J.M.E. Valença

Técnicas Criptográficas

Notas Teóricas

(http://cripto.di.uminho.pt/)

Docentes:

J. M. E. Valença	jmv@di.uminho.pt
M. A. Cunha	alcino@di.uminho.pt

Técnicas Criptográficas– Módulo I

Terminologia

1. Objectivo do curso.
2. Cifras.
    1. Noção de cifra
    2. Cifras simétricas e assimétricas.
3. Assinaturas digitais.
4. Funções de *Hash*
5. Identificação de Agentes.
6. Certificação de Chaves Públicas.
7. Geração e distribuição de chaves.

Terminologia

Objectivo do curso

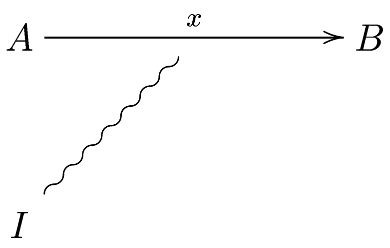
Garantia de **comunicação segura** entre **agentes** em ambiente hostil.

A $\equiv$ *emissor*.

B $\equiv$ *receptor*, *verificador*, ...

x $\equiv$ *mensagem*, *texto*, ...

I $\equiv$ *intruso*, *atacante*, *espião*, ...



Características da Segurança

Confidencialidade Garante o *controlo do conhecimento* da mensagem: após o seu envio, apenas A e B (e nenhum intruso) conhecem x. B deve ser o único receptor da mensagem.

Autenticidade Garante a *integridade* da mensagem: a mensagem que chega a B é a mesma que foi emitida por A. Garante a *originalidade* da mensagem: B tem certeza da origem A da mensagem x. Garante a A a *refutação* de mensagens fraudulentas e a B a *não-refutação* de mensagens legítimas.

Entidade Garante a *identificação* de um agente perante outro agente. B e A devem ter garantias que comunicam um com o outro e nenhum deles pode ser substituído por I.

Requisitos de Segurança

Em oposição às técnicas tradicionais de segurança em *canais fechados*, este curso estuda as técnicas de segurança em *canais abertos*.

A **segurança em canais fechados**, ou *segurança física*, baseia-se no *obscurantismo nas comunicações e nas técnicas*: o conhecimento sobre o tráfego de informação e sobre as técnicas de segurança é controlado.

$\ominus$ *Os sistemas de segurança, uma vez instalados, não são testados e se forem atacados não é fácil constatar a existência e a natureza do ataque.*

$\Uparrow$ **muito grave** $\Uparrow$

$\oplus$ *Não impõe requisitos especiais aos sistemas computacionais*

A **segurança em canais abertos** assume que todo o tráfego de informação é público e que as técnicas de segurança são publicamente conhecidas. O controlo do segredo restringe-se a certos items de informação: as **chaves**.

$\oplus$ *Os sistemas de segurança podem ser testados pela comunidade científica em geral e eventuais quebras de segurança têm mais possibilidade de ser detectadas.*

$\ominus$ *Porque o conhecimentos dos intrusos é maior, as técnicas de segurança devem ser mais exigentes e computacionalmente mais elaboradas dos que as usadas em canais fechados.*

$\Uparrow$ **pouco grave** $\Uparrow$

Cifras

Para garantir confidencialidade na comunicação usam-se **cifras**: funções invertíveis transformando textos em **criptogramas**.

$$A \xrightarrow{\{m\}} B$$

canal aberto

$$A \overset{}{\underset{m}{\cdots\cdots\rightarrow}} B$$

canal seguro equivalente

$\mathcal{M}$	*espaço de textos*
$m \in \mathcal{M}$	*texto claro, texto*
$\{m\}$	*criptograma de m*

Condição de Confidencialidade

> Apenas o receptor B deve ser capaz de recuperar o texto $\mathbf{m}$ a partir do criptograma $\mathbf{\{m\}}$.

Códigos

A comunicação exige uma fase prévia de **codificação** através de uma função invertível que converte textos em números inteiros (**códigos**).

$X \equiv$	*espaço de códigos*
$\mathsf{Cod} : \mathcal{M} \longrightarrow X$	*função de codificação*
$\mathsf{Cod}^{-1} : X \longrightarrow \mathcal{M}$	*função de descodificação*

As funções de codificação e descodificação constituem uma mudança de representação da informação e são publicamente conhecidas. *Sob o ponto de vista da segurança são irrelevantes.* Por isso:

> A menos que seja explicitamente dito o contrário, no que se segue não vamos distinguir mensagens de códigos.

Noção Geral de CIFRA

$X \equiv$ *códigos do texto* $\qquad Y \equiv$ *códigos do criptograma*

$\mathbf{c} : X \longrightarrow Y$ *função de cifragem*

$\mathbf{d} : Y \longrightarrow X$ *função de decifragem*

CORRECÇÃO

$$\mathbf{d}(\mathbf{c}(x)) = x \qquad \forall x \in X$$

NÃO INVERSÃO

Dado qualquer $y \in Y$ é computacionalmente intratável, sem conhecimento de $\mathbf{d}$, encontrar x tal que $\mathbf{c}(x) = y$.

CONFIDENCIALIDADE

Apenas B conhece $\mathbf{d}(\cdot)$

$$\boxed{\{x\} \doteq \mathbf{c}(x)}$$

Para controlo do conhecimento é conveniente que a preservação do segredo se faça através de chaves. Assim essas funções transformam-se em *famílias de funções* indexadas por um parâmetro: a *chave*.

Cifras Simétricas

a mesma chave é usada para cifrar e para decifrar

$$\mathbf{c}_k : X \longrightarrow Y \qquad k \in \mathcal{K} \quad \textit{funções de cifragem}$$

$$\mathbf{d}_k : Y \longrightarrow X \qquad k \in \mathcal{K} \quad \textit{funções de decifragem}$$

CORRECÇÃO

(i) $\mathbf{d}_k(\mathbf{c}_k(x)) = x$ para todo $x \in X$

(ii) $k \neq k' \implies \mathbf{d}_k \neq \mathbf{d}_{k'}$

NÃO INVERSÃO

(i) (*não-inversão da chave*) Conhecidos $x \in X$ e $y \in Y$, é computacionalmente intratável encontrar $k \in \mathcal{K}$ tal que $y = \mathbf{c}_k(x)$.

(ii) (*não-inversão do criptograma*) Conhecido $y \in Y$ é computacionalmente intratável determinar $x \in X$ e $k \in \mathcal{K}$ tal que $y = \mathbf{c}_k(x)$

CONFIDENCIALIDADE

(i) Apenas A e B conhecem k

$$\boxed{\{x\}_k \doteq \mathbf{c}_k(x)}$$

Genericamente, ...

$\{x\}_k$ é o criptograma de x cuja decifragem requer o conhecimento da chave k.

Cifras Assimétricas

A cifragem e a decifragem usam chaves distintas

$$\mathbf{c}_k : X \longrightarrow Y \qquad k \in \mathcal{K} \qquad \textit{cifragem}$$
$$\mathbf{d}_u : Y \longrightarrow X \qquad u \in \mathcal{U} \qquad \textit{decifragem}$$

CORRECÇÃO

(i) Para cada $k \in \mathcal{K}$ existe **um único** $u \in \mathcal{U}$ tal que

$$\mathbf{d}_u(\mathbf{c}_k(x)) = x \quad \text{para todo } x \in X$$

u é uma **chave privada** e k é a **chave pública** correspondente.

NÃO INVERSÃO

(i) (*não-inversão da chave pública*)
Conhecidos $x \in X$ e $k \in \mathcal{K}$ é computacionalmente intratável determinar $u \in \mathcal{U}$ tal que $\mathbf{d}_u(\mathbf{c}_k(x)) = x$.

(ii) (*não-inversão do criptograma*)
Conhecidos $y \in Y$ e $k \in \mathcal{K}$ é computacionalmente intratável determinar $x \in X$ tal que $y = \mathbf{c}_k(x)$

CONFIDENCIALIDADE

(i) Apenas B conhece u.

$$\boxed{\{x\}_u \doteq \mathbf{c}_k(x)}$$

Cifras Assimétricas Não Deterministas

Cada texto x pode ser cifrado aleatóreamente numa infinidade de criptogramas y.

$$\mathbf{c}_k : X \longrightarrow 2^Y \qquad k \in \mathcal{K} \qquad \textit{cifragem}$$
$$\mathbf{d}_u : Y \longrightarrow X \qquad u \in \mathcal{U} \qquad \textit{decifragem}$$

Correcção

(i) Para cada $k \in \mathcal{K}$ existe **um único** $u \in \mathcal{U}$ tal que $\mathbf{d}_u(y) = x$ para todo $x \in X$ e $y \in \mathbf{c}_k(x)$.

