptográfica, uma tal função é designada por uma $n \times n$ **S-Box**.

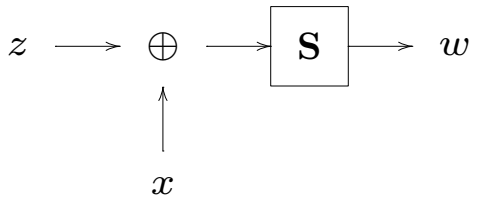
Facilmente se generaliza o conceito para S-Boxes não quadradas: uma $n \times m$ **S-Box** é uma função $\mathbf{S} : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)^m$ definida por um vector de m componentes em que, cada uma, é uma função $h_i : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$, com $i \in 0..m-1$.

Exemplo 35 : 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



(73)

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

$$\mathbf{S}(z \oplus x) = w \tag{74}$$

(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 36 : Recuperemos a **SBox** 3×3 do exemplo **??** e suponhamos o seguinte par *entrada-saída*

$$z = (1, 0, 1) \qquad w = (0, 1, 0)$$

A equação que resulta de (**??**) (com $w_0 = 0, w_1 = 1, w_2 = 0$) será

$$h_0(z \oplus x) \cdot (1 + h_1(z \oplus x)) \cdot h_2(z \oplus x) = 1$$

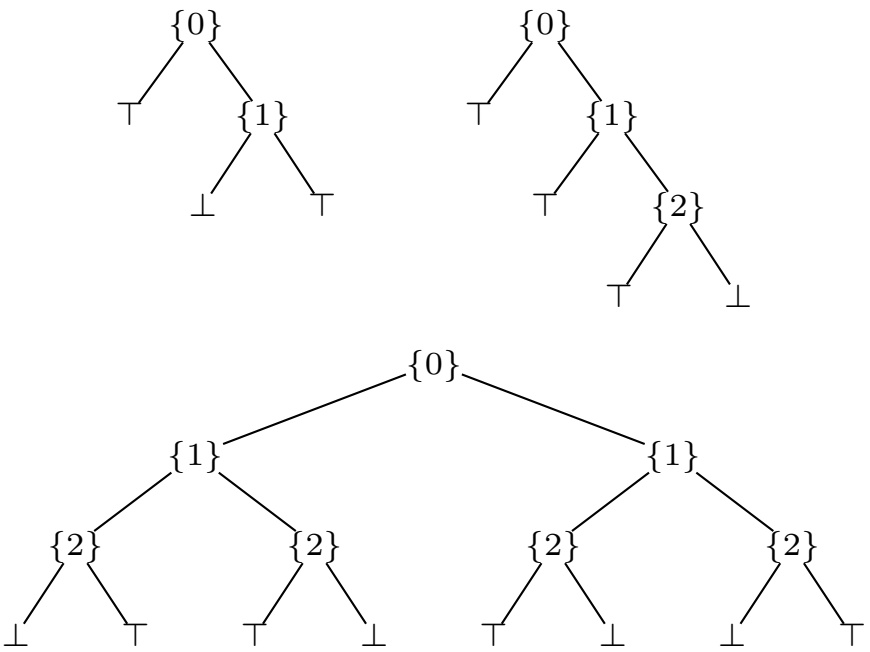
expandindo

$$(1 + (1 + x_0) \cdot x_1) \cdot (1 + (1 + x_0) \cdot (1 + x_1) \cdot (1 + x_2)) \cdot$$
$$\cdot (1 + (1 + x_0) + x_1 + (1 + x_2)) = 1$$

simplificando

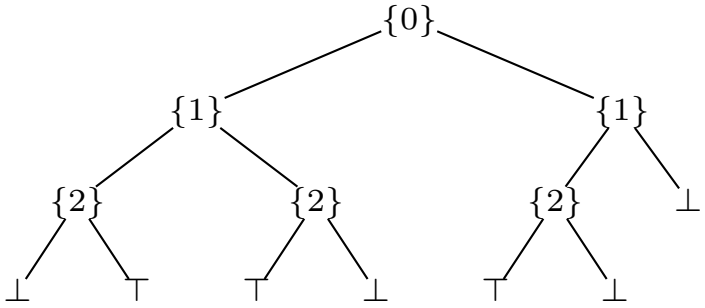
$$(1 + x_1 + x_0 \cdot x_1) \cdot (1 + (1 + x_0) \cdot (1 + x_1) \cdot (1 + x_2)) \cdot (1 + x_0 + x_1 + x_2) = 1$$

Os espectros dos três factores nesta equação serão (abrev. $\top \equiv \{\emptyset\}$, $\bot \equiv \emptyset$),



Usando as regras no facto **??** o espectro resultante será

(75)



Esta árvore permite resolver os problemas (**I**), (**II**) e (**III**), neste caso, usando o resultado seguinte

FACTO 20 *Seja* $f : \mathsf{GF}(2) \to \mathsf{GF}(2)$ *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* **??**. *Então:*

1. $\mathsf{supp}(f) = \emptyset$ *se e só se* $\alpha \equiv \bot$.
2. $x \in \mathsf{supp}(f)$ *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 $\mathsf{supp}(f)$ 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 $x \in \mathsf{supp}(f)$.

Conhecido o fraccionamento α se se procurar soluções para os problemas básicos, teremos:

1. *Problema de tipo* ***I****:* saber se $\mathsf{kwd}(f) = \emptyset$ reduz-se a saber se $\alpha = \bot$.

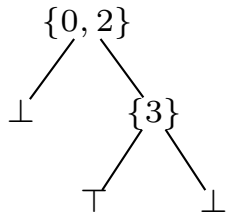
2. *Problema de tipo **II**:* se $\alpha \neq \perp$ encontrar um elemento arbitrário $x \in \text{kwd}(f)$ resolve-se gerando aleatoriamente um caminho válido em α. Isto significa criar um caminho que se inicie na raiz de α e siga, com igual probabilidade, por qualquer dos seus sub-fraccionamentos distintos de $\perp$.
3. *Problema de tipo **III**:* gerar todo o conjunto $\text{kwd}(f)$ 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 $\text{kwd}(f)$ 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 (**??**) é $\delta(w \oplus \mathbf{S}(z \oplus x)) = 1$, ou, equivalentemente,

$$\delta^{(w)}(\mathbf{S}^{(z)}(x)) = 1 \qquad \text{ou} \qquad [h^{(z)}]_{\bar{w}} = 1 \tag{76}$$

Podemos definir uma função boolena

$$\mathcal{Q}_{z,w}(x) \doteq \delta^{(w)}(\mathbf{S}^{(z)}(x)) \tag{77}$$

e a solução de um problema directo (procurar a chave x) como um ataque a $\mathcal{Q}_{z,w}$. Recuperando a definição de entropia de um ataque (ver página **??**) faz sentido definir:

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 $\mathcal{Q}_{z,w}$ (definida por (**??**)) 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 (**??**) 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 5.1..1 Seja μ um sub-conjunto de cardinalidade N do conjunto

$$\Omega \doteq \{ (z, w) \mid \mathbf{S}(z \oplus k) = w \} \tag{78}$$

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 (**??**), que representa o facto de, para todos os pares $(z, w) \in \mu$, se verificar $\mathcal{Q}_{z,w}(x) = 1$

$$\begin{aligned} \mathcal{Q}_{\mu}(x) &\doteq \Pi_{(z,w)\in\mu}\, \mathcal{Q}_{z,w}(x) = \\ &= \Pi_{(z,w)\in\mu}\, \delta(w \oplus \mathbf{S}(z \oplus x)) \end{aligned} \tag{79}$$

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 $\mathcal{Q}_{\mu}$.

Usando a notação da página **??**, tem-se

$$-\log_2\{\mathcal{Q}_\Omega : \mathcal{Q}_\mu\}_\phi \;=\; -\log_2 \mathrm{wt}(\mathcal{Q}_\Omega) + \log_2 \mathrm{wt}(\mathcal{Q}_\mu)$$

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 $\mathbf{S}$

$$\mathcal{H}(\mathcal{Q}_\Omega) \;=\; \min_\mu \left(\mathcal{H}(\mathcal{Q}_\mu) + \log_2 \mathrm{wt}(\mathcal{Q}_\mu) - \log_2 \mathrm{wt}(\mathcal{Q}_\Omega)\right) \qquad (80)$$

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 $\log_2 \mathrm{wt}(\mathcal{Q}_\mu)$.

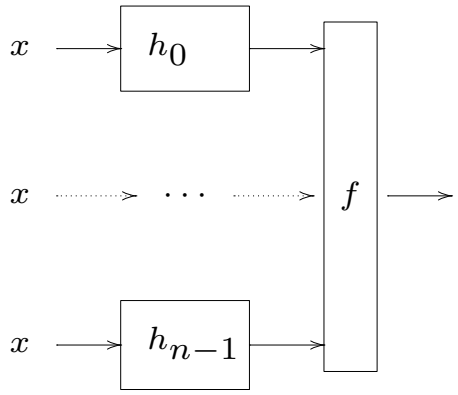
5.2.Composição de funções booleanas

Considere-se uma uma S-Box $k \times n$, $H : \mathrm{GF}(2)^k \longrightarrow \mathrm{GF}(2)^n$. Pode-se escrever $H = (h_0, \ldots, h_{n-1})$ em que os vários h_i são funções boolenas de k argumentos.

$$h_i : \mathrm{GF}(2)^k \longrightarrow \mathrm{GF}(2)$$

Seja $f : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$ 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

$$f \circ H \quad \text{ou} \quad H; f$$

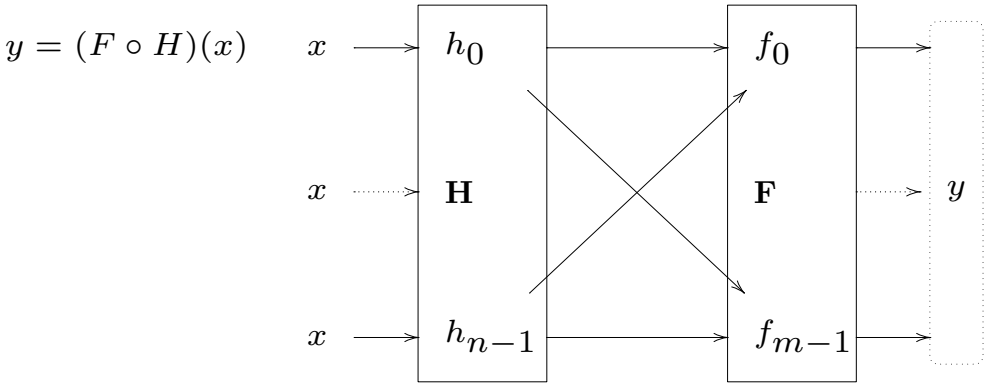
□

Considerando agora, não só uma função f, mas m funções deste tipo (formando uma S-Box de dimensão $n \times m$)

$$F = (f_0 \ , \ f_1 \ , \ \cdots \ , \ f_{m-1}) \qquad \text{com} \quad f_i : \mathrm{GF}(2)^n \longrightarrow \mathrm{GF}(2)$$

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 **??** (página **??**) a S-Boxes.

DEFINIÇÃO 43 *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 $\left(\boxplus_{i \in u}\, \mathcal{B}_i\right)$; 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

$$\biguplus_{u \in \mathcal{A}} \left(\boxplus_{i \in u}\, \mathcal{B}_i \right) \tag{81}$$

Funções Lineares

Infelizmente, no caso geral, a fórmula (**??**) 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 (**??**). Nomeadamente quando f é uma **função linear**.

Qualquer função linear $f : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$ 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 (**??**) 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

$$\mathrm{Tr}(x_0, x_1, \cdots, x_{n-1}) \doteq x_0 + x_1 + \cdots + x_{n-1} \tag{82}$$

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,

$$(c^f)_u = 1 \qquad \text{se e só se} \quad u \in \mathrm{Sp}(f) \tag{83}$$

Designamos c^f por **máscara** ou **paridade** de f. Nessas circunstâncias

FACTO 21 *Se f é linear e tem máscara c^f, então $f(x) = \mathrm{Tr}(x * c^f)$ para todo $x \in \mathrm{GF}(2)^n$.*

Exemplo: se for $f(x) = x_0 + x_2 + x_7$, com $x \in \mathrm{GF}(2)^8$, 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 $f(x)$.

5.3.Diferenças e Linearidade

Considere-se de novo uma S-Box $\mathbf{S}$ e o problema definido em (**??**) e (**??**) 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,

$$\mathbf{S}(a \oplus y) \oplus \mathbf{S}(y) = b \qquad \text{para todo} \quad y \in \mathrm{GF}(2)^n \tag{84}$$

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 5.3..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.

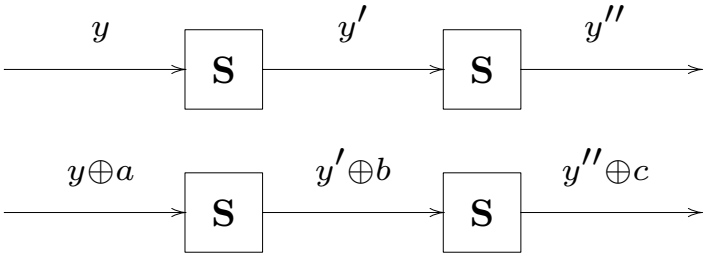
Note-se que:

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 $\mu_i = \{\, (z_j, w_j \oplus b_i) \mid j \in 1..\,\}$ 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 $y \oplus a$.



(85)

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

FACTO 22 *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

$$\Delta^{\mathbf{S}}_{a,b}(y) \;=\; \delta(\mathbf{S}(a \oplus y) \oplus \mathbf{S}(y) \oplus b) \tag{86}$$

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 (ver página **??**); a notação $\{\Delta^{\mathbf{S}}_{a,b}\}_{\varphi}$ 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 $\mathrm{GF}(2)^n$.

Usando (**??**) é possível exprimir a entropia de $\Delta^{\mathbf{S}}_{a,b}$ relativo a φ.

$$\mathcal{H}(\Delta^{\mathbf{S}}_{a,b})_{\varphi} \doteq \mathcal{H}(\varphi) - \log_2\{\Delta^{\mathbf{S}}_{a,b}\}_{\varphi}$$

com $\mathcal{H}(\varphi)$ determinado por (**??**).

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

$$\mathcal{H}(\Delta^{\mathbf{S}}_{a,b})_{\varphi} = -\log_2 \rho_{\mathbf{S}}(a, b) \qquad \text{ou} \qquad (87)$$

$$\rho_{\mathbf{S}}(a, b) = 2^{-\mathcal{H}(\Delta^{\mathbf{S}}_{a,b})_{\varphi}}$$

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 **??** para "candidatos" a pares de diferenças críticas

FACTO 23 $\quad \rho_{\mathbf{R}\circ\mathbf{S}}(a,c) \;\geq\; \rho_{\mathbf{S}}(a,b)\cdot\rho_{\mathbf{R}}(b,c)$

□

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

$$\mathcal{C}(f,g) \;\doteq\; 1 - 2^{-(n-1)}\cdot d(f,g) \tag{88}$$

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

FACTO 24 *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* $\mathcal{C}(g, \omega) = \mathcal{C}(f, \omega + \upsilon)$

DEFINIÇÃO 44 *Dada uma função boolena* $f : \mathsf{GF}(2)^n \to \mathsf{GF}(2)$ *e uma função linear* ω *, seja*

$$F(\omega) \doteq \mathcal{C}(f, \omega) \tag{89}$$

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***.*

FACTO 25 *Para todo* $f, g \in \mathsf{GF}(2)^n \to \mathsf{GF}(2)$ *verifica-se*

$$\mathcal{C}(f, g) = 2^{-n} \cdot \sum_x (-1)^{f(x)+g(x)} \tag{90}$$

e, se $F \doteq \mathcal{W}(f)$*, verifica-se para todo* $x \in \mathsf{GF}(2)^n$

$$(-1)^{f(x)} = \sum_\omega F(\omega)\,(-1)^{\omega(x)} \tag{91}$$

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*

$$(F \otimes G)(\omega) = \sum_\upsilon F(\omega \oplus \upsilon) \cdot G(\upsilon) \tag{92}$$

DEFINIÇÃO 45 *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*

$$\mathcal{C}^{\mathbf{S}}(\upsilon, \omega) \doteq \mathcal{C}(\upsilon \circ \mathbf{S}, \omega) \tag{93}$$

Vista como uma matriz $2^n \times 2^n$, *a função* $\mathcal{C}^{\mathbf{S}}$ *chama-se* **matriz de correlação** *de* $\mathbf{S}$.

FACTO 26 *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.*

$$\mathcal{C}^{\mathbf{R}\circ\mathbf{S}} = \mathcal{C}^{\mathbf{R}} \times \mathcal{C}^{\mathbf{S}}$$

5.4.Funções de argumento GF(2^n)

As funções $f : \mathrm{GF}(2^n) \rightarrow \mathrm{GF}(2^n)$ (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[34] por um polinómio de grau inferior a $2^n - 1$

$$f(x) \;=\; \sum_{i=0}^{2^n-1} a_i\, x^i \;, \qquad a_i \in \mathrm{GF}(2^n) \tag{94}$$

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:

$$f(x) \;=\; \begin{cases} a_0 & , \quad \text{se} \quad x = 0 \\ b_0 + \sum_{i=1}^{N} a_i\, x^i + b_i\, x^{-i} & , \quad \text{se} \quad x \neq 0 \end{cases} \tag{95}$$

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 37 : 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 ByteSub = Inv ∘ Afim. A primeira das quais é a função auto-inversa

$$\mathrm{Inv}(x) \;=\; \begin{cases} 0 & \text{se} \quad x = 0 \\ x^{-1} & \text{se} \quad x \neq 0 \end{cases} \tag{96}$$

[34] Qualquer função $f : \mathbb{F}_q \rightarrow \mathbb{F}_q$, 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 $q - 1$.

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 **??** 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).

$$x_\sigma \;\doteq\; (x \;,\; \sigma(x) \;,\; \sigma^2(x) \;,\; \cdots \;,\; \sigma^{n-1}(x)) \qquad (97)$$

2. As **funções boolenas** $f : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2)$ podem ser representadas pelas formas genéricas em (**??**) ou (**??**) tendo em atenção que o contradomínio $\mathrm{GF}(2)$ está imerso em $\mathrm{GF}(2^n)$.

exemplo: a função *traço* (ver definição **??** – página **??**), 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

$$f(x) \;=\; \sum_{k=0}^{n-1} a_k\,\sigma^k(x) \qquad \text{com} \quad a_k \in \mathcal{K} \tag{98}$$

Representanto por $\mathbf{a}$ o vector $(a_0, a_1, \cdots, a_{n-1})$, o polinómio (**??**) pode ser escrito como o produto escalar[35] de dois vectores

$$f(x) \;=\; \mathbf{a} \cdot x_\sigma \tag{99}$$

exemplo: 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 (**??**), 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

$$\mathbf{A}_{ij} = (a_k)^{2^i} \qquad \text{com} \quad k = j - i \pmod{n-1} \tag{100}$$

[35] 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.

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

$$f(x) \;=\; x_\sigma \cdot (\mathbf{A}\, y_\sigma) \tag{101}$$

sendo $\mathbf{A} \in \mathrm{GF}(2^n)^{n \times n}$ uma matriz com elementos em $\mathrm{GF}(2^n)$.
exemplo:
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** $g : \mathrm{GF}(2^n) \to \mathrm{GF}(2^n)$ têm a forma $g(x) = f(x,x)$ sendo $f(\cdot,\cdot)$ uma função bilinear. Isto é,

$$g(x) \;=\; x_\sigma \cdot \mathbf{A}\, x_\sigma \tag{102}$$

para uma matriz apropriada $\mathbf{A} \in \mathrm{GF}(2^n)^{n \times n}$.
exemplo:
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 $p_\mu : \mathrm{GF}(2^n) \to \mathrm{GF}(2)$, 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 $p_\mu(x)$

$$x \;=\; \sum_{\mu \in \mathcal{B}} p_\mu(x)\, \mu \tag{103}$$

Representemos por $(\cdot)^{\tilde{}} : \mathrm{GF}(2^n) \longrightarrow \mathrm{GF}(2)^n$ a função que associa x ao vector das suas projecções. Vectorialmente, (**??**) é

$$x \;=\; \mathcal{B} \cdot x^{\tilde{}} \tag{104}$$

Notas Algumas propriedades das projecções que derivam directamente de (**??**)

(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 **??**) vemos que elas coincidem; por isso é natural que existam semelhanças entre as duas noções. De facto é relativamente simples provar o seguinte resultado,

FACTO 27 *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* $\mathsf{GF}(2^n)$ *que será designada por* **base dual** *de* $\mathcal{B}$.

Sugestão de prova: Construa-se a matriz $n \times n$ de elementos $\mathbf{T}_{\mu\upsilon} \doteq \mathsf{tr}(\mu\,\upsilon)$; 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

$$\mathcal{B}' \;=\; \mathbf{T}^{-1}\,\mathcal{B} \tag{105}$$

em que $\mathbf{T}$ designa a matriz dos traços cruzados: $\mathbf{T}_{\mu\upsilon} = \mathsf{tr}(\mu\,\upsilon)$.

Exemplo 38 : Consideremos de novo $\mathsf{GF}(2^4)$ com uma base polinomial do tipo 0

$$\mathcal{B} = \{1\ ,\ \beta\ ,\ \beta^2\ ,\ \beta^3\}$$

Para calcular a matriz dos traços $\mathbf{T}_{ij} \doteq \mathsf{tr}(\beta^i \cdot \beta^j) = \mathsf{tr}(\beta^{i+j})$ 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
$\mathsf{tr}(\beta^i)$	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 $\mathcal{B}$)

$$\mathcal{B}' \;=\; \{1+\beta^3\ ,\ \beta^2\ ,\ \beta\ ,\ 1\}$$

e as projecções respectivas serão

$$p_1(x) = \mathrm{tr}(x) + \mathrm{tr}(x\,\beta^3) \quad , \quad p_\beta(x) = \mathrm{tr}(x\,\beta^2)$$

$$p_{\beta^2}(x) = \mathrm{tr}(x\,\beta) \quad , \quad p_{\beta^3}(x) = \mathrm{tr}(x)$$

o que permite concluir a identidade

$$x \;=\; \mathrm{tr}(x) + \mathrm{tr}(x\,\beta^3) + \mathrm{tr}(x\,\beta^2)\,\beta + \mathrm{tr}(x\,\beta)\,\beta^2 + \mathrm{tr}(x)\,\beta^3$$

□

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 $\mu \in \mathcal{B}$

$$p_\mu(x) = \mathrm{tr}(\mu'\,x) = \mathrm{tr}(\mu\,x') = p_{\mu'}(x') \tag{106}$$

Tomando como implícita a base $\mathcal{B}$, o operador $(\cdot)^{\tilde{}}$ associa cada elemento de $\mathrm{GF}(2^n)$ ao vector das suas projecções nessa base. Então verifica-se

$$x'^{\tilde{}} \;=\; \mathbf{T}^{-1}\,x^{\tilde{}} \quad , \quad x' \;=\; \mathcal{B}\cdot x'^{\tilde{}} \;=\; \mathcal{B}'\cdot x^{\tilde{}} \tag{107}$$

□

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

FACTO 28 *Se $\mathcal{B}, \mathcal{B}'$ são um par de bases duais então:*

1. $(\mathcal{B}')_\sigma = \mathbf{T}^{-1}\,\mathcal{B}_\sigma$
2. $x^{\tilde{}} = (\mathcal{B}')_\sigma\,x_\sigma \quad e \quad x_\sigma \;=\; (\mathcal{B}_\sigma)^t\,x^{\tilde{}}$

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 **??**. 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 **??**) 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* $\mathrm{GF}(2^n)$, *dada uma função* $f^\sim$ *de domínio* $\mathrm{GF}(2)^n$, *construir a função* f *de domínio* $\mathrm{GF}(2^n)$ *que verifica* $f^\sim(x^\sim) = f(x)^\sim$ *para todo* x.
> *Ou, inversamente, dada a função* f, *construir* $f^\sim$.

Note-se que: $f^\sim(x^\sim)$ denota a função de palavras de bits $f^\sim$ aplicada ao vector de bits que representa x (visto como elemento do corpo de Galois); $f(x)^\sim$ 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 $f^\sim : \mathrm{GF}(2)^n \to \mathrm{GF}(2)^n$ ou $f^\sim : \mathrm{GF}(2)^n \to \mathrm{GF}(2)$.

$f^{\tilde{}}(x^{\tilde{}}) = x^{\tilde{}} \oplus c^{\tilde{}}$ com $c \in \mathrm{GF}(2^n)$.

Esta é a função *branqueamento* ("whitening") que ocorre quase sempre nos andares das cifras. A função de domínio $\mathrm{GF}(2^n)$ equivalente é

$$f(x) = x + c$$

$f^{\tilde{}}(x^{\tilde{}}) = c^{\tilde{}}(x^{\tilde{}}) = \mathrm{Tr}(c^{\tilde{}} * x^{\tilde{}})$ com $c \in \mathrm{GF}(2^n)$

Forma genérica da função booleana linear; a função de domínio $\mathrm{GF}(2^n)$ equivalente é

$$f(x) = \mathrm{tr}(c' x) = (c')_\sigma \cdot x_\sigma$$

Prova: Simples utilização das identidades no facto **??**

$$f^{\tilde{}}(x^{\tilde{}}) = c^{\tilde{}} \cdot x^{\tilde{}} =$$

$$(\mathcal{B}_\sigma\,(c')_\sigma) \cdot (\mathcal{B}')_\sigma\, x_\sigma = (((\mathcal{B}')_\sigma)^t \mathcal{B}_\sigma\,(c')_\sigma) \cdot x_\sigma = (c')_\sigma \cdot x_\sigma$$

$f^{\tilde{}}(x^{\tilde{}}) = \mathbf{H}\, x^{\tilde{}}$ 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

$$f(x) = \mathbf{h} \cdot x_\sigma = \sum_{k=0}^{n-1} \mathbf{h}_k\, x^{2^k} \qquad (108)$$

em que $\mathbf{h} \in \mathrm{GF}(2^n)^n$ é o vector

$$\mathbf{h} = (\mathcal{B}')^t_\sigma\, \mathbf{H}^t\, \mathcal{B}$$

Prova:

$$f(x) = \mathcal{B} \cdot f^{\tilde{}}(x^{\tilde{}}) =$$

$$\mathcal{B} \cdot (\mathbf{H}\,(\mathcal{B}')_\sigma\, x_\sigma) = ((\mathcal{B}')^t_\sigma \cdot \mathbf{H}^t\, \mathcal{B}) \cdot x_\sigma$$

É importante ter-se em conta que (**??**), apesar da forma aparentemente complexa, é uma função linear. porque tem a forma prevista em (**??**).