Não Inversão

(i) (*não-inversão da chave pública*)
Conhecidos o texto $x \in X$, a chave $k \in \mathcal{K}$ e qualquer criptograma $y \in \mathbf{c}_k(x)$, é computacionalmente intratável determinar $u \in \mathcal{U}$ tal que $\mathbf{d}_u(y) = x$.

(ii) (*não-inversão do criptograma*)
Conhecidos $y \in Y$ e $k \in \mathcal{K}$ é computacionalmente intratável determinar $x \in X$ tal que $y \in \mathbf{c}_k(x)$

Confidencialidade

(i) Apenas B conhece u.

$\{x\}_u$ é um elemento aleatório de $\mathbf{c}_k(x)$

Cifras Simétricas vs. Cifras Assimétricas

Assimétricas

- $\oplus$ Só a chave privada precisa de ser mantida confidencial.
- $\oplus$ A gestão das chaves públicas de uma comunidade de N agentes pode ser feita numa base de dados aberta requerendo apenas a *certificação* das chaves por um *agente de confiança* ("trusted agent"). Cada agente individualmente só se tem de preocupar com a gestão de uma única chave: a sua chave privada.
- $\ominus$ A complexidade computacional é várias ordens de grandeza superior à das cifras simétricas. O tamanho das chaves, a complexidade dos algoritmos, a complexidade de criação das chaves são sempre superiores aos parâmetros equivalentes nas cifras simétricas.
- $\ominus$ Não é fácil alterar um par de chave pública e privada; estas chaves tendem a ser **chaves de longa duração**.

Cifras Simétricas

- $\ominus$ Numa comunicação entre dois agentes a chave deve ser mantida confidencial em ambos os extremos.
- $\ominus$ Numa comunidade de N agentes cada par de agentes necessita de uma chave própria. São necessários globalmente $N \times (N-1)$ chaves distintas; cada agente, individualmente, tem de garantir o segredo para $(N-1)$ dessas chaves.
- $\oplus$ Numa comunicação entre dois agentes é possível (recomendável) alterar a chave frequentemente. Possivelmente uma chave só deve ser usada num única comunicação; nesse caso chama-se uma **chave de sessão**.
- $\oplus$ As cifras simétricas são, computacionalmente, extremamente eficientes e a sua segurança pode ser garantida com chaves substancialmente menores dos que as usadas nas cifras assimétrica.

Assinaturas Digitais

Para garantir a *autenticidade* das comunicações usam-se **assinaturas digitais**. As mais recentes são as **não deterministas**: o mesmo texto x pode dar origem, de forma aleatória, a um número infinito de assinaturas.

x $\Rightarrow$ *texto*

s $\Rightarrow$ *assinatura*

β $\Rightarrow$ *informação pública*

u $\Rightarrow$ *informação privada*

A $\Rightarrow$ *emissor.*

B $\Rightarrow$ *verificador*

Esquema simples de assinaturas

$$\begin{tikzcd} A \arrow[rr, "{x\,,\,s}"] & & B \\ u & {} \arrow[dl, dash, squiggly] & \beta \\ I & & \end{tikzcd}$$

assinatura (não determinista)	$\Rightarrow$	$\mathsf{ass}_u : X \longrightarrow 2^{\mathcal{S}}$
teste de verificação	$\Rightarrow$	$\mathsf{ver}_\beta : X \times \mathcal{S} \longrightarrow \mathbb{B}$

CORRECÇÃO $\quad s \in \mathsf{ass}_u(x)$ sse $\mathsf{ver}_\beta(x,s) = 1$

CONFIDENCIALIDADE $\quad$ Apenas A conhece u.

NÃO INVERSÃO/SEGURANÇA

(i) Conhecidos β e x é computacionalmente intratável determinar u

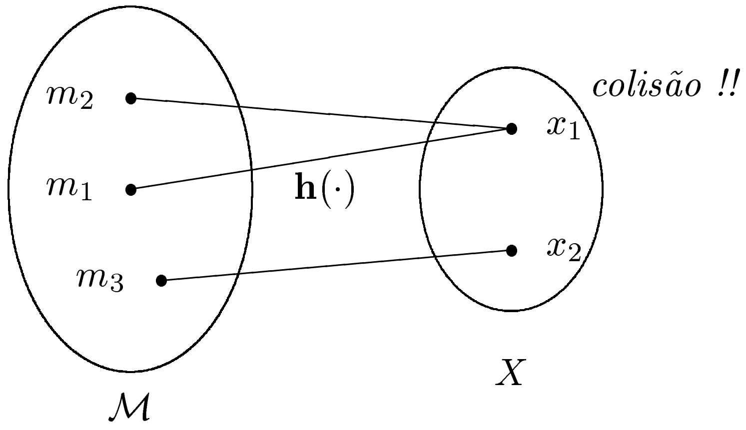
(ii) Conhecidos x e uma qualquer assinatura s de x, é computacionalmente intratável encontrar $x' \neq x$ tal que $\mathbf{ver}_\beta(x', s) = 1$.

$$\boxed{(x)_u \text{ denota um elemento aleatório } s \in \mathsf{ass}_u(x)}$$

Funções de Hash

Funções computacionalmente muito eficientes que mapeiam espaços de mensagens de tamanho arbitrário em códigos, chamados **valores de hash**, que são representáveis por um número pré-estipulado de bits. Tipicamente são usados *hash*'s de 128 bits e de 160 bits.

Sendo computacionalmente impraticável assinar directamente uma mensagem de tamanho realista é assinado, em sua substituição, o *hash* da mensagem.



$|\mathcal{M}| \gg |X|$

$|X| \sim 2^{160}$

Fortemente Livres de Colisões

É computacionalmente intratável encontrar mensagens $m \neq m'$ tais que $\mathbf{h}(m) = \mathbf{h}(m')$

Fracamente Livres de Colisões

Dada uma qualquer mensagem $m \in \mathcal{M}$, é computacionalmente intratável encontrar uma outra mensagem m', distinta de m, tal que $\mathbf{h}(m) = \mathbf{h}(m')$

$\overline{m}$ denota um valor de *hash* da mensagem m

Identificação de Agentes

O processo que permite o reconhecimento por uma agente B da entidade de um outro agente A.

$$
\begin{array}{ccc}
A & \xrightarrow{A\,,\,u} & B \\
u & & \mathsf{id} \\
I & & \mathsf{id}(A) = u
\end{array}
$$

O mecanismo **login-password**.

Muito Frágil !! o intruso I facilmente reenvia A e u.

$$
\begin{array}{ccc}
A & \xrightarrow{A\,,\,\beta} & B \\
u & & \mathsf{id}(A) = \beta \\
 & & \\
A & \xleftarrow{x} & B \\
 & & x \\
 & & \\
A & \xrightarrow{y} & B \\
y = f(u,x) & & x = g(\beta,y)
\end{array}
$$

O mecanismo **desafio-resposta**.

São públicas id, f e g

Correcção

$$x = g(\beta, y) \quad \text{sse} \quad y = f(u, x)$$

Não Inversão

Não é possível recuperar u de β, x e y.

Este mecanismo exige **certificação** da informação pública β associando-a ao agente A: a função id é certificada.

Certificação de Chaves Públicas

É o mecanismo que garante a associação correcta entre agentes e as respectivas chaves públicas.

É exigido por qualquer processo criptográfico que envolva uma componente de identificação (mesmo que essa não seja a sua função principal).

$$
\begin{tikzcd}
 & & B_1 \\
A \arrow[r, dotted, "A,\beta"'] & \mathsf{TA} \arrow[ur, dotted, "\mathsf{id}"] \arrow[r, dotted, "\mathsf{id}"'] \arrow[ddr, dotted, "\mathsf{id}"'] & B_2 \\
 & & \vdots \\
 & & B_n
\end{tikzcd}
$$

O mecanismo envolve uma **entidade certificadora** que assume a forma de um agente designado por TA (*trusted agent*). O TA recebe por *canal seguro* os diversos pares $\langle A, \beta \rangle$ e agrega-os numa função id que distribui, também por *canais seguros*, pelos diferentes agentes B_i interessados nessa identificação.

Destes canais seguros pretende-se que garantam a *integridade* da informação; para tal podem ser implementados de várias formas:

(i) Através da distribuição física directa e usando meios não electrónicos de certificação. É o que se usa normalmente na comunicação do par A, β.

(ii) Sob a forma de um canal virtual sob um canal aberto, distribuindo a função id por um conjunto de **certificados**: tuplos da forma

$$\langle\, A\,,\, \beta\,,\, (A,\beta)_{\mathsf{ta}}\,\rangle$$

Para isso a chave pública do TA deve ser comunicada aos diversos agentes B_i também por um canal seguro recorrendo a uma destas formas.