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

$$\mathbf{h}\cdot x_\sigma \;=\; (\mathbf{H}^t\,\mathcal{B})\cdot(\mathcal{B}')_\sigma\,x_\sigma = (\mathbf{H}^t\,\mathcal{B})\cdot x^{\tilde{}}$$

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 39 : Regressando à cifra AES e ao exemplo **??**, 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 $\mathcal{B}_\sigma$

$$\mathbf{T} = (\texttt{05}\ ,\ \texttt{0A}\ ,\ \texttt{15}\ ,\ \texttt{2B}\ ,\ \texttt{57}\ ,\ \texttt{AE}\ ,\ \texttt{5C}\ ,\ \texttt{B9})$$

$$\mathbf{T}^{-1} = (\texttt{94}\ ,\ \texttt{0D}\ ,\ \texttt{1A}\ ,\ \texttt{A0}\ ,\ \texttt{65}\ ,\ \texttt{CA}\ ,\ \texttt{25}\ ,\ \texttt{2A})$$

$$\mathcal{B}_\sigma = \begin{bmatrix} \texttt{01} & \texttt{01} & \texttt{01} & \texttt{01} & \texttt{01} & \texttt{01} & \texttt{01} & \texttt{01} \\ \texttt{02} & \texttt{04} & \texttt{10} & \texttt{1B} & \texttt{5E} & \texttt{E4} & \texttt{4D} & \texttt{FA} \\ \texttt{04} & \texttt{10} & \texttt{1B} & \texttt{5E} & \texttt{E4} & \texttt{4D} & \texttt{FA} & \texttt{02} \\ \texttt{08} & \texttt{40} & \texttt{AB} & \texttt{B3} & \texttt{E8} & \texttt{1D} & \texttt{4A} & \texttt{EF} \\ \texttt{10} & \texttt{1B} & \texttt{5E} & \texttt{E4} & \texttt{4D} & \texttt{FA} & \texttt{02} & \texttt{04} \\ \texttt{20} & \texttt{6C} & \texttt{97} & \texttt{94} & \texttt{91} & \texttt{80} & \texttt{9A} & \texttt{C5} \\ \texttt{40} & \texttt{AB} & \texttt{B3} & \texttt{E8} & \texttt{1D} & \texttt{4A} & \texttt{EF} & \texttt{08} \\ \texttt{80} & \texttt{9A} & \texttt{C5} & \texttt{20} & \texttt{6C} & \texttt{97} & \texttt{94} & \texttt{91} \end{bmatrix}$$

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 $\texttt{E4} = \texttt{11100100}$ 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

$$f(x) = \quad 63 + 05 \cdot x^{254} + 09 \cdot x^{253} + F9 \cdot x^{251} + 25 \cdot x^{247} + \tag{109}$$
$$+ F4 \cdot x^{239} + 01 \cdot x^{223} + B5 \cdot x^{191} + 8F \cdot x^{127}$$

□

Finalmente vamos examinar a representação das **funções bilineares**

$$f(x, y) \quad = \quad x_\sigma \cdot \mathbf{A}\, y_\sigma \tag{110}$$

ou equivalentemente, com $\boldsymbol{\alpha} \doteq \mathcal{B}_\sigma\, \mathbf{A}\, \mathcal{B}_\sigma^t$

$$f(x, y) \quad = \quad \tilde{x} \cdot \boldsymbol{\alpha}\, \tilde{y} \tag{111}$$

Usando as identidades $x_\sigma = \mathcal{B}_\sigma^t\, \tilde{x}$ e $\boldsymbol{\alpha} = \mathcal{B}_\sigma\, \mathbf{A}\, \mathcal{B}_\sigma^t$ tem-se

$$f(x, y) = (\mathcal{B}_\sigma^t\, \tilde{x}) \cdot \mathbf{A}\, \mathcal{B}_\sigma^t\, \tilde{y} = \tilde{x} \cdot \mathcal{B}_\sigma\, \mathbf{A}\, \mathcal{B}_\sigma^t\, \tilde{y} = \tilde{x} \cdot \boldsymbol{\alpha}\, \tilde{y}$$

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

$$z_k = \tilde{x} \cdot \boldsymbol{\alpha}_k\, \tilde{y}$$

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 40 : 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

$$\boldsymbol{\alpha}_0 = \begin{bmatrix} 1000 \\ 1100 \\ 0000 \\ 0000 \end{bmatrix} \quad \boldsymbol{\alpha}_1 = \begin{bmatrix} 0000 \\ 1010 \\ 1000 \\ 0000 \end{bmatrix} \quad \boldsymbol{\alpha}_2 = \begin{bmatrix} 0000 \\ 0000 \\ 0000 \\ 0001 \end{bmatrix} \quad \boldsymbol{\alpha}_3 = \begin{bmatrix} 0010 \\ 0000 \\ 1000 \\ 0000 \end{bmatrix}$$

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 **??** (página **??**).

A base dual calculada nesse exemplo foi

$$\mathcal{B}' = \{1 + \beta^3\ ,\ \beta^2\ ,\ \beta\ ,\ 1\}$$

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

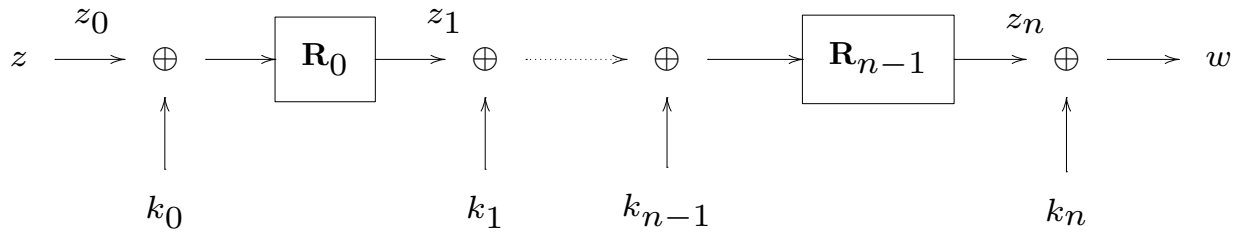
$$z_k \;=\; \sum_{ij} \boldsymbol{\alpha}_k^{ij} \, x_i \, x_j \tag{112}$$

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 $x_i \, x_j$.

5.5.Criptoanálise Algébrica

As principais cifras simétricas[36] modernas têm a seguinte estrutura



(113)

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

$$z_0 = z \quad , \quad z_{i+1} = \mathbf{R}_i(z_i \oplus k_i) \qquad w = z_n \oplus k_n \tag{114}$$

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 $i \in 0..n-1$

$$w_0 = w \qquad w_{i+1} = \mathbf{R}'_i(w_i \oplus k'_i) \qquad z = w_n \oplus k'_n$$

[36] Cifras que usam a mesma chave para cifrar e decifrar.

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[37]. 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 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[38].
  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.

[37] 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.

[38] 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.

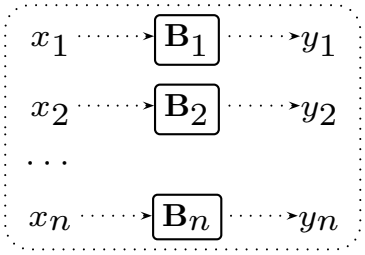
Idealmente os objectivos de "boa mistura" e "boa difusão" deveriam ser assegurados por um "design" único das SBoxes. No entanto, se atendermos que é necessário assegurar também boas propriedades computacionais numa função não-linear invertivel, este "design único" torna-se muito difícil.

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 (**??**) 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.

$$\begin{cases} z_0 & = z \\ z_n \oplus k_n & = w \\ z_{i+1} \oplus \mathbf{R}_i(z_i \oplus k_i) & = 0 \qquad i \in 0..n-1 \end{cases} \tag{115}$$

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

$$\begin{cases} z_0 & = z \\ z_n + k_n & = w \\ \mathbf{F}_i(z_{i+1}, z_i + k_i) & = 0 \qquad i \in 0..n-1 \end{cases} \tag{116}$$

em que as funções $\mathbf{F}_i(\cdot,\cdot)$ têm uma estrutura algebricamente muito mais símples do que os $\mathbf{R}_i(\cdot)$.

Exemplo 41 : 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 (**??**) e (**??**) (página **??**) 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

$$1 \;=\; \boldsymbol{\alpha} \cdot X \tag{117}$$

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 (**??**) 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?

6.Curvas Elípticas

O objectivo desta secção é o estudo de uma classe de grupos cíclicos particularmente importante em Criptografia: os grupos das **curvas elípticas**.

Essencialmente cada grupo é formado por pontos de uma curva plana definida por um polinómio $f(x, y)$ a duas variáveis: cada ponto é um par $P = (x, y)$ que verifica $f(x, y) = 0$.

Sobre esse conjunto de pontos associa-se uma construção geométrica que permite, a partir de dois pontos arbitrários P e Q obter um terceiro ponto, representado por $P + Q$; a adição que iremos definir vai ter as propriedades de operação de grupo.

Certos sub-grupos do grupo de pontos assim formado têm a estrutura de grupo cíclico onde a "exponenciação" é computacionalmente tratável mas o "logaritmo discreto" é um problema defícil.

Como as nossas construções são essencialmente polinomiais não é necessário por restricções nas coordenadas x e y para além de dizer que devem pertencer a um corpo apropriado $\mathbb{K}$. Por isso, é conveniente iniciar o estudo das curvas elípticas com a apresentação de certas noções gerais da de Geometria Algébrica sobre corpos arbitrários.

6.1.Grupos

Um **grupo** $\mathbb{G} = \langle G, \cdot, \varepsilon \rangle$ é um conjunto $G \neq \emptyset$ equipado com um elemento $1 \in G$ (a "unidade") e uma operação $\_ \cdot \_ : G \times G \to G$ associativa tal que, para todo $x \in G$: *(i)* $x \cdot \varepsilon = \varepsilon \cdot x = x$ e *(ii)* existe $y \in G$ tal que $x \cdot y = y \cdot x = \varepsilon$.

O grupo é **abeliano** se a operação é comutativa.

Exemplo 42 : Considere-se um qualquer conjunto X com n elementos representados por "índices" $1, 2, \ldots, n$. Uma permutação é uma representação dos elementos do conjunto numa sequência de índices; por exemplo, para $n = 4$, as seguintes sequências são permutações:

$$1, 2, 3, 4 \qquad 4, 3, 2, 1 \qquad 2, 1, 4, 3 \qquad 4, 2, 3, 1$$

Já as seguintes sequências não são permutações

$$1, 1, 2, 3, 4 \qquad 1, 3, 4 \qquad 2, 2, 2, 2 \qquad \ldots$$

ou porque um elmento aparece repetido ou então não aparece.

A primeira das permutações anteriores mantém os elementos na ordem inicial; é a chamada *permutação identidade*; a segunda inverte a sua ordem; a terceira transpõe[39] o primeiro e o segundo elementos e transpõe o terceiro e o quarto. A permutação transpõe o primeiro com o quarto elemento.

Duas permutações podem ser "compostas" por aplicação sucessiva dessas ordenações; por exemplo as permutações $2, 1, 3, 4$ e $1, 2, 4, 3$ são duas transposições; a sua composição resulta da aplicação sucessiva destas duas transformações

$$2, 1, 3, 4 \cdot 1, 2, 4, 3 = 2, 1, 4, 3$$

Esta operação não é comutativa como claramente se vê a partir de $2, 1, 3, 4$ e $1, 3, 2, 4$.

$$2, 1, 3, 4 \cdot 1, 3, 2, 4 = 2, 3, 1, 4 \neq 1, 3, 2, 4 \cdot 2, 1, 3, 4 = 3, 1, 2, 4$$

O grupo gerado por estas permutações de n elementos, com a unidade definida pela permutação vazia e com a operação $\cdot$ definida pela composição, chama-se **grupo simétrico de ordem** n e representa-se por $\mathbb{S}_n$.

[39] Chamamos **transposições** às permutações que se limitam a trocar a ordem entre dois elementos.

De uma forma genérica, define-se **permutação** num qualquer conjunto finito X como uma qualquer função bijectiva $\pi : X \longrightarrow X$. O chamado **grupo simétrico** de X, representado por $\mathbb{S}_X$, tem por elementos estas permutações e por operação de grupo a composição de funções; o elemento neutro é, obviamente, a função identidade.

É facilmente verificável que, se o número de elementos $|X|$ é n, então o número de permutações distintas realizadas sobe X elementos é $n!$. Por isso $|\mathbb{S}_X| = n!$.

Uma transposição entre inteiros i e j representa-se por $(i\,j)$; por exemplo $(1\,2)$ é uma representação para a permutação $2, 1, 3, 4$, enquanto que $(3\,4)$ denota a permutação $1, 2, 4, 3$. Então $(1\,2) \cdot (3\,4)$ denota a permutação $2, 1, 4, 3$.

A noção de transposição pode ser generalizada para qualquer número de inteiros e, nessas circunstâncias passa a designar-se por **ciclo**; por exemplo, $(1\ 2\ 4)$ é um ciclo de 3 elementos que muda o primeiro elemento para a posição 2, muda o segundo elemento para a posição 4 e muda o quarto elemento de volta à posição 1; i.e, a permutação $4, 1, 3, 2$. Genericamente $(\,)$ denota o ciclo vazio (permutação identidade) e $(i_1\, i_2\, \ldots\, i_k)$ denota a permutação $i_i \mapsto i_2 \mapsto i_3 \mapsto \cdots i_k \mapsto i_1$.

Um famoso teorema da teoria dos grupos diz-nos que:

Toda a permutação é a soma de ciclos disjuntos (i.e. sem elementos comuns). Adicionalmente a soma é única a menos da ordem das parcelas e da inclusão de ciclos com 0 *ou* 1 *elementos (ditos* ciclos nulos*).*

Note-se que a soma de dois ciclos disjuntos α e β é comutativa: i.e. $\alpha + \beta = \beta + \alpha$. Note-se também que ciclos com 0 ou 1 elementos são equivalentes à identidade e, por isso, não afectam as somas.

O grupo $\mathbb{S}_n$ não é, normalmente, cíclico mas vai ser essencial ao estudo de outros grupos que tenham a propriedade de serem cíclicos; e isso deriva das propriedades dos sub-grupos de $\mathbb{S}_n$. Nomeadamente

Todo o grupo $\mathbb{G}$ *de ordem finita* n *é isomórfico com um sub-grupo de* $\mathbb{S}_n$.

Para cada $z \in G$ a função $\lambda_z : x \mapsto z \cdot x$ é uma bijecção e portanto um elemento de $\mathbb{S}_G \simeq \mathbb{S}_n$. Facilmente se verifica que o morfismo $z \mapsto \lambda_z$ é um homomorfismo injectivo.

Uma das questões tradicionais da álgebra (com origem em Lagrange, Galois, etc. . .) é o estudo do comportamento de funções de raízes de um polinómio quando o conjunto das raízes é sujeito a uma permutação.

Concretizando: considere-se um qualquer polinómio de coeficientes racionais

$$\mathrm{p}[X] \;=\; a_0 + a_1\,X + \ldots + a_n\,X^n \qquad a_i \in \mathbb{Q}$$

e uma função $f : \mathbb{C}^n \longrightarrow Y$ de n variáveis complexas e valores sobre um conjunto Y.

Vamos representar por $\tilde{x} = (x_1, x_2, \cdots, x_n) \in \mathbb{C}^n$ uma sequência de n números complexos e por $f\,\tilde{x} \doteq f(x_1, x_2, \ldots, x_n)$ o resultado de aplicar $f(\cdot)$ a essa sequência de argumentos.

A pergunta fundamental colocada por Lagrange e Galois é:

> No caso particular em que $\tilde{x}$ uma sequência das n raízes do polinómio $\mathrm{p}[X]$ (repetidas caso sejam múltiplas), aplicando uma permutação $\sigma \in S_n$ a $\tilde{x}$ obtém-se um novo arranjo $\tilde{x}_\sigma$ das mesmas raízes; como se compara $f\tilde{x}$ com $f\tilde{x}_\sigma$?
> Por exemplo: são iguais? se $f\tilde{x}$ for racional será que $f\tilde{x}_\sigma$ ainda é racional? etc. . .

Para cada permutação $\sigma \in S_n$ vamos denotar por $f_\sigma : \mathbb{C} \longrightarrow Y$ a função que se obtém de f permutando os seus argumentos de acordo com σ ; i.e. $f_\sigma\,\tilde{x} = f\,\tilde{x}_\sigma$. A função f diz-se **simétrica** se $f_\sigma \equiv f$ para todo σ .

Exemplo 43 : Tome-se uma qualquer função complexa $f : \mathbb{C}^3 \longrightarrow Y$ da forma

$$f : (x_1, x_2, \cdots, x_n) \mapsto h(x_1) \star h(x_2) \star \cdots \star h(x_n) \qquad (118)$$

em que $(\star) : Y^2 \longrightarrow Y$ é qualquer operador binário comutativo em Y e $h : \mathbb{C} \longrightarrow Y$ é uma outra função sobre Y de uma só variável complexa.

Esta função f é sempre simétrica porque, premutando os argumentos de qualquer forma, o facto de $\star$ ser comutativo assegura que o resultado se mantém invariante.

Como exemplos de funções nesta classe são a soma e multiplicação dos argumentos; i.e.

$f\tilde{x} = \sum_i x_i$ ou $f\tilde{x} = \prod_i x_i$ e a construção de um polinómio a partir das raízes:

$$f\tilde{x} \doteq \prod_i (X - x_i)$$

Neste último caso h é a função que constrói monómios $h : x \mapsto (X - x)$ e $(*)$ é a multiplicação de polinómios.

Uma generalização da forma (**??**) são as funções da forma

$$f\tilde{x} \doteq \sum_{\gamma \in \Gamma} h_\gamma \, \tilde{x} \tag{119}$$

em que $\Gamma \subseteq S_n$ é um sub-grupo de S_n, $(+) : Y^2 \to Y$ é comutativo e $h : \mathbb{C}^n \to Y$ é uma função com a propriedade de, para todo $\sigma \in S_n$ verifica-se $h_\sigma = h_\gamma$ para algum $\gamma \in \Gamma$.

□

Como exemplo de função não simétrica temos

$$g : (x_1, x_2, \cdots, x_n) \mapsto \prod_{i<j} (x_i - x_j) \tag{120}$$

É fácil verificar que, para um dado σ, se verifica (neste caso) $g_\sigma = \pm g$ isto porque, para cada $i \neq j$, se o factor $(x_i - x_j)$ ocorre em g então ou ocorre também em g_σ ou, se não ocorrer, ocorre o factor $(x_j - x_i)$.

Se f for simétrica a resposta às perguntas anteriores é trivial já que $f\tilde{x} = f\tilde{x}_\sigma$ independentemente do facto de $\tilde{x}$ denotar as raízes do polinómio ou não. A questão interessante é saber a resposta quando f não é simétrica mas $\tilde{x}$ é formado pelas raízes do polinómio; isso vai permitir estabelecer uma relação entre a função f e o polinómio propriamente dito.

Se a função f não for simétrica pode ainda acontecer que, para alguns σ se verifique $f_\sigma = f$. Põe-se então a questão de localizar essas permutações especiais. Assim é costume definir $\mathcal{S}(f) \doteq \{ \sigma \in S_n \mid f_\sigma = f \}$ como o sub-grupo de S_n formado por todas as permutações que **fixam** f.

Exemplo 44 : Considere-se o polinómio $X^3 - 1$ que tem 3 raízes apresentadas na sequência

$$\tilde{x} = \left(1 \ , \ \frac{-1 + i\sqrt{3}}{2} \ , \ \frac{-1 - i\sqrt{3}}{2} \right)$$

Com 3 raízes tem-se o grupo de permutações S_3 com $3! = 6$ elementos:

$$(\,) \ , \ (1\,2) \ , \ (1\,3) \ , \ (2\,3) \ , \ (1\,2\,3) \ , \ (1\,3\,2)$$

Estas permutações serão aplicadas sucessivamente à sequência inicial $\tilde{x}$. Por exemplo a sequência $\tilde{x}_{(1\,2)}$ será

$$\tilde{x}_{(1\,2)} = \left(\frac{-1 + i\sqrt{3}}{2} \ , \ 1 \ , \ \frac{-1 - i\sqrt{3}}{2} \right)$$

□

Seja $\mathbb{G} = \langle G, \cdot, \varepsilon \rangle$ um grupo, seja X um conjunto qualquer e $\gamma : G \times X \longrightarrow X$ – representando-se $\gamma(g, x)$ por $g\,x$ – uma função binária que verifica as propriedades: *(i)* $\varepsilon\,x = x$ *(ii)* $g(h\,x) = (g \cdot h)\,x$.

Nestas circunstâncias diz-se que γ é uma **acção**, que X é um $\mathbb{G}$-conjunto e que o grupo $\mathbb{G}$ age sobre X.

A cada $g \in G$ associamos função $\tilde{\gamma}(g) : x \mapsto g\,x$. A função $\tilde{\gamma}(g)$ é injectiva e tem por inversa a função $\tilde{\gamma}(g^{-1})$; isto porque $g^{-1}\,(g\,x) = (g^{-1} \cdot g)\,x = \varepsilon\,x = x$.

Portanto $\tilde{\gamma} : G \longrightarrow S_X$ associa a cada elemento do grupo uma permutação em X.

Nota: De facto $\tilde{\gamma}$ é um homomorfismo de grupos e, desta forma, é possível generalizar a observação no final do exemplo anterior, da forma seguinte

Se X é um $\mathbb{G}$-conjunto, então $\mathbb{G}$ identifica-se, a menos de um isomorfismo, com um sub-grupo de S_X.

Note-se que cada grupo $\mathbb{G}$ age sobre si próprio com a operação do grupo: $\gamma(x, y) = x \cdot y$.

Exemplo 45 : Outro grupo com importância particular no estudo das curvas elípticas é o chamado **grupo modular** Γ .

Tome-se uma função racional complexa

$$g(z) \;=\; (a\,z+b)/(c\,z+d) \tag{121}$$

em que a, b, c, d são inteiros e $(a\,d - c\,d = 1)$.

É conveniente caracterizar uma forma modular $g(\cdot)$ pela "matrix dos coeficientes" $M_g \doteq \begin{pmatrix} a & b \\ c & d \end{pmatrix}$. Temos sempre $\det(M_g) = a\,d - c\,b = 1$, por hipótese.

O conjunto de todas as matrizes 2×2 de inteiros e com determinante igual a 1, equipado com a operação de multiplicação de matrizes forma um grupo, chamado **grupo linear simples**, e representa-se por $SL_2(\mathbb{Z})$. É também simples verificar que $M_{g^{-1}} \equiv (M_g)^{-1}$ e que $M_{g\circ f} \equiv M_g\,M_f$. Portanto as funções $g(\cdot)$ da forma (**??**) injectivas numa região apropriada X do plano complexo $\mathbb{C}$, com a operação de composição e com a função identidade como unidade formam um grupo a que chamamos o **grupo modular** Γ . As funções tomam o nome de **formas modulares**.

Pode-se portanto afirmar que cada matriz $M \in SL_2(\mathbb{Z})$ determina um endomorfismo da forma (**??**) numa sub-região X de $\mathbb{C}$ onde todas as funções $g(\cdot)$ sejam bem definidas. Porém duas matrizes podem definir a mesma forma modular. Note-se que a matriz $M' \doteq (-1)\,M$ também pertence a $SL_2(\mathbb{Z})$ e define o mesmo endomorfismo

$$\det(M') = (-1)^2 \cdot \det(M) = 1 \qquad \frac{(a\,z+b)}{(c\,z+d)} \;=\; \frac{(-a\,z-b)}{(-c\,z-d)}$$

Por isso, o grupo modular Γ identifica-se com $SL_2(\mathbb{Z})$ a menos da equivalência $M \sim (-1)\,M$. Este será o grupo quociente $SL_2(\mathbb{Z})/\{1,-1\}$.

Falta esclarecer qual é a sub-região $X \subset \mathbb{C}$ na qual os elelementos $g \in \Gamma$ definem funções injectivas. Várias hipóteses são possíveis:

1. Pode-se considera o semi-plano complexo superior $\mathbb{H} \doteq \{\, z \in \mathbb{C} \mid \Im(z) > 0 \,\}$. Note-se que, ao tomar argumentos z com parte imaginária positiva a imagem tem parte imagínária positiva; de facto, fazendo $z = x + i\,y$, com $y > 0$, e expandindo $g(z)$, verifica-se que $\Im(g(z)) = y/\lambda$ com $\lambda > 0$.
   Note-se que os polos dos vários $g(z)$ são números racionais e, portanto, estão excluídos de $\mathbb{H}$.

2. Em alternativa pode-se considerar a compactificação $\tilde{\mathbb{C}} \doteq \mathbb{C} \cup \{\infty\}$ que se obtém juntando ao plano complexo um único ponto de infinito[40].
   Em $\tilde{\mathbb{C}}$ define-se $g(\infty) = \lim_{z\to\infty} g(z) = a/c$ e $g(-b/c) = \infty$.

Portanto os elementos de Γ actuam tanto sobre o semi-plano $\mathbb{H}$ como sobre a compactificação $\tilde{\mathbb{C}}$. Em qualquer dos casos, deve-se considerar dois elementos

$$T(z) \doteq z+1 \qquad S(z) \doteq -1/z \tag{122}$$
$$M_T = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \qquad M_S = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

que determinam a estrutura do grupo modelar Γ. Os resultados essenciais são:

1. O grupo Γ é gerado pelos dois elementos T e S; i.e., qualquer elemento $g \in \Gamma$ pode ser escrito como uma palavra nos símbolos S, T, S^{-1} e T^{-1}.
2. $S^2 \sim (S\,T)^3 \sim (T\,S)^3 \sim 1$.

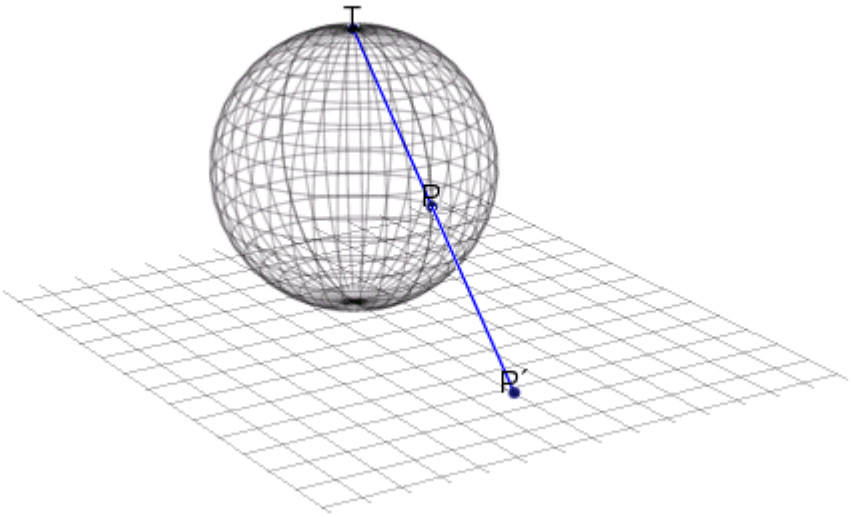


Figura 3: Esfera de Riemann

[40] $\tilde{\mathbb{C}}$ é a chamada *esfera de Riemann* e está representada na figura **??**. Cada ponto P da esfera associa-se a um ponto P' do plano fazendo passar uma recta pela *topo* T e por P; a intersecção dessa recta no plano determina P'. Por esta correspondência ve-mos que o único ponto ∞ no plano é a imagem do topo T da esfera.

Por vezes é necessário distinguir duas operações de grupo definidas no mesmo conjunto base G designado uma como "soma" e outra como "multiplicação". Os grupos respectivos passam a classificar-se como "aditivos" e "multiplicativos", respectivamente.

Um grupo aditivo é normalmente representado na forma $\mathbb{G} = \langle G, +, 0\rangle$ enquanto que um grupo multiplicativo é representado na forma $\mathbb{G} = \langle G, \cdot, 1\rangle$.

Seja $\mathbb{G} = \langle G, +, 0\rangle$ um grupo aditivo[41] onde a adição não é necessariamente comutativa; seja $g \in G$ e n um inteiro; o **produto escalar** $n\,g$ define-se como $\underbrace{g + g + \cdots + g}_{n \text{ vezes}}$, se for $n > 0$; $0\,g$ é definido como 0; se $n < 0$ então $n\,g$ define-se como $-((-n)\,g)$.

Algumas noções importantes ligadas ao produto escalar:

1. Para cada $g \in G$ o morfismo $\hat{g} : \mathbb{Z} \longrightarrow \mathbb{G}$ definido como $\hat{g}(n) = n\,g$ é um homomorfismo de grupos. Analogamente, cada $n \in \mathbb{Z}$ determina um endomorfismo $[n] : g \mapsto n\,g$ em $\mathbb{G}$.
2. A imagem $\hat{g}(\mathbb{Z}) = \{\, n\,g \mid n \in \mathbb{Z} \,\}$ representa-se por $\langle g\rangle$; é imediato verificar que $\langle g\rangle$ é um sub-grupo de $\mathbb{G}$.
3. Se $\hat{g}$ for surjectivo (i.e. o sub-grupo $\langle g\rangle$ coincide com a totalidade de $\mathbb{G}$) então $\mathbb{G}$ diz-se **cíclico** e **gerado** por g. Em qualquer dos casos o sub-grupo $\langle g\rangle$ é, ele próprio, um grupo cíclico.
4. A **ordem** de $\mathbb{G}$ é a cardinalidade do conjunto G. A **ordem** de um elememto $g \in G$ é a ordem do sub-grupo $\langle g\rangle$; i.e. é o menor $n > 0$, se existir, tal que $n\,g = 0$ ou ∞ se não existir.