Geração, Distribução e Gestão de Chaves

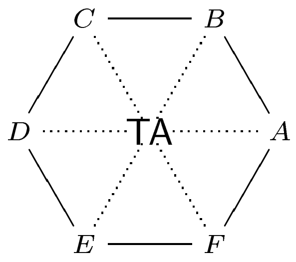
Toda a gestão de segredos passa por três fases, todas críticas em termos de segurança, e que têm de ser suportadas por técnicas adequadas:

Geração: o processo pelo qual uma chave adequire entidade e passa a ser conhecido pelo seu agente gerador.

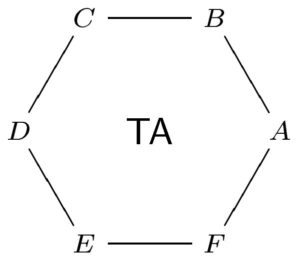
Distribuição: o processo pelo qual uma chave passa a ser conhecida por um, dois ou mais agentes.

As técnicas de distribuição dividem-se em duas grandes categorias:

Pré-distribuição: quando um agente priveligiado, o TA, envia de forma segura a chave a cada um dos receptores apropriados.



Acordo: quando dois ou mais agentes trocam entre si de forma segura, com o eventual contributo do TA para a identificação mútua, informação que permite a cada um reconstruir a chave partilhada sem conhecimento de qualquer outro agente (incluindo o TA).



Gestão: o conjunto de procedimentos que garante a manutenção, ao longo do tempo, de uma relação de confiança entre agentes e chaves incluindo, se necessário, a substituição de chaves.

Cifra por Substituição Simples

Dado um alfabeto $\mathcal{A}$ com N símbolos, a função de cifragem é uma permutação π desses símbolos. Uma mensagem m é uma sequência finita de símbolos

$$m \quad = \quad a_1\, a_2\, \ldots\, a_n$$

e é cifrada aplicando π a cada um dos elementos

$$\{m\}_\pi \quad = \quad \pi(a_1)\, \pi(a_2)\, \ldots\, \pi(a_n)$$

É uma cifra **mono-alfabética** porque a função de cifragem é aplicada símbolo a símbolo e cada símbolo a do texto aparece sempre cifrado como $\pi(a)$ no criptograma.

Consequência: é facilmente atacável por **análise de frequências** já a frequência de ocorrência típica de símbolo a no texto reproduz-se como a frequência do símbolo $\pi(a)$ no criptograma.

A chave da cifra é a própria permutação π .

Consequência: é facilmente atacável por procura exaustiva (*ataque por força bruta*) já que, existindo N símbolos em $\mathcal{A}$ existem apenas $N!$ chaves distintas.

Cifras por Substituição Homomórfica

Alfabeto do texto $\mathcal{A}$ e alfabeto do criptograma $\mathcal{B}$. Cada símbolo $a \in \mathcal{A}$ é associado a um conjunto $B(a) \subset \mathcal{B}$. Os conjuntos $B(a)$ são disjuntos entre si.

A mensagem m é cifrada numa sequência

$$\{m\} \quad = \quad b_1\, b_2\, \ldots\, b_n \qquad \text{com } b_i \text{ aleatório em } B(a_i)$$

Decifra-se $b \in \mathcal{B}$ procurando o único $a \in \mathcal{A}$ tal que $b \in B(a)$.

Pretende evitar ataques por frequência com escolha apropriada dos $B(a)$. A menos que seja $|\mathcal{B}| \gg |\mathcal{A}|$ (dificultando a decifragem) não melhora muito a segurança contra ataques por força bruta.

Cifras Poli-alfabéticas

Numa cifra **mono-alfabética** pura cada símbolo b_i do criptograma só depende do símbolo a_i do texto.

$$\begin{tikzcd} a_1 \arrow[d] & a_2 \arrow[d] & \cdots & a_n \arrow[d] \\ b_1 & b_2 & \cdots & b_n \end{tikzcd}$$

Numa cifra **poli-alfabética** pura cada símbolo do criptograma depende de <u>todos</u> os símbolos do texto claro.

$$\begin{tikzcd} a_1 \arrow[d] \arrow[dr] \arrow[drrr] & a_2 \arrow[dl] \arrow[d] \arrow[drr] & \cdots & a_n \arrow[dlll] \arrow[dll] \arrow[d] \\ b_1 & b_2 & \cdots & b_n \end{tikzcd}$$

Numa cifra **mono-alfabética** a transformação $a_i \mapsto b_i$ é independente de i.

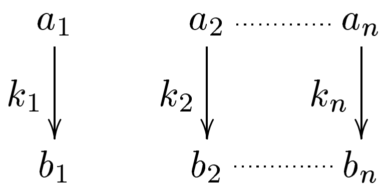
Numa cifra **poli-alfabética** pura a transformação $\begin{tikzcd} a_1 \arrow[dr] & a_2 \cdots \arrow[d] & a_n \arrow[dl] \\ & b_i & \end{tikzcd}$ depende de i

- $\oplus$ O mesmo símbolo, em posições diferentes, é cifrado de forma diferente. Imune aos ataques simples por análise de frequência. O ataque a um símbolo, mesmo se bem sucedido, não dá informação para os restantes.
- $\ominus$ Complexidade computacional: é preciso ler toda a mensagem antes que seja possível calcular o primeiro símbolo do criptograma. Impraticável para mensagens de tamanho realista.

Aproximações às cifras poli-alfabéticas

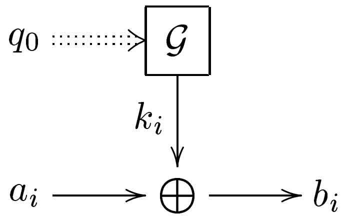
"One Time Pad"

Cada símbolo b_i depende apenas do símbolo a_i mas a chave de cifragem k_i é aleatóriamente gerada.



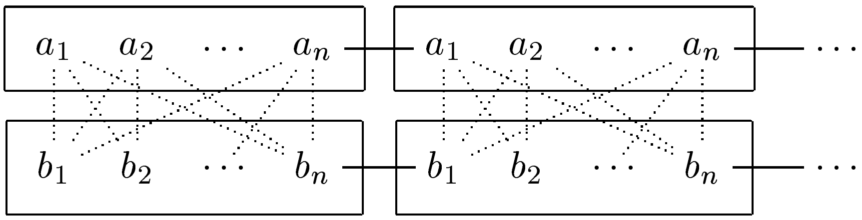
Cifras "Stream"

Uma aproximação do "one time pad" em que cada chave k_i é gerada por uma máquina de estados $\mathcal{G}$, o **gerador de chaves**, a partir de chaves e símbolos k_j, a_j anteriores.
O estado inicial q_0, de $\mathcal{G}$, é a chave da cifra.



Cifras por Blocos

A sequência de símbolos do texto claro é dividida em **blocos** de tamanho fixo. Cada bloco individualmente é tratado como uma mensagem completa cifrado com uma cifra poli-alfabética pura. Todos os blocos são cifrados do mesmo modo.



Se o tamanho do texto não for múltiplo do tamanho do bloco, ao último bloco são acrescentados símbolos adicionais, chamados **pad**'s, até atingir o tamanho pré determinado.

Técnicas Criptográficas– Módulo II

Cifras Simétricas

1. Classificação geral das cifras simétricas.
2. Cifras Sequenciais ("*stream*")
    1. Classificação geral
    2. "*Linear Feedback Shift Registers*" e sua segurança.
    3. Combinações não lineares de LFSR.
3. Cifras Simétricas por Blocos
    1. Modos de cifras e cifras múltiplas.
    2. Elementos básicos de cifras: operações de grupo, rotações, permutações, S-*Boxes* e Circuitos de Feistel.
    3. Estrutura base de uma cifra iterada.
4. A geração 64 bits:
    1. DES ("*Data Encryption Standard*"): descrição, propriedades e criptoanálise.
    2. Criptoanálise diferencial e linear.
    3. IDEA ("International Data Encryption Algorithm"): descrição e propriedades.
5. A geração 128 bits: o AES ("*Advanced Encryption Standard*"). Paradigmas de algoritmos concorrentes:
    1. `SAFER+`
    2. *Twofish*

Cifras Sequenciais ("Stream Ciphers")

Cifras vocacionadas para gerir a confidencialidade de sinais "em tempo real". Aplicam-se a sequências não limitadas de símbolos que frequentemente (mas não sempre!) são *bits*.

As funções de cifragem e decifragem combinam sequencialmente "streams" de chaves k_i com "streams" de código de texto a_i (na cifragem) ou do criptograma b_i (na decifragem).

cifragem

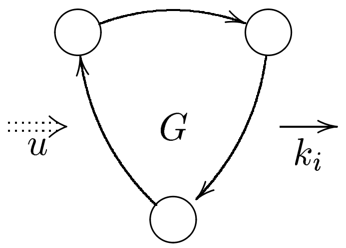
$$\begin{tikzcd} a_i \arrow[r] & \otimes \arrow[r] & b_i \\ & k_i \arrow[u] & \end{tikzcd}$$

decifragem

$$\begin{tikzcd} b_i \arrow[r] & \otimes' \arrow[r] & a_i \\ & k_i \arrow[u] & \end{tikzcd}$$

As várias classes de cifras "stream" distinguem-se pelo tipo de mecanismo usado na criação da sequência de chaves k_i: o **gerador de chaves**.