Quando o grupo $\mathbb{G} = \langle G, +, 0\rangle$ é **abeliano** (i.e. a soma é comutativa)

[41] Analogamente considerarando um grupo multiplicativo $\mathbb{G} = \langle G, \cdot, 1\rangle$ (não necessariamente comutativo) e definir **exponenciação escalar** g^n a partir do caso base $g^n = g \cdot g \cdots g$, n vezes para $n > 0$. O morfismo de grupos $\hat{g} : \mathbb{Z} \longrightarrow \mathbb{G}$ define-se agora como $\hat{g}(n) = g^n$.

então surgem novos conceitos e propriedades. No que se segue os grupos são considerados aditivos e abelianos.

1. Os elementos de ordem finita de $\mathbb{G}$ formam um sub-grupo chamado $t\mathbb{G}$ chamado **sub-grupo de torsão** de $\mathbb{G}$.
2. Se $\mathbb{F} = \langle F, +, 0 \rangle$ for um grupo abeliano e se for a soma directa de grupos cíclicos de ordem infinita então diz-se um **grupo abeliano livre**. Pela definição deverá existir um subconjunto $\mathcal{B} \subset F$, chamado a **base** do grupo, formado por elementos $g \in \mathcal{B}$ de ordem infinita de tal forma que cada $x \in F$ se pode escrever como uma soma

$$x = \sum_{g \in \mathcal{B}} n_g \, g \qquad \text{com} \quad n_g \in \mathbb{Z}$$

3. $\mathbb{G}$ é de **geração finita** quando qualquer um dos seus sub-grupos livres tem base finita e **independente**.
   Nota: Um sub-conjunto finito $X = \{x_i\}_{i=1}^{n}$ de elementos de $\mathbb{G}$ é **independente** quando, para qualquer sequência de inteiros $\{k_i\}$, verifica-se $\left(\sum_i k_i \, x_i\right) = 0$ se e só se $(k_i \, x_i) = 0$ para todo $i \in 1..n$.
4. Seja p um primo; um grupo é p-**grupo** quando a ordem de cada um dos seus elementos é da forma p^n (para algum $n \geq 0$).
   Para um grupo $\mathbb{G}$ chama-se p-**componente primária**, e representa-se por $\mathbb{G}_p$, o seu maior sub-grupo que é um p-grupo.

   FACTO 29 (DECOMPOSIÇÃO PRIMÁRIA) *Cada grupo de torsão* $\mathbb{G}$ *é a soma directa das suas componentes* p*-primárias:* $\mathbb{G} \equiv \sum_p \mathbb{G}_p$.

6.2.Corpos, Extensões e Teoria de Galois

Seja $\mathbb{K}$ um corpo arbitrário. A sua **característica** $\text{char}(\mathbb{K})$ é o menor inteiro positivo p tal que $p\,1 = 0$. Se não existir nenhum p finito que verifique esta igualdade, então $\text{char}(\mathbb{K}) \doteq 0$. Se $p > 0$ existir então é necessariamente primo.

Uma **extensão** de $\mathbb{K}$ é um corpo $\mathbb{K}$' que contém $\mathbb{K}$ como sub-corpo. Um elemento de $\alpha \in \mathbb{K}'$ diz-se **algébrico** em $\mathbb{K}$ se é raíz de um polinómio não-nulo, de coeficientes em $\mathbb{K}$ e irredutível em $\mathbb{K}$. Se não for algébrico, α diz-se **transcendente** em $\mathbb{K}$. A extensão $\mathbb{K}$' diz-se **algébrica** quando qualquer elemento $\alpha \in \mathbb{K}'$ é algébrico em $\mathbb{K}$.

Exemplo 46 : O corpo dos racionais $\mathbb{Q}$, o corpo do reais $\mathbb{R}$ e o corpo dos complexos $\mathbb{C}$ são todos corpos de característica 0 relacionados por várias extensões.

$\mathbb{R}$ é uma extensão de $\mathbb{Q}$ e existem elementos de $\mathbb{R}$ que são algébricos em $\mathbb{Q}$. Por exemplo, $\sqrt{2}$ é um elemento[42] de $\mathbb{R}$ que é raíz do polinómio $(X^2 - 2)$ de coeficientes em $\mathbb{Q}$.

Porém existem muitos elementos de $\mathbb{R}$ que são transcendentes em $\mathbb{Q}$. A prova não é simples; por exemplo, só nos finais do século XIX é que se provou que tanto π como a base dos logaritmos neperianos e são transcendentes. Já no século XX provou-se que, para um qualquer a algébrico em $\mathbb{Q}$ (diferente de 0 ou 1) e para qualque não-racional b, o número a^b é sempre transcendente. Por exemplo, $2^{\sqrt{5}}$ é transcendente.

Uma raíz do polinómio (X^2+1) (que representamos normalmente por i) é um elemento algébrico numa extensão de $\mathbb{Q}$. Porém não pertence a $\mathbb{R}$ porque não é possível construir uma sequência de Cauchy em $\mathbb{Q}$ que tenha esse limite. A menor extensão de $\mathbb{Q}$ que contém i é o chamado corpo dos *racionais gaussianos* $\mathbb{Q}[i]$ (análogos aos números complexos mas com as componentes restrictas a racionais) e essa extensão é obviamente algébrica.

Se virmos agora a menor extensão de $\mathbb{Q}$ que contém $\mathbb{R}$ e i obtém-se, obviamente, os números complexos $\mathbb{C}$. De facto $\mathbb{C}$ é uma extensão quadrática de $\mathbb{R}$ gerada pela base $\{1, i\}$.

[42] $\sqrt{2}$ é um elemento de $\mathbb{R}$ porque é possível definir uma sequência de Cauchy em $\mathbb{Q}$ com esse limite.

O corpo $\mathbb{Z}_p$ (com p primo) é o exemplo mais simples de corpo com característica p. Aqui pode-se também colocar a questão de saber se $\mathbb{Z}_p$ contém as raízes do polinómio X^2+1 (raizes quadradas de -1); ou, equivalentemente, saber se este polinómio é irredutível no corpo.

Sabe-se que qualquer $a \in \mathbb{Z}_p$ é resíduo quadrático se e só se $1 = a^{(p-1)/2} \pmod p$; logo -1 tem raíz quadrada se e só se $(p-1)/2$ é par; por exemplo $\mathbb{Z}_5$ contém duas raízes de -1 mas já $\mathbb{Z}_7$ não contém qualquer raíz de -1. Por isso, se $(p-1)/2$ for ímpar existirá uma extensão quadrática de $\mathbb{Z}_p$, representada por $\mathbb{Z}_p[i]$, cujo elemento genérico tem a forma $a + b \cdot i$ com $a, b \in \mathbb{Z}_p$.

O **grupo de Galois** $G_{\mathbb{K}'/\mathbb{K}}$ é o grupo formado por todos os automorfismos[43] $\tau : \mathbb{K}' \longrightarrow \mathbb{K}'$ que fixam os elementos de $\mathbb{K}$; a operação de grupo é a composição de morfismos.

Exemplo 47 :

Alguns factos, conceitos e notações relacionados com extensões $\mathbb{K}' \supset \mathbb{K}$

1. $\mathbb{K}$' e $\mathbb{K}$ têm a mesma característica.
2. $\mathbb{K}$' identifica-se com $\mathbb{K}$-espaço vectorial; a dimensão de $\mathbb{K}$' como $\mathbb{K}$-espaço vectorial é chamado **grau** da extensão e escreve-se $[\mathbb{K}' : \mathbb{K}]$. Se $[\mathbb{K}' : \mathbb{K}]$ é finito diz-se que $\mathbb{K}$' é uma **extensão finita** de $\mathbb{K}$. Uma **extensão quadrática** tem grau 2.
3. Se α é algébrico em $\mathbb{K}$ e d é o grau do menor polinómio irredutível, não nulo, de coeficientes em $\mathbb{K}$ que tém α como raíz, representa-se por $\mathbb{K}(\alpha)$ a extensão de grau d de $\mathbb{K}$ de **base** $(1, \alpha, \cdots, \alpha^{d-1})$; isto é, o elemento genérico de $\mathbb{K}(\alpha)$ tem a forma

$$x_0 + \alpha \cdot x_1 + \alpha^2 \cdot x_2 + \cdots + \alpha^{d-1} \cdot x_{d-1}$$

com $(x_0, x_1, \cdots, x_{d-1}) \in \mathbb{K}^d$. $\mathbb{K}(\alpha)$ é a menor extensão de $\mathbb{K}$ que contém α.

[43] Isomorfismos que preservam as operações do corpo.

4. Se $\alpha, \beta \in \mathbb{K}'$ são elementos algébricos de $\mathbb{K}$ então $\alpha + \beta$ e $\alpha \cdot \beta$ são também elemnetos algébricos. O sub-conjunto de $\mathbb{K}$' formado por todos os elementos algébricos de $\mathbb{K}$ forma um corpo $\bar{\mathbb{K}}_{\mathbb{K}'}$ designado por **fecho algébrico** de $\mathbb{K}$ em $\mathbb{K}'$. Obviamente que $\mathbb{K}'$ é uma extensão algébrica de $\mathbb{K}$ se e só se coincidir com o referido fecho.
5. Seja α um elemento de uma extensão $\mathbb{K}'$ (não necessariamente algébrica) de $\mathbb{K}$. Representa-se por $\mathbb{K}[\alpha]$ o menor sub-anel de $\mathbb{K}'$ que contém α e $\mathbb{K}$. O elemento genérico $z \in \mathbb{K}[\alpha]$ tem a forma $z = \sum_{k=0}^{m} a_k \cdot \alpha^k$, com $a_k \in \mathbb{K}$.
   Note-se que $\mathbb{K}[\alpha]$ é um anel mas não é necessariamente um corpo. Porém prova-se que, caso α seja um elemento algébrico (i.e. raíz de um polinómio de coeficientes em $\mathbb{K}$) então $\mathbb{K}[\alpha]$ é realmente um corpo e coincide com $\mathbb{K}(\alpha)$.
6. As notações anteriores, $\mathbb{K}(\alpha)$ e $\mathbb{K}[\alpha]$, sugerem outras notações.
   Seja $\mathbb{D}$ um qualquer **domínio integral**[44]. Representamos por $\mathbb{D}[X]$ o anel dos polinómios de coeficientes em $\mathbb{D}$ e variável anónima X. Representamos por $\mathbb{D}(X)$ o corpo das funções racionais de coeficientes em $\mathbb{D}$ e variável anónima X.
7. As notações anteriores podem ser generalizadas para mais do que uma variável, para mais do que um elemento e para conjuntos de elementos. Assim para qualquer domínio integral $\mathbb{D}$ e qualquer corpo $\mathbb{K}$
   (i) $\mathbb{D}[X, Y]$ representa o anel de polinómios a duas variáveis, X e Y, e de coeficientes em $\mathbb{D}$. Generalizando, $\mathbb{D}[X_1, \cdots, X_n]$ denota o anel dos polinómios a n variáveis.
   (ii) Se $\mathcal{A} \subset \mathbb{K}'$ é um sub-conjunto de uma extensão de $\mathbb{K}$, então $\mathbb{K}[\mathcal{A}]$ é o menor sub-anel de $\mathbb{K}'$ que contém $\mathcal{A}$ e $\mathbb{K}$.
   (iii) $\mathbb{D}(X_1, \cdots, X_n)$ denota o corpo das funções racionais a n variáveis e coeficientes em $\mathbb{D}$. O seu elemento genérico tem a forma P/Q com P e Q polinómios a n variáveis sobre $\mathbb{D}$.
   (iv) Se $\mathcal{A}$ é um sub-conjunto de uma extensão $\mathbb{K}'$ de $\mathbb{K}$, então

[44] Um domínio integral é um anel comutativo em que $1 \neq 0$ e sem divisores de zero: i.e. $a \cdot b = 0$ implica $a = 0$ ou $b = 0$. Os inteiros $\mathbb{Z}$ são o exemplo mais simples de domínio integral. Também é domínio integral o anel $\mathbb{Z}[X]/\mathbf{c}(X)$ dos polinómios de coeficientes inteiros módulo um polinómio irredutível $\mathbf{c}(X)$.

$\mathbb{K}(\mathcal{A})$ denota a menor extensão de $\mathbb{K}$ que contém $\mathcal{A}$.
Se for finito $\mathcal{A} = \{a_1, \cdots, a_n\}$, então, para cada $z \in \mathbb{K}(\mathcal{A})$ existe um par de polinómios $P, Q \in \mathbb{K}[X_1, \cdots, X_n]$ tais que $z = P(a_1, \cdots, a_n)/Q(a_1, \cdots, a_n)$.

(v) Nas condições anteriores, sendo algébricos todos os elementos de $\mathcal{A}$, tem-se $\mathbb{K}[\mathcal{A}] \equiv \mathbb{K}(\mathcal{A})$.

□

Exemplo 48 : Os corpos finitos $\mathbb{F}_q$ (página **??**) são os exemplos mais importantes de corpo.

Vimos que a característica p de $\mathbb{F}_q$ é sempre diferente de zero e é um primo. Vimos também que existe um $n > 0$, chamado a *dimensão do corpo*, tal que o número de elementos q é $q = p^n$ e, a menos de um isomorfismo, $\mathbb{F}_q$ se identifica com $(\mathbb{Z}_p)^n$.

Qualquer extensão finita de $\mathbb{F}_q$ tem também característica p e identifica-se com $(\mathbb{F}_q)^d$, sendo d o grau da extensão.