O gerador de chaves é um mecanismo, controlado por uma **chave de cifra** u, que emula sequências perfeitamente aleatórias de símbolos. Nomeadamente deve ser tal que

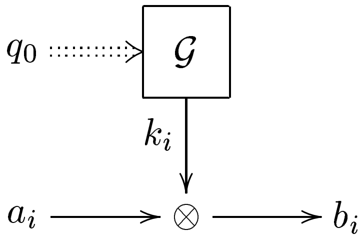


Segurança Do conhecimento de todos os k_j e a_j, com $j < i$, é computacionalmente intratável determinar k_i sem o conhecimento de u.

Existem duas grandes famílias de *stream ciphers*

Cifras Síncronas

O gerado $\mathcal{G}$ é uma máquina de estados determinista onde, no instante i, o estado q_i determina todas as saídas futuras k_j, $j \geq i$. Nomeadamente o estado inicial q_0 determina completamente toda a sequência de chaves; por isso q_0 é usado como chave da cifra.

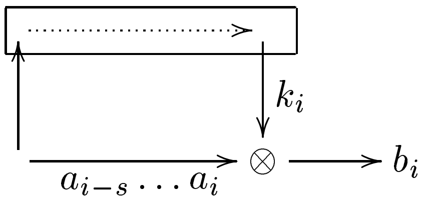


$\oplus$ Simplicidade, eficiência computacional, capacidade de prever propriedades de segurança. Não propaga erros.

$\ominus$ Exige mecanismos auxiliares complexos para manter o sincronismo entre a cifragem e a decifragem. A remoção ou repetição de trechos de criptograma altera o sincronismo e, observando os resultados dos mecanismos de re-sincronização assim activados, obter informação sobre o efeito dessas alterações no texto claro.

Cifras Auto-Sincronizáveis

A chave k_i é função de um número finito s de entradas anteriores a_j, com $i - s \leq j \leq i$, e é independente de chaves k_j.



$\oplus$ Simples mecanismos de auto-sincronização com perda de um número limitado de símbolos. Imprevisibilidade das chaves é garantida pelas entradas.

$\ominus$ Propagação de erros. Dificuldade em estudar as propriedades de segurança. Remoções ou repetições de trechos do criptograma são difíceis de detectar.

Período do Gerador de Chaves

Uma sequência de símbolos $\mathbf{k} = k_0 k_1 \ldots k_i \ldots$ tem **período** N se

$$k_{i+N} = k_i \qquad \text{para todo} \quad i \geq 0$$

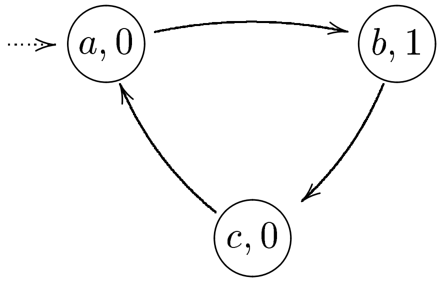
Uma sequência de período N pode ser organizada em *blocos* de N símbolos que se repetem infinitamente.

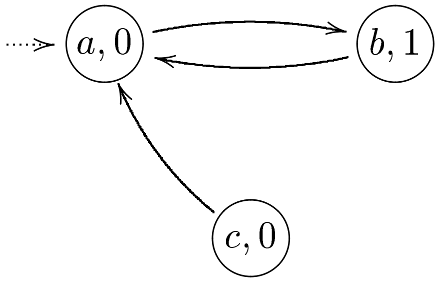
$$\underbrace{k_0 k_1 \ldots k_{N-1}}\,\underbrace{k_0 k_1 \ldots k_{N-1}} \ldots$$

A imprevisibilidade da sequência de chaves não é satisfeita. Um atacante que reconstrua o primeiro bloco pode sempre usá-lo na segunda vez para decifrar a mensagem.

A menos que o período N seja muito maior que o comprimento previsível agregado de todas as mensagens que sejam cifradas com a mesma sequência de chaves $\mathbf{k}$; neste caso todas essas mensagens são cifradas dentro do primeiro bloco e nenhum atacante tem a oportunidade de reutilizar o bloco de chaves para atacar mensagens (ou partes de mensagem) futuras.

Qualquer máquina de estados com N estados, determinista e sem entradas, é periódica e tem um *período óptimo* igual a N O período efectivo das sequências $\mathbf{k}$ geradas não ultrapassa esse valor óptimo mas pode ser consideravelmente inferior.





$\mathbf{k} = 010010010\ldots \qquad N = 3$ $\qquad \mathbf{k} = 01010101\ldots \qquad N = 2$

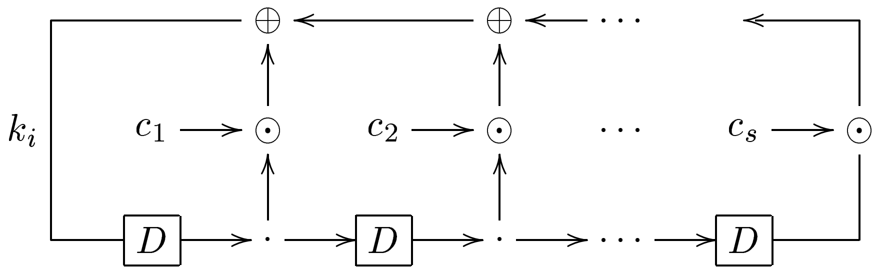
Linear Feedback Shift Registers (LFSR)

Um *Linear Feedback Shift Register* de comprimento s é dado por uma relação de recorrência

$$k_i + c_1 \cdot k_{i-1} + \ldots + c_s \cdot k_{i-s} \;=\; 0 \qquad i \geq s$$

com valores iniciais $k_0\,,\, k_1\,,\, \ldots\,,\, k_{s-1}$, definida no anel booleano $\mathbb{Z}_2$

$\mathbb{Z}_m$ é o anel dos números naturais $\{0, 1, \ldots, m-1\}$ com adição $+$ e multiplicação $\cdot$ efectuadas módulo m.



O polinómio no operador "*delay*" D

$$\mathcal{C}[D] \;\equiv\; 1 + c_1 \cdot D + c_2 \cdot D^2 + \ldots + c_s \cdot D^s$$

chama-se **polinómio característico** do LFSR. Se $\mathcal{C}[D]$ não for factorizável em $\mathbb{Z}_2$ nem dividir $1 + D^{2^s-1}$ então o polinómio diz-se **primitivo**.

Se $\mathcal{C}[D]$ é primitivo então, para qualquer vector de inicialização $\langle k_0, k_1, \ldots, k_{s-1}\rangle$, que não seja identicamente zero, a sequência de chaves k_i tem o período óptimo $2^s - 1$.

Segurança do LFSR

As sequências geradas directamente por um LFSR <u>não satisfazem</u> a condição de segurança. De facto

Se for conhecido $\mathcal{C}[D]$

A observação de s chaves $k_0 \ldots k_{s-1}$ permite determinar, pela relação de recorrência, todas as restantes chaves k_i , $i \geq s$.

Se não for conhecido $\mathcal{C}[D]$ mas for conhecido s.

A observação de $2\,s$ chaves $k_0 \ldots k_{2s-1}$ permite construir um sistema de equações lineares que determina os coeficientes $c_1, \ldots c_s$. Usam-se estes coeficientes para prever o resto da sequência.

$$k_3 = c_1 \cdot k_2 + c_2 \cdot k_1 + c_3 \cdot k_0$$
$$k_4 = c_1 \cdot k_3 + c_2 \cdot k_2 + c_3 \cdot k_1$$
$$k_5 = c_1 \cdot k_4 + c_2 \cdot k_3 + c_3 \cdot k_2$$

Se não for conhecido nem $\mathcal{C}[D]$ nem s.

Por tentativas usa-se o processo anterior para um determinado s e tenta-se verificar se o próximo bloco de s chaves é correctamente previsto. Se não for incrementa-se s uma unidade.

Se a "stream" observada for só "*aproximada*" por um LFSR

A **complexidade linear** de uma sequência de bits $\mathbf{k}$ é o tamanho s do menor LFSR que gera $\mathbf{k}$; se nenhum LFSR gerar $\mathbf{k}$ a sua complexidade linear é ∞.

Toda a sequência finita tem uma complexidade linear finita que pode ser facilmente determinada pelo *algoritmo de Berlekamp-Massey.*

Cifras "stream" baseadas em LFSR

Os LFSR são adequados ao uso em cifras

- $\oplus$ são de simples realização em *hardware* e têm um tratamento algébrico simples a partir do qual é possível prever e projectar as suas propriedades estatísticas, incluindo o período.
- $\ominus$ observada um prefixo da sequência de saída é possível prever toda a sequência.