Exemplo 49 : Os corpo dos racionais $\mathbb{Q}$, dos reais $\mathbb{R}$ e dos complexos $\mathbb{C}$ têm característica 0.

O corpo $\mathbb{C}$ é uma extensão quadrática de $\mathbb{R}$ com a base $\{1, i\}$; isto é, cada elemento $z \in \mathbb{C}$ identifica-se com um par $(x, y) \in \mathbb{R}^2$ de tal forma que $z = x \cdot 1 + y \cdot i$.

O **fecho algébrico** $\bar{\mathbb{K}}$ é a menor extensão algébrica de $\mathbb{K}$ que contém todas as raízes dos polinómios de coeficientes em $\mathbb{K}$.

6.3.Variedades Algébricas

As variedades algébricas surgiram do estudo das curvas definidas num espaço a n dimensões por polinómios em n variáveis sobre um determinado corpo $\mathbb{K}$. Por exemplo, os dois polinómios a duas variáveis (c é um inteiro positivo) seguintes

$$\alpha(x,y) \doteq y^2 - x\,(x-c)^2 \qquad \gamma(x,y) \doteq y^2 - x^3$$

definem duas curvas no plano $\mathbb{R}^2$ como o lugar geométrico dos pares (x,y) para os quais $\alpha(x,y) = 0$ e $\gamma(x,y) = 0$.

Figura 4: Variedades $y^2 - x\,(x-c)^2$ ("curva α") e $y^2 - x^3$ ("curva γ").

Note-se que ambos os polinómios (definidos inicialmente sobre os inteiros) estão definido sobre o corpo dos racionais $\mathbb{Q}$ mas também sobre o corpo dos reais $\mathbb{R}$ e sobre os complexos $\mathbb{C}$ (i.e. pertencem a $\mathbb{Q}[x,y]$, a $\mathbb{R}[x,y]$ e a $\mathbb{C}[x,y]$). De facto ambos os polinómios estão definidos sobre qualquer extensão de $\mathbb{Q}$.

Genericamente $\alpha \in \mathbb{K}[x,y]$ sendo $\mathbb{K}$ uma qualquer extensão de um corpo que contenha c, enquanto que $\gamma \in \mathbb{K}[x,y]$ em que $\mathbb{K}$ é qualquer corpo.

O mesmo tipo de análise pode ser feita com polinómios muito semelhantes aos anteriores (uma vez mais c é um inteiro positivo)

$$\rho_0(x,y) \doteq y^2 - x\,(x^2 + c^2) \qquad \rho_1(x,y) \doteq y^2 - x\,(x^2 - c^2)$$

O lugar geométrico dos pontos no plano $\mathbb{R}^2$ para os quais $\rho_0(x,y)=0$ e $\rho_1(x,y)=0$ definem as curvas na figura **??**.

Figura 5: Variedades $y^2-x\,(x^2+c^2)$ e $y^2-x\,(x^2-c^2)$.

Apesar da semelhança nos polinómios que definem as quatro variedades (todos têm a forma genérica $y^2-x\,(x^2+\mu\,x+\upsilon)$, com $\mu,\upsilon\in\mathbb{K}$ para algum corpo $\mathbb{K}$) existem diferenças geométricas óbvias. Mais importante do que isso, essas diferenças geométricas traduzem-se em diferenças nas propriedades das estruturas algébricas que as variedades determinam.

Essencialmente tudo tem a haver com a definição da tangente e da curvatura a estas curvas.

Na figura **??** e na curva definida pela variedade $\alpha(x,y)\doteq y^2-x\,(x-c)^2$ vê-se que existe um ponto (nomeadamente $(0,c)$) que tem duas tangentes distintas. A curva definida por $\gamma(x,y)\doteq y^2-x^3$ tem um ponto (nomeadamante $(0,0)$) onde a tangente está bem definida (tem inclinação 0) mas que tem duas curvaturas distintas; no ponto $(0,0)$ a curva tem, simultanemante, curvatura "para cima" e "para baixo".

Na terminologia da Geometria Algébrica pontos de uma variedade onde a tangente está indefinida, chamam-se **nodos**; pontos onde a curvatura está indefinida chamam-se **cúspides**. Estes pontos e todos os pontos onde curvaturas de ordem superior estejam indefindas tomam o nome genérico de **pontos singulares**.

Já as curvas apresentadas na figura **??** não contém quaisquer pontos singulares: a tangente e a curvatura estão (aparentemente) únicamente definidas em todos os seus pontos.

Nota: É preciso, porém, ter em atenção que a forma geométrica que representámos é relevante apenas para as extensões de $\mathbb{Q}$. Com as variedades ρ_0 e ρ_1 definidas noutro corpo já a afirmação pode não ser válida. Por exemplo, num corpo de característica 2, os polinómios $y^2 - x\,(x^2 + c^2)$ e $y^2 - x\,(x^2 - c^2)$ coincidem e coincidem com o polinómio $y^2 + x\,(x + c)^2$; por isso definem todos a mesma variedade.

O problema central das variedades algébricas consiste em saber, fixando um corpo $\mathbb{K}$, quantos pontos de coordenadas em $\mathbb{K}$ existem sobre essa curva e que estruturas algébricas é possível definir sobre esses pontos. São essas estruturas algébricas que são particularmente interessantes nas aplicações criptográficas. Em todos os exemplos aqui estudados iremos, para já, supor que $\mathbb{K}$ é uma extensão de $\mathbb{Q}$; veremos adiante que esta restricção não limita excessivamente a nossa análise e que grande parte das conclusões se extendem naturalmente para um $\mathbb{K}$ arbitrário.

Se escolhermos, por exemplo, $\mathbb{K} \equiv \mathbb{Q}$, vê-se imediatamente que os pontos $(0, 0)$ e $(0, c)$ têm coordenadas inteiras e estão sobre a primeira curva. Na segunda curva os pontos de coordenadas inteiras são, pelo menos, $(0, 0)$ e $(0, \pm c)$. Estes são os pontos da variedade ditos *triviais*.

Exemplo 50 : Considere-se a variedade $\phi(x, y) = y^2 - x\,(x^2 - c^2)$.

A procura de pontos não-triviais (x, y) de coordenada racionais nesta variedade está ligada a um dos problemas clássicos da história da Matemática: procurar por "construções de régua e compasso" (isto é, por operações nos racionais) construir um triângulo cuja área fosse igual ao inteiro c.

Um racional positivo $c > 0$ igual à área de um triângulo rectângulo cujos lados sejam racionais chama-se **número congruente**. Nomeadamente um inteiro congruente é, obviamente, um número congruente que é inteiro. Por exemplo, 6 é um inteiro congruente porque existe um triângulo rectângulo de lados $\langle 3, 4, 5\rangle$ (note-se que $3^2 + 4^2 = 5^2$) cuja área é $(3 \cdot 4)/2 = 6$.

Outro exemplo, é o triângulo de lados $\langle 1\frac{1}{2}, 6\frac{2}{3}, 6\frac{5}{6}\rangle$ cuja área é $(1\frac{1}{2} \cdot 6\frac{2}{3})/2 = 5$. De facto 5 é o menor inteiro congruente.

Supúnhamos, então, que é possível construir um triângulo rectângulo de lados racionais $\langle r, s, t\rangle$ (com $r \neq s$) cuja área é c. Temos então

$$r^2 + s^2 = t^2 \qquad \text{e} \qquad (r \cdot s)/2 = c \qquad \text{com} \quad r, s, t, c \in \mathbb{Q} \tag{123}$$

Seja $u \doteq r + s$ e $v = r - s$; defina-se

$$x \doteq (t/2)^2 \qquad \text{e} \qquad y \doteq (t/2) \cdot (u/2) \cdot (v/2) \tag{124}$$

Calculando $\phi(x, y) = y^2 - x(x^2 - c^2)$, com (x, y) determinado por (**??**) e simplificando com as igualdades (**??**), verifica-se imediatamente que $\phi(x, y) = 0$.

Note-se que, sendo r, s, t racionais, também x, y são racionais. Portanto,

Se c for congruente, a variedade $y^2 - x\,(x^2 - c^2)$ tem, pelo menos, um ponto não-trivial de coordenadas racionais.

O inverso não é necessariamente verdadeiro; pode existir um ponto (x, y) de coordenadas racionais nesta variedade sem que tal provenha de um triângulo $\langle r, s, t\rangle$ de área c. Porém

Se $(x, y) \in \mathbb{Q}^2$ é um ponto da variedade $y^2 - x\,(x^2 - c^2)$ e $(x + c)$ e $(x - c)$ são quadrados de racionais, então c é congruente.

Para provar esta asserção basta considerar racionais $u, v \neq 0$ tais que $(x + c = u^2)$ e $(x - c = v^2)$; defina-se $t \doteq (2\,y)/(u\,v)$, $r \doteq u + v$ e $s \doteq u - v$. Facilmente se verifica que $\langle r, s, t\rangle$ é um triângulo rectângulo de lados racionais e de área c.

Formalmente, tome-se por referência um corpo K e o seu fecho algébrico $\bar{K}$. Tome-se uma extensão arbitrária L (com $\bar{K} \supseteq L \supseteq K$). Recordemos que G_L é o grupo dos automorfismos em $\bar{K}$ que fixam L.

Alguns conceitos de base em geometria algébrica

No que se segue fixamos uma dimensão $n > 0$.

DEFINIÇÃO 46 *O* **espaço afim** *de ordem n sobre K, representado por $\mathbb{A}^n$, é o conjunto dos n-tuplos*

$$\mathbb{A}^n = \{\, X \mid x_i \in \bar{K} \,\} \qquad \textit{com} \quad X = (x_1, \ldots, x_n)$$

Os elementos de $\mathbb{A}^n(L) \doteq \mathbb{A}^n \cap L^n$ *chamam-se* L*-racionais de dimensão* n*.*

Um **ideal** *é uma família* $I \subseteq \bar{K}[X]$ *de polinómios a* n *variáveis tal que, para todos* $\forall p, q \in I$ *e* $r \in \bar{K}[X]$*, verifica-se* $p\,r \in I$ *e* $p - q \in I$*.*

Para qualquer $n \geq 0$*,* I^n *é o menor sub-ideal de* I *que contém todos os polinómios* f^n *com* $f \in I$*.*

Um ideal $\mathfrak{p}$ *é* **primo** *(ou* **absolutamente irredutível***) se* $r\,s \in \mathfrak{p}$ *implica* $r \in \mathfrak{p}$ *ou* $s \in \mathfrak{p}$*.*

Um ideal I *é* **irredutível** *sobre* L *se* $I \cap L[X]$ *for primo.*

Um **conjunto algébrico** *é um conjunto da forma*

$$V \;=\; \{\, P \in \mathbb{A}^n \;\mid\; f(P) = 0 \quad \forall\, f \in I \,\} \tag{125}$$

para algum ideal I*. Se o ideal* I *for irredutível sobre* L*, então* V *é uma* **variedade algébrica** *sobre* L*. Uma variedade irredutível sobre* $\bar{K}$ *diz-se* **absolutamente irredutível***.*

Se V *é uma variedade gerada por um ideal* I*, denota-se por* $K(V)$ *o* **corpo das funções racionais** *de* V*. Os elementos de* $K(V)$ *são fracções de polinómios* $\frac{p}{q}$ *identificados pela relação de equivalência*

$$\frac{p}{q} \;=\; \frac{r}{s} \qquad \textit{sse} \quad p\,s - r\,q \in I$$

A **dimensão** *de* V *é o grau de transcendência de* $\bar{K}(V)$ *sobre* $\bar{K}$*; isto é, o tamanho do maior sub-conjunto de* $K(V)$ *que é algébricamente independente*[45]*. Uma* **curva algébrica** *sobre* L *é uma variedade sobre* L *de dimensão* 1*.*

Notas

1. Por acção natural sobre as coordenadas os morfismos em G_L mantêm invariantes os elements de $\mathbb{A}^n(L)$; de facto este espaço pode ser definido como o sub-espaço de $\mathbb{A}^n$ cujos elementos são invariantes sobre qualquer $\tau \in G_L$.

[45] Se L é uma extensão de K, um subconjunto $S \subseteq L$ é **algébricamente independente** se, dados quaisquer dois $a, b \in S$, não existe nenhum polinómio não-nulo $f \in \bar{K}[x, y] \neq 0$ tal que $f(a, b) = 0$.

2. A forma mais comum de representar ideais é através do seu conjunto de **geradores** $I = \langle g_1, g_2, \ldots, g_l \rangle$. Neste caso $f \in I$ se e só se pode ser escrito como

$$f \;=\; r_1\, g_1 + r_2\, g_2 + \cdots + r_l\, g_l \qquad \text{com } r_i \in \bar{K}[X]$$

3. O facto de um ideal ser irredutível depende da extensão considerada.
   Considere-se, por exemplo, os ideais gerados pelos dois polinómios sobre $\mathbb{Q}$:

$$I_1 \colon x_1^2 - 2x_2^2 \qquad , \qquad I_2 \colon x_1^2 + x_2^2 - 1$$

   Ambos polinómios são irredutíveis sobre $\mathbb{Q}[x_1, x_2]$. No entanto o primeiro pode-se escrever como

$$(x_1 - \sqrt{2}\, x_2)\,(x_1 + \sqrt{2}\, x_2)$$

   e, portanto, I_1 não é irredutível sobre a extensão $\mathbb{Q}(\sqrt{2})$.
   Já o polinómio $(x_1^2 + x_2^2 - 1)$ é irredutível em qualquer extensão de $\mathbb{Q}$. Por isso I_2 é absolutamente irredutível.

4. Note-se que $\bar{\mathbb{Q}} = \mathbb{C}$ e que $\mathbb{A}^2$, sobre $\mathbb{Q}$, coincide com $\mathbb{C}^2$. Neste contexto o conjunto algébrico V_1 determinado pelo ideal I_1 é dado por

$$V_1 \;=\; \{\, (x, y) \in \mathbb{C}^2 \;\mid\; x^2 - 2\, y^2 = 0 \,\}$$

   Em qualquer extensão de $\mathbb{Q}(\sqrt{2})$ este conjunto fraciona nas duas rectas $x = \pm\sqrt{2}\, y$ e, portanto, não define uma variedade.
   Já o conjunto algébrico determinado por I_2

$$V_2 \;=\; \{\, (x, y) \in \mathbb{C}^2 \;\mid\; x^2 + y^2 - 1 = 0 \,\}$$

   já define uma variedade sobre qualquer extensão de $\mathbb{Q}$ dado que I_2 é absolutamente irredutível. De facto, sobre $\mathbb{R}$ ou qualquer sua extensão, V_2 determina o círculo de raio unitário.