A desvantagem indica que uma cifra que use directamente um LFSR como gerador de chaves é frágil porque é facilmente atacada por um *ataque de texto conhecido*: conhecido os a_i e os b_i correspondentes determina-se $k_i = a_i + b_i$.

É possível construir *shift registers* não lineares, mas a complecidade da construção e do tratamento algébrico desaconselham esta abordagem.

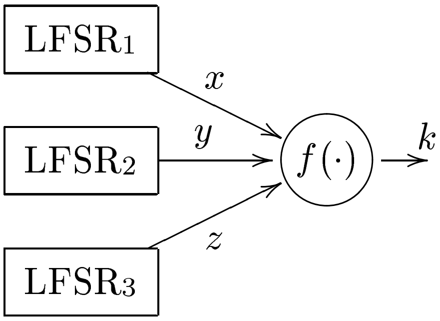
Solução: Combinações não lineares de LFSR

(i) Combinação não-linear das saídas de vários LFSR
(ii) Usar um filtro não linear na saída de um LFSR
(iii) Usar a saída de um LFSR para controlar o relógio de outros LFSR
(iv) Qualquer combinação das abordagens anteriores.

Pretende-se na saída do gerador:

(i) Períodos óptimos ou, pelo menos, muito longos.
(ii) Complexidade linear infinita ou muito grande.
(iii) Boa distribuição estatística, preferencialmente imprevisibilidade estatística

Combinação não-linear de várias saídas de LFSR's



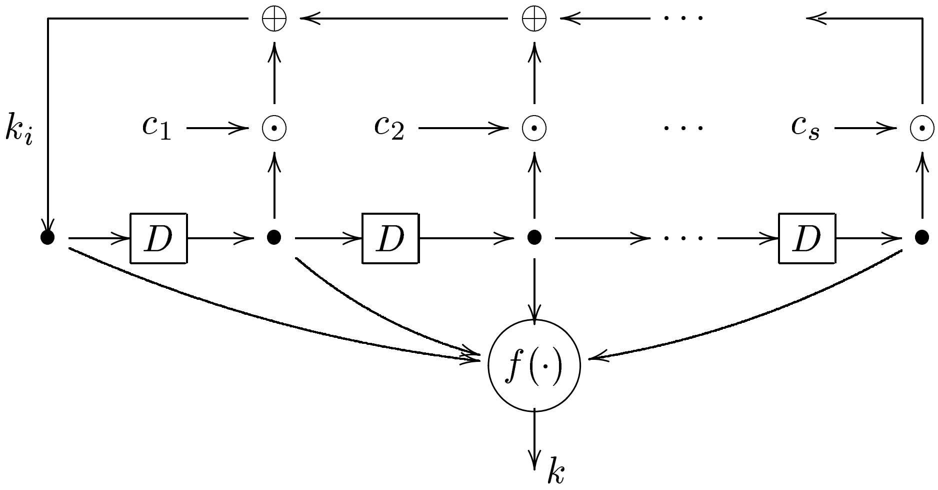
Alguns exemplos de funções $f(\cdot)$ não-lineares em $\mathbb{Z}_2$ são definidas por operações booleanas; ex:

$$k = f(x, y, z) = x \wedge (y \vee z)$$

As funções $f(\cdot)$ devem ser escolhidas de modo que a saída k não seja correlacionável estatisticamente com nenhuma das saídas dos LFSR. Se k for, com probabilidade $> 1/2$, igual a uma das saídas de LFSR então é possível atacar esse LFSR passando a prever o seu comportamento futuro e abrindo caminho a ataques similares nos restantes LFSR's.

(exemplos no `Handbook of Applied Cryptography`, Capítulo 6º)

Filtro não-linear da saída de um LFSR



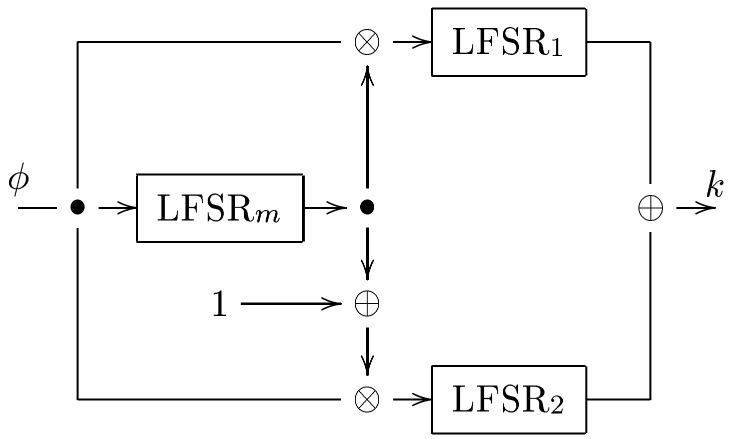
A função $f(\cdot)$ usa operadores booleanos sobre todos os estados do LFSR $k_i, k_{i-1}, \ldots, k_{i-s}$.

Combinação por controlo do relógio

Cada LFSR é controlado por um *relógio*. O LFSR só muda de estado quando recebe um 1 no pino ϕ do relógio.
Na cifra, um LFSR *mestre* controla o relógio de outros LFSR's secundários. As saídas deste são combinadas linearmente na saída.

Alternating Step

O LFSR mestre comanda, em 1, o LFSR_1 e em 0 o LFSR_2 .



(existem várias variantes deste esquema; ver `Applied Cryptography` de B.Schneier)

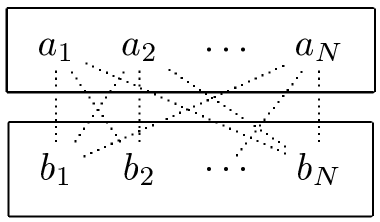
Cifras "stream" em software

Existem importantes cifras "stream", com geradores realizadas em *software*, que emitem blocos de n bits vistos como elementos de $\mathbb{Z}_{2^m}$ ou como elementos de ${\mathbb{Z}_2}^n$ e que são adicionados a blocos análogos para cifrar ou decifrar.

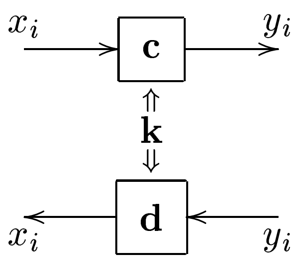
Os exemplos mais importantes são o RC4 (é proprietário e oficialmente secreto mas existe um programa em domínio público que aparentemente o substitui) e o SEAL (derivado das técnicas usadas nas funções de *hash* SHA e SHA-1; estas cifras são do domínio público).

Cifras Simétricos por Blocos

O texto claro e o criptograma são sequências de **blocos** de N bits. A cifra é aplicada bloco-a-bloco e, dentro de cada bloco, é uma cifra polialfabética pura controlada por uma única chave $\mathbf{k}$. A **eficiência** computacional é um dos objectivos essenciais.



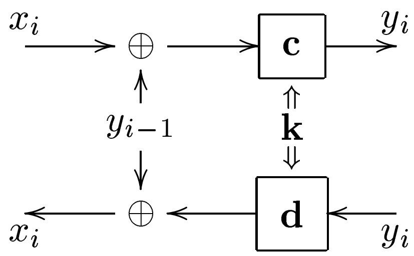
Modo ECB (*Electronic Code Book*)
O mesmo bloco x é cifrado sempre do mesmo modo: não propaga erros de cifragem.
No entanto:
A cifra não esconde **estereótipos** (secções do texto que se repetem em posições fixas), *e*



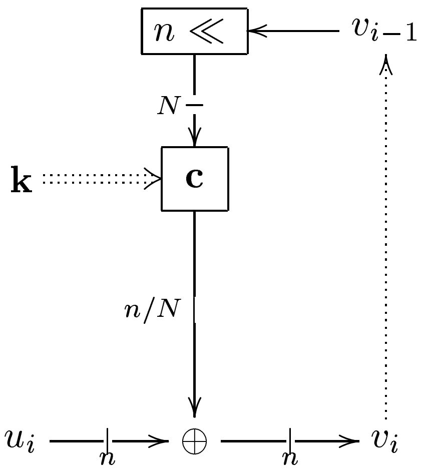
É vulnerável a **ataques por repetição de blocos**: o intruso força um bloco x (por si escolhido) a ser enviado, intercepta o criptograma y e reenvia-o repetidamente ao destinatário da mensagem.

Não se deve usar em mensagens maiores do que 1 bloco.

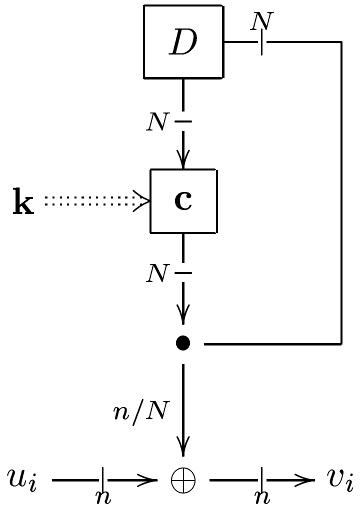
Modo CBC (*Cipher-block Chaining*)
Erros são corrigidos com atraso e perca de 1 bloco. Não recomendável a acessos de escrita/leitura aleatórios (p.ex. registos cifrados de DBMS).



Uma cifra por blocos de N bits pode ser usada para construir geradores de chaves de cifras "*stream*" para símbolos de n bits (com $n < N$).



Modo CFB (*Cipher Feedback*)

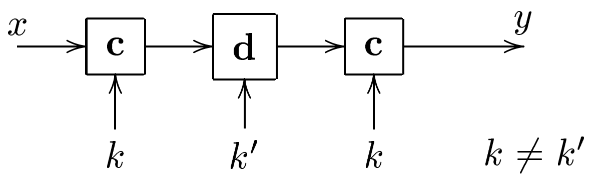


Modo OFB (*Output Feedback*)

Cifras Múltiplas

Numa chave **k** com N_k bits existe um **ataque por procura exaustiva** (*força bruta*) que sucede em 2^{N_k} tentativas.

Cifras em cascata sequenciam t cifragens e decifragens com chaves diferentes. O ataque por força bruta pode exigir $2^{t \times N_k}$ tentativas.



ex: *TripleDES*

O valor óptimo raramente é alcançado, porque:

- A cifra pode *constituir um grupo*: duas chaves consecutivas podem ser equivalentes a uma única chave,
- Existem ataques por *meet-in-the-middle* onde se caminha "dos extremos para dentro": p.ex. para $t = 2$, cifra-se x e decifra-se y num passo e comparam-se os resultados. O número de tentativas exigidas é consideravelmente diminuído.

Elementos básicos de cifras

As modernas cifras por blocos usam blocos múltiplos de 32 bits (64 é o mais frequente mas as novas propostas são para 128 bits), organizados em **sub-blocos** de 8 ou 16 bits que são processados por operações de **mistura** e **difusão** de bits segundo certas componentes elementares típicas.

Operações de Grupo

A cifra base usa códigos x, y e chaves k como elementos de um grupo $\mathcal{G}$.

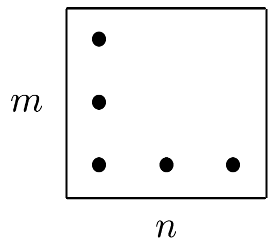
$$\mathbf{c}(x) = x \bullet k \qquad \mathbf{d}(y) = y \bullet k^{-1}$$

Exemplos comuns para sub-blocos de n bits ($n = 8$ ou $n = 16$):

- Grupo aditivo $(\mathbb{Z}_2)^n$: operação de grupo é a adição bit-a-bit ($\oplus$).
- Grupo aditivo $\mathbb{Z}_{2^n}$: operação de grupo é a adição módulo 2^n ($\boxplus$).
- Grupo multiplicativo $\mathbb{Z}_p^*$ de ordem 2^n ; ex. $p = 2^{16} + 1$ é um número primo, os inteiros $1, 2, \ldots, 2^{16}$ são representáveis com 16 bits e formam um grupo com a multiplicação módulo p ($\boxtimes$).

Cifras formadas por uma só operação de grupo são frágeis e vulneráveis a ataques de texto conhecido: são chamadas *cifras lineares.* Toda a cifra segura precisa de *elementos não-lineares.* Os mais vulgares são:

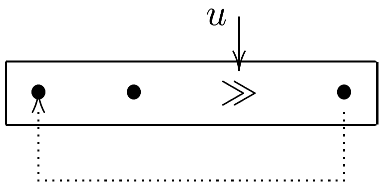
S-Box $m \times n$ Substituição não-linear com m-bits de entrada e n-bits de saída



É um memória estática com m bits de endereço e n bits de tamanho de célula cuja função é "*misturar bits*"; normalmente não são injectivas nem sequer sobrejectivas. A escolha das melhores S-Boxes é uma "*arte*" que tem em vista frustrar as técnicas de criptoanálise que veremos futuramente.

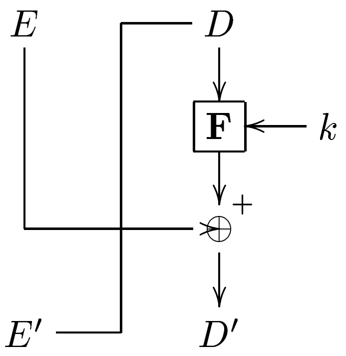
Permutações Formas particulares de S-Boxes $n \times n$ bijectivas: o bit na posição i à entrada passa a ocupar a posição j à saida. Fácilmente implementadas por uma tabela com n entradas em que cada entrada contém um elemento de $\mathbb{Z}_n$.

Rotações controladas Os N bits de uma célula de memória são rodados circularmente tantas posições quanto o valor u da entrada.



Circuitos de Feistel

Estrutura base de muitas cifras simétricas modernas.



A entrada de $2 \times n$ bits é dividida em dois sub-blocos de n bits, E e D (esquerdo e direito). A saída é organizada da mesma forma e calculada por:

$$E' = D$$

$$D' = E \oplus \mathbf{F}(D,k)$$

A chave k controla $\mathbf{F}(\cdot,k)$, a **função de Feistel**, que determina a cifra.

As relações fundamentais são:

$$\begin{pmatrix} E' \\ D' \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & \mathbf{F}(\cdot,k) \end{pmatrix} \begin{pmatrix} E \\ D \end{pmatrix} \qquad \begin{pmatrix} D \\ E \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & \mathbf{F}(\cdot,k) \end{pmatrix} \begin{pmatrix} D' \\ E' \end{pmatrix}$$

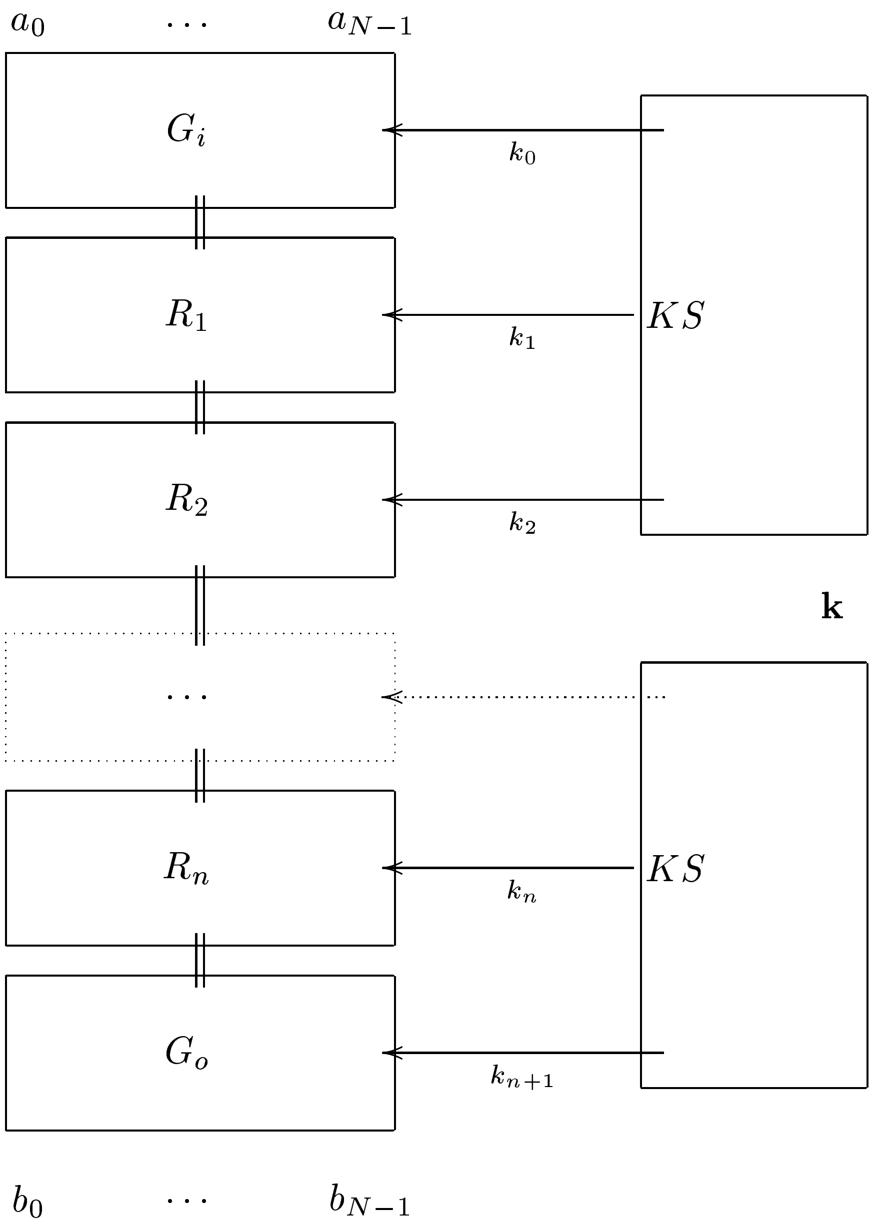
Donde

> A menos da ordem dos sub-blocos a função de decifragem é igual à função de cifragem.