6.4.Variedades Projectivas

Para descrever variedades de uma forma genérica convém definir o espaço sobre o qual estão definidas.

Vamos supor, por momentos, que se quer definir exclusivamente variedades a n dimensões. Um "truque" muito útil consiste em acrescentar uma dimensão extra, construir $\mathbb{A}^{n+1}$, e depois definir uma relação de equivalência entre pares de pontos nesse espaço.

No que se segue K denota um corpo arbitário e L uma extensão arbitrária de K contida no seu fecho algébrico $\bar{K}$.

DEFINIÇÃO 47 *O* **espaço projectivo** $\mathbb{P}^n$ *é definido como o espaço quociente*

$$\mathbb{A}^{n+1} \setminus (0, \cdots, 0)/\cong$$

em que $\cong$ *é a relação de equivalência*

$(x_0, x_1, \cdots, x_n) \cong (x'_0, x'_1, \cdots, x'_n)$ *sse*

$$\exists \lambda \in \bar{K} : x_i = \lambda\, x'_i \; i \in \{0..n\}$$

Os n*-tuplos* $(x_0, \cdots, x_n) \in \bar{K}$ *chamam-se* **coordenadas homogéneas** *e as classes de equivalência chamam-se* **pontos projectivos**. *O ponto projectivo que contém* $(x_0, \cdots, x_n)$ *representa-se por* $[x_0, \cdots, x_n]$.

O triplo $(x_0, \cdots, x_i) \in \bar{K}$ *diz-se* **normalizado** *se algum dos* x_i *for igual a* 1.

O espaço $\mathbb{P}^n(L)$ *dos* L**-pontos racionais** *é o sub-espaço de* $\mathbb{P}^2$ *formado por todos os pontos projectivos* $[x_0, \cdots, x_n]$ *tais que, para algum* $\lambda \in \bar{K}$ *, se tem* $\lambda\, x_i \in L$ *para todo* i.

Notas

Por simplicidade de notação os seguintes comentários dirigem-se apenas a $\mathbb{P}^2$ mas aplicam-se igualmente a todo $\mathbb{P}^n$.

1. Essencialmente estamos a embeber o espaço a duas dimensões num espaço a três dimensões (que não contém a origem) e em que todos os pontos numa mesma recta que passa pela origem, são considerados equivalentes.
   Por exemplo, se K for o corpo $\mathbb{Q}$ dos racionais e L como o seu fecho algébrico $\mathbb{C}$, então tem-se
   $$(1, \alpha, \bar{\alpha}) \cong (\alpha, i, 1) \cong (\bar{\alpha}, 1, -i) \qquad \text{com} \quad \alpha \doteq \frac{1+i}{\sqrt{2}} \quad \bar{\alpha} \doteq \frac{1-i}{\sqrt{2}}$$
2. Se $[x, y, z] \in \mathbb{P}^2(L)$ isto não significa que todas as coordenadas sejam elementos de L. No entanto seleccionando uma qualquer coordenada diferente de zero (por exemplo, se for $x \neq 0$), então tanto $y\,x^{-1}$ como $z\,x^{-1}$ são elementos de L.
   Genericamente, se (x, y, z) estiver normalizado, qualquer das componentes pertence a L.
3. Seja $\tau \in G_L$ um qualquer automorfismo em $\bar{K}$ que fixa os elementos de L. Então τ estende-se para pontos projecivos actuando sobre as diferentes coordenas; isto é, atendendo que $\tau(\lambda x) = \tau(\lambda)\tau(x)$, para todo $x \in \bar{K}$, tem-se que
   $$\begin{aligned} (x, y, z) &\cong (x', y', z') \\ \Rightarrow \quad (\tau(x), \tau(y), \tau(z)) &\cong (\tau(x'), \tau(y'), \tau(z')) \end{aligned}$$
   Por isso faz sentido definir
   $$\tau[x, y, z] \doteq [\tau(x), \tau(y), \tau(z)] \tag{126}$$
   Nessas circunstâncias é fácil verificar que os L-racionais são precisamente os pontos de $\mathbb{P}^2$ que são fixados por qualquer $\tau \in G_L$.

O espaço projectivo $\mathbb{P}^2$ será a base de estudo para a maioria dos conceitos que iremos apresentar. No entanto a maioria desses conceitos generalizam-se para uma dimensão n arbitrária e por isso, apesar de tal introduzie um grau de complexidade adicional, vale a pena apresentá-los no contexto de um $\mathbb{P}^n$ arbitrário.

Existem no entanto algumas noções que são particularmente simples na versão $\mathbb{P}^2$ e convém apresentá-las dessa forma.

DEFINIÇÃO 48 *Sejam* $P = [a, b, c]$ *e* $Q = [a', b', c']$ *dois pontos de* $\mathbb{P}^2$. *Se* $P \neq Q$ *define-se*

$$P \otimes Q \doteq [bc' - cb' \ , \ ca' - ac' \ , \ ab' - ba'] \qquad (127)$$

Representa-se por $\mathfrak{l}(P)$, *e designa-se por* **recta** *de* P, *o ideal gerado pelo polinómio* $u \in \bar{K}[x, y, z] = ax + by + cz$.

Facilmente se verifica que $P \times Q$ e $\mathfrak{l}(P)$ são independentes do representante escolhido para os pontos projectivos P e Q.

Como $P\otimes$ Q só está definido para pontos distintos, pode-se estender a definição acrescentando um ponto 0 extra ao espaço $\mathbb{P}^2$ e fazendo $P \otimes P = P \otimes 0 = 0 \otimes 0 = 0$. Também $\mathfrak{l}(0) \doteq \mathbb{P}^2$. Nestas circunstâncias

PROPOSIÇÃO 5 *(i) O operador* $\otimes : \mathbb{P}^2 \cup \{0\} \times \mathbb{P}^2 \cup \{0\} \to \mathbb{P}^2 \cup \{0\}$ *é comutativo e verifica* $P \otimes Q = 0$ *se e só se* $P = Q \vee P = 0 \vee Q = 0$.

(ii) Para todo $P, Q, R \in \mathbb{P}^2$ *verifica-se*

$$R \in \mathfrak{P} \otimes \mathfrak{Q} \quad sse \quad P \in \mathfrak{R} \otimes \mathfrak{Q} \quad sse \quad Q \in \mathfrak{P} \otimes \mathfrak{R}$$

□

Regressando à versão genérica $\mathbb{P}^n$, no que se segue X denota o vector de variáveis $(x_0, x_1, \cdots, x_n)$ e λX denota o vector $(\lambda x_0, \lambda x_1, \cdots, \lambda x_n)$.

DEFINIÇÃO 49 *Um polinómio* $f \in K[X]$ *é* **homogéneo** *de grau* d *se, para todo* $\lambda \neq 0 \in \bar{K}$, $f(\lambda X) = \lambda^d f(X)$.

Os polinómios homogéneos em $K[X]$ formam um anel que designaremos por $K_h[X]$.

De forma abreviada, se tivermos $P \in \mathbb{P}^n$ e $f \in K_h[X]$, escreve-se $f(P)$ para representar $\{ f(x_0, \cdots, x_n) \mid [x_0, x_1, \cdots, x_n] = P \}$. Note-se que $f(P)$ contém 0 só se for exactamente o conjunto singular $\{0\}$. De forma abreviada, designaremos essa situação por $f(P) = 0$. Se $f(P) \neq 0$, então dois pontos $a, b \in f(P)$ estão relacionados por $a = \lambda^d b$ para algum $\lambda \in \bar{K}$.

Exemplo 51 : Tomemos por corpo de base os racionais $\mathbb{Q}$ e consideramos duas das suas extensões: $L \doteq \mathbb{Q}(\alpha)$ e $L' \doteq \mathbb{Q}(\beta)$, em que α e $\beta \neq 1$ são, respectivamente, raízes dos polinómios $X^2 - 2$ e $X^3 - 1$.

Os polinómios $\pi_1, \pi_2, \pi_3 \in \mathbb{Q}[x, y, z]$

$$\pi_1 = x^2 - 2y^2 \quad , \quad \pi_2 = x^2 + x\,z + z^2$$

$$\pi_3 = x^4 + x^3 z + x^2 z^2 - 2\,y^2 x^2 - 2\,y^2 xz - 2\,y^2 z^2$$

são exemplos de polinómios homogéneos de graus 2, 2 e 4 respectivamente.

Os polinómios π_1 e π_2 são irredutíveis em $\mathbb{Q}[x, y, z]$; o polinómio π_3 facturiza em $\pi_1 \cdot \pi_2$ e, portanto, é redutível.

Note-se que, em $L[x, y, z]$ se tem

$$\pi_1 = (x - \alpha\, y)\,(x + \alpha\, y)$$

e em $L'[x, y, z]$ se tem

$$\pi_2 = (x - \beta z)\,(x - (1 - \beta) z)$$

Portanto os dois polinómios π_1 e π_2 já não são irredutíveis se se escolher a extensão apropriada de $\mathbb{Q}$.

DEFINIÇÃO 50 *O espaço das* **fracções homogéneas** $K_h(X)$ *é definido pelo quociente* $K_h[X] \times K_h[X] \setminus 0/\sim$ *em que*

$$(p, q) \sim (p', q') \quad sse \quad p\,q' - p'\,q = 0$$

Para $f \in, K_h(X)$ *o* **grau** *de* f *é dado por* $\deg(p) - \deg(q)$ *para um qualquer representante* $p/q \in f$.

Notas

Fracções homogéneas são classes de equivalências de pares de polinómios homogéneos (p, q), em que $q \neq 0$ e que considera equivalentes pares (p, q) e (p', q') tais que $p\,q' = q\,p'$.

Isto significa que $\deg(p) + \deg(q') = \deg(q) + \deg(p')$; portanto

$$\deg(p) - \deg(q) \;=\; \deg(p') - \deg(q')$$

Portanto a diferença entre o grau do numerador p e o grau do denominador q é independente do representante da classe. Por isso a definição de grau de F como $\deg(p) - \deg(q)$ faz sentido porque este valor é independente do representante escolhido.

A fracção $f \in K_h(X)$ diz-se **regular** no ponto $P \in \mathbb{P}^n$ se existe um representante $p/q \in f$ tal que $q(P) \neq 0$.

FACTO 30 *Se f é uma fracção homogénea regular em P e de grau 0, então existe um único valor, designado por $f(P)$, tal que*

$$f(P) \;=\; p(x_0, \cdots, x_n)/q(x_0, x_1, \cdots, x_n$$

para qualquer representante $(p/q) \in f$ e qualquer representante $(x_0, \cdots, x_n)$ de P.

Prova

Seja p/q um representante de f tal que $q(P) \neq 0$ e seja $X = (x_0, \cdots, x_n)$ um qualquer representante de P.

Como f tem grau zero, os polinómios p e q têm de ter o mesmo grau d. Então

$$\frac{p(\lambda X)}{q(\lambda X)} = \frac{\lambda^d p(X)}{\lambda^d q(X)} = \frac{p(X)}{q(X)}$$

Portanto $(p/q)(X)$ é independente do representante específico de P e faz sentido definir $(p/q)(P)$ como este valor.

Tomemos agora um segundo representante p'/q' de f. Pela equivalência $(p/q) \sim (p'/q')$ temos

$$(p/q)(X) - (p'/q')(X) \;=\; \frac{p(X)\,q'(X) - q(X)\,p'(X)}{q(X)\,q'(X)} \;=\; 0$$

Portanto o valor de $(p/q)(P)$ é independente do representante de f e, deste modo, determina $f(P)$.

FACTO 31 *$K_h(X)$ tem a estrutura de um corpo. O espaço $K_h^0(X) \subset K_h(X)$ das fracções homogéneas de grau 0 é um sub-corpo de $K_h(X)$.*

A prova é simples com as definições

$0 = (0,1) \quad 1 = (1,1) \quad (p,q)+(r,s) = (p\,s+q\,r, q\,s) \quad (p,q)+(r,s) = (p\,r, q\,s)$

Para fracções de grau 0 é óbvio que somas e multiplicações preservam essa ordem.

DEFINIÇÃO 51 *Um* **ideal homogéneo** *$I \subseteq \bar{K}[X]$, em $\mathbb{P}^n$, é uma família de polinómios da forma $f = \sum_i f_i$ em que cada f_i é múltiplo de um polinómio homogéneo.*

Para cada extensão $L \supseteq K$, representa-se por $I(L)$ o ideal $I \cap L[X]$. I é **irredutível** *sobre L se $I(L)$ é primo.*

I é **absolutamente irredutível** *se é irredutível sobre qualquer extensão de K (equivalentemente, se for primo).*

$V \subseteq \mathbb{P}^n(L)$ é uma **conjunto algébrico projectivo** *sobre L se e só existe um ideal homogéneo I que verifica*

$$V \;=\; \{\, P \in \mathbb{P}^n(L) \;\mid\; f(P) = 0 \text{ para todo } f \in I \,\} \qquad (128)$$

Se o ideal I for irredutível sobre L então V é uma **variedade projectiva** *sobre L. Se, adicionalmente, I for absolutamente irredutível então a variedade V diz-se* **absolutamente irredutível**.

Notas

1. Note-se que se um ideal homogéneo I tiver um único polinómio gerador f_I que seja factorizável sobre $L[X]$, então não será irredutível sobre L; de facto tem-se $f_I = p\,q$ sem que p ou q pertençam ao ideal I uma vez que, para tal, teriam de ser múltiplos de f_I.
2. As variedades "interessantes" são as absolutamente irredutíveis; isto é, determinadas por um ideal primo. Como veremos em seguida estas variedades estão fortemente relacionadas como o seu espaço de funções racionais.