Todas as modernas cifras simétricas, com ênfase na eficiência computacional, situam-se na classe das **Cifras de Blocos Iterados**.

- Cada bloco é processado iterativamente por uma **função de iteração** ("*round*" em inglês). O número de *rounds* necessários para que cada bit do texto claro seja alterado pelo menos uma vez, chama-se **ciclo** da cifra.
- O primeiro *round* é eventualmente antecedido de uma **pré-cifragem** e o último é eventualmente seguido de uma **pós-cifragem**.
- Todos os *rounds* e blocos de pré ou pós-cfiftragem são comandados por **sub-chaves** geradas a partir da **chave principal** por um **programador de chaves** ("*key scheduler*").
- Cada *round* é uma função injectiva, *one way*, computacionalmente muito simples e muito eficiente. Transforma cada bloco segundo objectivos de **mistura de bits** (cada sub-bloco é transformado juntamente com outros sub-blocos e com sub-chaves) e **difusão de bits** (cada bit à entrada vai influenciar cada um dos bits da saída nesse ciclo ou em ciclos futuros).
- Cada *round*, per si, não é particularmente seguro. A segurança deriva da **iteração** de *rounds* um número de ciclos sificientes para garantir as propriedades de mistura e difusão óptimas previstas na noção de cifra polialfabética pura:

Para todo o bloco de texto claro $x \in (\mathbb{Z}_2)^N$, cada um dos seus bits do $a_i \in \mathbb{Z}_2$, influencia cada um dos bits $b_j \in \mathbb{Z}_2$, do bloco cifrado e de modo diferente para cada um.



G_i, G_o $\equiv$ pré-cifragem e pós-cifragem

R_i $\equiv$ *rounds* de 1 a n

k_i $\equiv$ sub-chaves

KS $\equiv$ *key scheduler*

Decifragem

Em todas as cifras simétricas modernas os *rounds* têm uma função inversa dada por

$$R^{-1}(k) = T^{-1} \cdot R(k) \cdot T$$

para algum operador T. Portanto n *rounds* são invertíveis por

$$(R(k_n) \ldots R(k_1))^{-1} = T^{-1} \cdot R(k_1) \ldots R(k_n) \cdot T$$

Ou seja, *a função de decifragem é a mesma que a função de cifragem* mas:

(a) As sub-chaves são apresentadas pela ordem inversa,

(b) São introduzidas uma pré-cifragem T e uma pós-cifragem T^{-1}.

Se a própria cifra tiver uma pré-cifragem G_e e um pós-cifragem inversa da primeira (que é a norma), então

$$(G_e^{-1} \cdot R(k_n) \ldots R(k_1) \cdot G_e)^{-1} = (G_d)^{-1} \cdot R(k_1) \ldots R(k_n) \cdot G_d$$

com

$$G_d = T \cdot G_e$$

A pré-cifragem muda de G_e para $T \cdot G_e$ e as sub-chaves têm a ordem invertida.

DES (*Data Encription Standard*)

É o standard estabelecido em 1977 para cifras simétrica e usado universalmente no sector financeiro e no sector de serviços. Deverá ser brevemente substituído pelo **AES** (*Advanced Encription Standard*).

- Tem uma tamanho de bloco de 64 bits e um tamanho de chave de 56 bits úteis organizados em 8 bytes no formato 7+1 (1 bit de paridade). O pequeno espaço de chaves (2^{56}) é uma das suas fraquezas.

- É uma cifra de blocos iterada (como são todas!) com *rounds* que são circuitos de Feistel: uma **cifra de Feistel**. Como pré-cifragem tem uma permutação inicial fixa (não dependente de chaves) e a sua inversa como pós-cifragem. Na sua versão comercial base usa 8 ciclos.

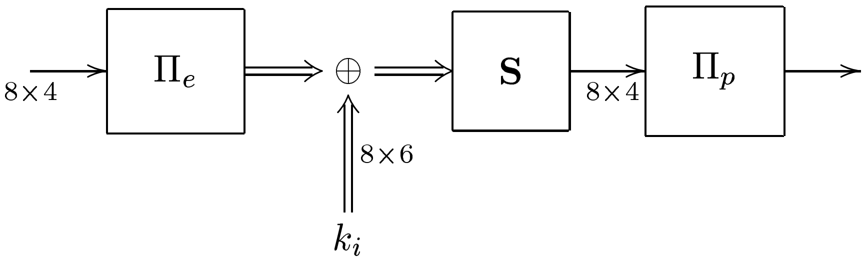
  Nota Em qualquer cifra de Feistel um ciclo necessita de 2 *rounds* já que cada *round* deixa meio bloco inalterado.

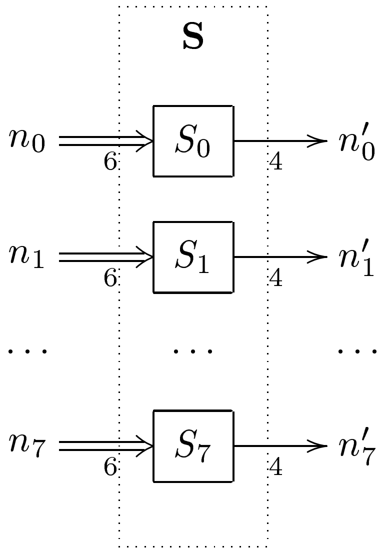
- O *key scheduler* gera, a partir da chave de cifra de 56 bits, 16 subchaves de 48 bits (uma por *round*).

- A permutação inicial (designada por Π_0 ou IP) e a sua inversa são permutações 64×64 implementadas em tabelas de 64 entradas e 6 bits cada: equivalente a uma S-box bijectiva 6×6 que opere nas posições dos bits.
  Como são fixas e conhecidas não intervêm na segurança da cifra; a sua utilidade é objecto de controvérsia.

DES - Função de Feistel

Recebe 32 bits (meio bloco) e uma sub-chave de 48 bits e devolve 32 bits.





Π_e expansão $32 \Rightarrow 48$ bits; diagonal formada por 8 instâncias da mesma expansão $4 \Rightarrow 6$ bits.
Π_p permutação 32×32 bits.
S diagonal formada por 8 S-Boxes diferentes 6×4. Esta é a única componente não linear da função.

Key Schedule

Os 56 bits da chave são divididos em metades de 28 bits que são rodadas um ou dois bits e passadas por permutações. Em cada passo 48 bits são aproveitados.

Ver detalhes no `Handbook of Applied Cryptography` onde também se encontram todas as restantes tabelas do DES (Π_0 e a sua inversa, Π_e, Π_p e as S-Boxes $S_0, \ldots, S_7$.

Decifragem no DES

Como em qualquer cifra de Feistel cada *round* verifica

$$R^{-1}(k) \;=\; T \,\cdot\, R(k) \,\cdot\, T$$

O operador T representa a troca entre as metades esquerda e direita. A decifragem é análoga à cifragem invertendo a ordem das chaves e trocando as metades esquerda e direita antes da permutação inicial e depois da permutação final.

Propriedades do DES

COMPLEMENTARIDADE $\mathbf{c}(\bar{x},\bar{k}) = \overline{\mathbf{c}(x,k)}$

CHAVES FRACAS
Com uma **chave fraca** o gerador de chaves produz uma sequência "em capicua": $k_1 = k_{16}, k_2 = k_{15}, \ldots$. Logo a cifra é uma **involução**: cifrar duas vezes consecutivas recupera o texto inicial.

Chaves fracas dão origem a **pontos fixos** (i.e., textos x tais que $\mathbf{c}(x,k) = x$) ou **anti-pontos fixos** (i.e., textos x que verificam $\mathbf{c}(x,k) = \bar{x}$).

Em qualquer dos casos o texto x recupera-se directamente do criptograma. O DES tem chaves fracas que dão origem a $\sim 2^{34}$ pontos fixos ou anti-pontos fixos.

DES NÃO É UM GRUPO
Genericamente, dadas duas chaves k_1 e k_2, não existe uma terceira chave k tal que $\mathbf{c}(\mathbf{c}(x,k_1),k_2) = \mathbf{c}(x,k)$.