Exemplo 52 : Retomamos os polinómios π_1, π_2 e π_3 do exemplo **??** definidos no espaço projectivo $\mathbb{P}^2$ e considere-se, para $i = 1, 2, 3$, os ideais

$$I_i \;=\; \{\, f \in \mathbb{C}[x,y,z] \;\mid\; \pi_i \text{ divide } f \,\}$$

Ambos os ideais I_1 e I_2 são irredutíveis sobre $\mathbb{Q}$ uma vez que os respectivos polinómios geradores não são factorizáveis em $\mathbb{Q}[X]$. Já I_3, porque o polinómio gerador é factorizável, já é redutível; de facto tem-se

$$I_3 \;=\; I_1 \cap I_2$$

Se passarmos para a extensão $\mathbb{Q}(\sqrt{2})$ já o polinómio $(x^2 - 2y^2)$, gerador de I_1, é factorizável em $(x - \sqrt{2}\,y)\,(x + \sqrt{2}\,y)$. Portanto I_1 não é irredutível sobre $\mathbb{Q}(\sqrt{2})$.

Do mesmo modo I_3 não é irredutível sobre $\mathbb{Q}(\beta)$ (com $\beta = \sqrt[3]{1} \neq 1$) já que o polinómio gerador $(x^2 + x\,z + z^2)$ factoriza em $(x - \beta\,z)\,(x - z + \beta\,z)$.

Exemplo 53 : O espaço $\mathbb{P}^1$, designado por **recta projectiva**, tem uma importância particular.

As coordenadas homogéneas de $\mathbb{P}^1$ são pares $(x,y) \in \bar{K}^2$ que não podem ser simultâneamente nulos. Por isso as coordenadas projectivas em $\mathbb{P}^1$ são apenas

$$\mathbb{P}^1 \;=\; \{[1,0]\} \,\cup\, \{\, [\lambda, 1] \;\mid\; \lambda \in \bar{K} \,\} \tag{129}$$

O ponto especial $[1,0]$ é o único **ponto do infinito** de $\mathbb{P}^1$.

As variedades em $\mathbb{P}^1$ são determinadas pelos ideais homogéneos irredutíveis. Por exemplo o polinómio (x^2-2y^2) é irredutível em $\mathbb{Q}[x,y]$ mas já não é irredutível em $\mathbb{Q}(\sqrt{2})[x,y]$.

O polinómio $f(x,y)=0$ é irredutível e determina uma variedade que coincide com todo $\mathbb{P}^1$. Um polinómio de 1° grau $f(x,y)=\mu x+\lambda y$, com $\mu,\lambda\in L\supseteq K$ não simultaneamente nulos, determina uma variedade formada por um só ponto de $\mathbb{P}^1$: o ponto no infinito $[1,0]$ caso seja $\mu=0$ ou o ponto $[-\mu^{-1}\lambda,1]$ caso seja $\mu\neq 0$. Qualquer polinómio homogéneo f de grau superior a 1 não pode ser simultaneamente irredutível em $L\supseteq K$ e verificar $f(x,y)=0$.

Por isso $\mathbb{P}^1$ tem uma variedade de dimensão 1, que coincide com o próprio $\mathbb{P}^1$, e uma variedade de dimensão 0 determinada por cada um dos seus pontos projectivos.

DEFINIÇÃO 52 *Se* $\mathfrak{p}\subset \bar{K}[X]$ *é um ideal homogéneo primo, então* $K(\mathfrak{p})$, *designa o espaço quociente* $K_h^0(X)/\sim_{\mathfrak{p}}$ *com*

$$\frac{p}{q}\sim_{\mathfrak{p}}\frac{p'}{q'} \quad sse \quad p\,q'-p'\,q\in\mathfrak{p} \tag{130}$$

Se V *for a variedade gerada por* $\mathfrak{p}$ *então* $K(\mathfrak{p})$, *representa-se, também, por* $K(V)$. *O espaço* $K(\mathfrak{p})=K(V)$, *é o* **corpo de funções racionais** *determinado por* $\mathfrak{p}$ *(ou* V*).*

Igualmente se define $\bar{K}(V)$ *a partir do espaço de fracções homogéneas sobre* $\bar{K}$. *A* **dimensão** *da variedade* V *é o grau de transcendência de* $\bar{K}(V)$ *sobre* $\bar{K}$.

Notas

Note-se que as funções racionais são, em primeiro lugar, fracções homogéneas de grau 0; isto é, o grau do numerador é igual ao do denominador.

Depois consideram-se equivalentes duas fracções f,f' quando a diferença $f-f'$ tem um numerador que pertence ao ideal $\mathfrak{p}$.

Na sequência imediata do último comentário temos

FACTO 32 *Se V é uma variedade projectiva absolutamente irredutível então, para toda a função racional $f \in K(V)$ e todo o ponto $P \in V$ onde f é regular, existe um valor designado por $f(P)$ que verifica $f(P) = \tilde{f}(P)$ para todo o representante $\tilde{f} \in f$ regular em P.*

Prova

Se g, h são fracções homogéneas na classe f e são ambas regulares em P, então $g(P)$ e $h(P)$ são bem definidos (facto **??**) e, como $g - h$ tem um numerador que pertence ao ideal da variedade V, tem-se $g(P) - h(P) = 0$. Logo o valor $g(P) = h(P)$ é independente do representante de f e, deste modo, determina $f(P)$.

Exemplo 54 : Uma vez que a variedade $\mathbb{P}^1$ é determinada pelo ideal $\{0\}$, qualquer fracção homogénea representa uma única função racional nessa variedade. Por exemplo, considerando $K = \mathbb{Q}$

$$\frac{x^2 - 2y^2}{x^2 + y^2} \quad , \quad \frac{x^2 + y^2 - 2xy}{x^2 - y^2}$$

são representantes de fracções homogéneas (e, portanto, funções racionais) porque são pares de polinómios homogéneos em $\mathbb{Q}[x, y]$.

Note-se o cuidado com a palavra "representante" já que, por exemplo, $\frac{x^2+y^2-2xy}{x^2-y^2}$ e $\frac{x-y}{x+y}$ representam a mesma fracção.

O numerador e o denominados, como qualquer polinómio homogéneo em $\mathbb{C}[x, y]$, factorizam num número finito de polinómios do 1^o grau $(\alpha x - \beta y)$, com $\alpha, \beta \in \mathbb{C}$ não simultaneamente nulos. Por exemplo, nos complexos $\mathbb{C}$, as fracções anteriores são factorizadas da seguinte forma

$$\frac{(x - \sqrt{2}y)(x + \sqrt{2}y}{(x + iy)(x - iy)} \quad , \quad \frac{(x - y)^2}{(x - y)(x + y)}$$

Genericamente qualquer p/q representante de uma fracção homogénea em $\mathbb{P}^1$ pode-se escrever

$$p/q \;=\; \kappa y^d \frac{\prod_i (x - \alpha_i y)^{n_i}}{\prod_j (x - \beta_j y)^{m_j}} \tag{131}$$

com $\kappa, \alpha_i, \beta_j \in \mathbb{C}$. Dado que o numerador e o denominados têm o mesmo grau, tem de se verificar

$$d + \sum_i n_i - \sum_j m_j \;=\; 0 \tag{132}$$

Nesta circunstâncias diz-se que p/q tem um **zero** no ponto $(\alpha_i, 1)$ de ordem n_i, e um **pólo** no ponto $(\beta_j, 1)$ de ordem m_j. Se for $d > 0$, diz-se que tem um zero no infinito de ordem d; se for $d < 0$, terá um pólo no infinito de ordem $-d$.

DEFINIÇÃO 53 *Uma* **curva** *em* $\mathbb{P}^n(L)$ *é uma variedade projectiva* V/L *irredutível sobre* L *e de dimensão* 1.

Notas

1. A questão essencial na definição da curva é a dimensão. Recorde-se que a dimensão da variedade V/K é o grau de transcendência de $K(V)$ sobre $\bar{K}$; isto é, o tamanho do maior sub-conjunto de $K(V)$ cujos elementos são algebricamente independentes.
   Ao afirmar-se que a dimensão é 1 está-se a afirmar que, para quaisquer duas funções racionais $f, g \in \bar{K}(V)$ já não são algebricamente independentes; portanto existe um polinómio a duas variáveis $\phi \in \bar{K}[x, y]$, não-nulo, tal que $\phi(f(X), g(X)) = 0$.
   Note-se que, apesar de f e g serem geradas a partir de polinómios homogéneos, o polinómio ϕ não é necessariamente homogéneo. Seja D_a o conjunto algébrico afim que ϕ determina em $\mathbb{A}^2$; isto é $D_a = \{\, (x, y) \in \mathbb{A}^2 \;\mid\; \phi(x, y) = 0 \,\}$.
   Este conjunto algébrico não é necessariamente uma variedade (porque ϕ pode ser redutível) mas é sempre uma união finita de variedades. No entanto D_a depende exclusivamente das duas funções racionais escolhidas f e g; por isso será eventualmente possível escolher f e g de forma a que D_a não só seja uma variedade afim mas seja precisamemente uma variedade específica; por exemplo, uma curva afim em $\mathbb{A}^2$.
   Porque $\phi(f(X), g(X)) = 0$, para todo o ponto $P \in C$ onde f e g são ambos regulares, o ponto $(f(P), g(P))$ é um elemento de D_a. Portanto faz sentido defininir a seguinte aplicação de pontos de C em pontos de $D_a \cup \infty$.

$$(f, g) \colon P \mapsto \begin{cases} (f(P), g(P)) & \text{se } f \text{ e } g \text{ são regulares em } P \\ \infty & \text{se } f \text{ ou } g \text{ não é regular em } P \end{cases} \tag{133}$$

   Se f e g forem regulares em todos os pontos de C e forem escolhidos de forma a D_a ser uma curva afim, então este processo ilustra como qualquer curva, num espaço genérico $\mathbb{P}^n$, é mapeada numa curva afim em $\mathbb{A}^2$.

2. Uma outra forma de mapeamento é possível.
   Considere-se uma qualquer função racional $f \in \bar{K}(C)$ e defina-se

$$\hat{f}\colon P \mapsto \begin{cases} [f(P),1] & \text{se } f \text{ é regular em } P \\ [1,0] & \text{em caso contrário} \end{cases} \tag{134}$$

   Então $\hat{f}\colon C \to \mathbb{P}^1$ é um morfismo de variedades projectivas (ver a definição de morfismo a seguir).
   Isto significa que o operador $\hat{\cdot}\colon K(C) \to \mathrm{Homo}(C,\mathbb{P}^1)$ mapeia funções racionais em morfismos de C em $\mathbb{P}^1$.
   Este operador tem um inverso óbvio; se tomarmos um qualquer morfismo $\phi\colon C \to \mathbb{P}^1$ sabemos que ele é determinado por um para de funções racionais $\phi = [p,q]$ com a propriedade de, para todo $P \in C$, não ser simultaneamente $p(P) = 0$ e $q(P) = 0$. Este par determina uma função racional $f = q^{-1}p$ que verifica $\hat{f} = \phi$.

Estas notas justificam o seguinte resultado:

LEMA 3 *Seja C uma curva projectiva absolutamente irredutível. Então*

(i) Cada par de funções racionais $f, g \in K(C)$ determina um conjunto algébrico afim $D_{f,g}$ em $\mathbb{A}^2$ e um morfismos $(f,g)\colon C \to D_{f,g}$.

(ii) Existe um isomorfismo canónico $\hat{\cdot}\colon K(C) \to \mathrm{Homo}(C,\mathbb{P}^1)$ entre as funções racionais em C e os morfismos de C para $\mathbb{P}^1$.

Exemplo 55 : Uma curva em $\mathbb{P}^2$ é determinada por um único polinómio f; é costume representar esse facto por $C\colon f$; seja I_f o respectivo ideal.

Como caso particular de curvas temos as "rectas" em $\mathbb{P}^2$

Seja $P \in \mathbb{P}^2(L)$ um qualquer L-racional. Para cada representante $(a,b,c) \in P$, ideal I_P gerado pelo polinómio $(a\,x + b\,y + c\,z)$ é, claramente, irredutível e independente do representante escolhido. A variedade $V_P\colon L$ gerada por I_P é a **linha** (ou **recta**) determinada por P.

□

Construamos agora os 3 polinómios que resultam de calcular as derivadas parciais de f em relação a cada uma das 3 variáveis, x, y e z:

Constrói-se desta forma um vector de polinómios

$$\langle\, \partial f/\partial x \;,\; \partial f/\partial y \;,\; \partial f/\partial z \,\rangle$$

que são todos homogéneos e, portanto, determinam um ideal homogéneo; esse ideal é representado por $\partial I/\partial X$.

Note-se que $\partial I/\partial X$ não é, normalmente, irredutível e, por isso, não determina uma variedade. No entanto determina um conjunto algébrico projectivo que representamos por ∂C .

Os **pontos singulares** de C são os elementos de $C \cap \partial C$. Se este conjunto não é vazio (a curva C contém pelo menos um ponto singular) então C diz-se **singular**.

Considere-se a curvas C_1 determinada pelo polinómio $z\,y^2 - x^2\,(x+z)$; isto é,

$$C_1 : z\,y^2 - x^2\,(x+z)$$

O conjunto algébrico ∂C_1 é gerado por

$$\langle\, x\,(3x+2z) \;,\; y\,z \;,\; x^2 - y^2 \,\rangle$$

O ponto $[0,0,1]$ pertence a este conjunto algébrico e também pertence a C_1. Por isso é um ponto singular de C_1.

Considere-se agora a curva

$$C_2 : zy^2 - x\,(x^2 - z^2)$$

Neste caso o conjunto algébrico ∂C_2 é gerado por

$$\langle\, 3\,x^2 - z^2 \;,\; y\,z \;,\; y^2 - 2xz \,\rangle$$

e não contém qualquer elemento; portanto C_2 não tem pontos singulares.

Como vimos, cada ponto projectivo $[x,y,z] \in \mathbb{P}^2$ pode ser visto como o lugar geométrico da recta em $\mathbb{A}^3$ que passa pela origem e pelo ponto (x,y,z) .