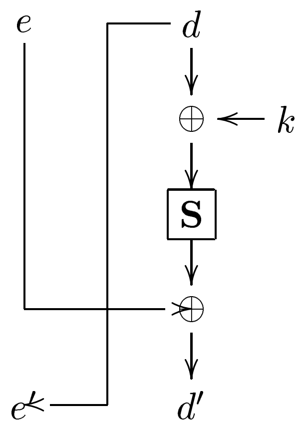
O TAMANHO DO ESPAÇO DE CHAVES É CURTO
É possível pensar em situações onde um ataque por força bruta a 2^{56} casos é viável.

CRIPTOANÁLISE
O DES é susceptível de ser atacado por criptoanálise linear e criptoanálise diferencial (veremos adiantes o que são estas técnicas).

Criptoanálise Diferencial

Analisa *diferenças entre pares de textos* e o modo como essas diferenças se propagam ao longo de várias iterações da cifra.

Como exemplo considere-se um modelo abstracto de um circuito de Feistel



determinado por uma única S-Box $\mathbf{S}$.

$$\langle\, e'\,,\, d'\,\rangle = \langle\, d\,,\, e \oplus \mathbf{S}(k \oplus d)\,\rangle$$

Para a mesma chave k, escolhem-se duas entradas $\langle\, e_0\,,\, d_0\,\rangle$ e $\langle\, e_1\,,\, d_1\,\rangle$ e observa-se a diferença entre os respectivos criptogramas.

$$\Delta \;=\; \langle\, e'_0\,,\, d'_0\,\rangle \oplus \langle\, e'_1\,,\, d'_1\,\rangle \qquad (\dagger)$$

Definindo

$$K \equiv k \oplus d_0 \qquad D \equiv d_0 \oplus d_1 \qquad E \equiv e_0 \oplus e_1$$
$$\Delta \equiv \langle\, E'\,,\, D'\,\rangle \qquad \mathbf{U}(K,D) \;\equiv\; \mathbf{S}(K) \oplus \mathbf{S}(K \oplus D)$$

tem-se

$$E' \;=\; D \qquad \bigwedge \qquad D' \;=\; E \oplus \mathbf{U}(K,D) \qquad (\ddagger)$$

Note-se que:

- Os valores e_0, d_0, e_1 e d_1 (ou, equivalentemente, E e D) podem ser escolhidos pelo atacante.
- Δ é determinado por ($\dagger$) observando a diferença nos criptogramas; daí determina-se E' e D'.
- Determinar a chave k é o objectivo. Para isso basta determinar K "resolvendo" ($\ddagger$).

A definição da S-Box **S** permite construir uma **tabela de inputs** definida do seguinte modo

$$\mathbf{inp}(u,v) \equiv \{x \mid u = \mathbf{U}(x,v)\}$$

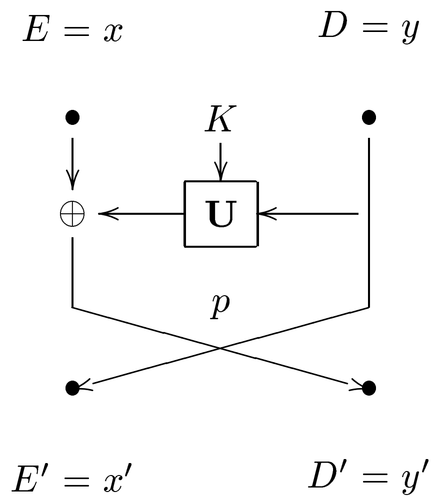
Para resolver (‡) basta calcular o conjunto

$$\mathbf{inp}(D' \oplus E, D)$$

para se obter uma *aproximação* à chave; i.e., um conjunto que contém o valor procurado K.

Repetindo a experiência com outros valores E e D vão-se calculando diversas aproximações: na intersecção de todas elas está K. Reduz-se esta intersecção até ser possível determinar K directamente ou por procura exaustiva.

Problema É normalmente impraticável ter toda a tabela **inp** armazenada.



Procura-se localizar **características** que são valores particulares x, x', y, y' para as diferenças E, D, E' e D' com uma propabilidade de ocorrer de p, sendo este um valor elevado.

Para estes a tabela é calculada. Um par de entradas que dá origem a diferenças que satisfazem a característica é usado para determinar, como anteriormente, aproximações à chave.

Criptoanálise Linear

Por vezes é possível construir aproximações lineares (em $\mathbb{Z}_2$) da chave a partir de bits das entradas e das saídas.

Seja k_i o bit de ordem i da chave k; com máscaras apropriadas m_i, m'_i, n_i, n'_i será

$$k_i \;=\; m_i \cdot e \oplus \cdot n_i \cdot d \oplus m'_i \cdot e' \oplus n'_i \cdot d'$$

com probabilidade p_i.

Caso seja $p_i > 1/2$ então repetidos experiências com vários valores e, d, e', d' permitem decidir o valor de k_i.

IDEA

O IDEA (*International Data Encryption Algorithm*) usa blocos de 64 bits e chaves de 128 bits.

A cifra é *orientada à palavra* de 16 bits. Sendo $\mathcal{W}$ o conjunto das palavras, o espaço de chaves é isomórfico com $\mathcal{W}^8$ e o espaço dos blocos (códigos) é isomórfico com $\mathcal{W}^4$

As palavras $X \in \mathcal{W}$ são operadas por três grupos:

- $\langle\, \mathbb{Z}_2^{16}\,, \oplus \,\rangle$ adição em $\mathbb{Z}_2$ bit-a-bit.
- $\langle\, \mathbb{Z}_p\,, + \,\rangle$ com $p = 2^{16}$ e a adição inteira módulo p
- $\langle\, \mathbb{Z}_q^*\,, \times \,\rangle$ com $q = 2^{16} + 1$, identificação do elemento 2^{16} com 0 e multiplicação inteira módulo q.

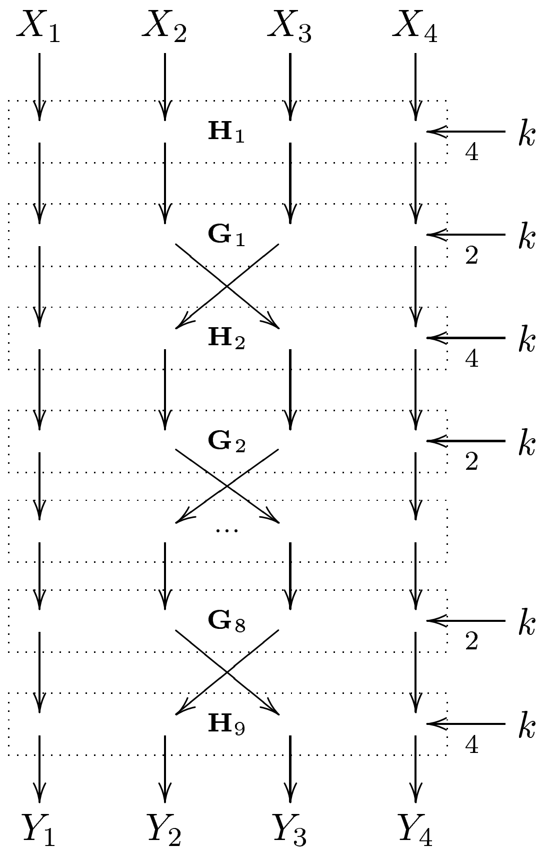
O IDEA é uma cifra iterada por blocos com 8 ciclos organizada do seguinte modo:

(a) Cada *round* é formada por uma *fase difusora* **H** seguida de uma *fase misturadora* **G**.

(b) Existe uma *pós-cifragem* análoga à fase difusora.

(c) Entre cada *round* são trocadas a segunda e a terceira componentes de cada bloco.

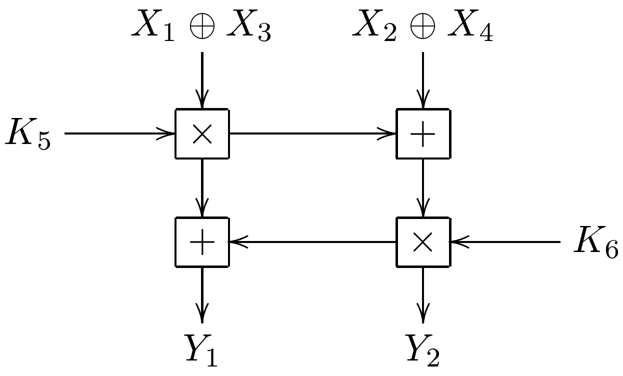
(d) O *programador de chaves* KS gera, a partir da chave de cifra, sub-chaves de 16 bits. Cada fase difusora consome 4 sub-chaves e cada fase misturadora consome 2 sub-chaves.



$\mathbf{H}_1$

$$X_1' = X_1 \times K_1 \quad X_4' = X_4 \times K_4$$
$$X_2' = X_2 + K_2 \quad X_3' = X_3 + K_3$$

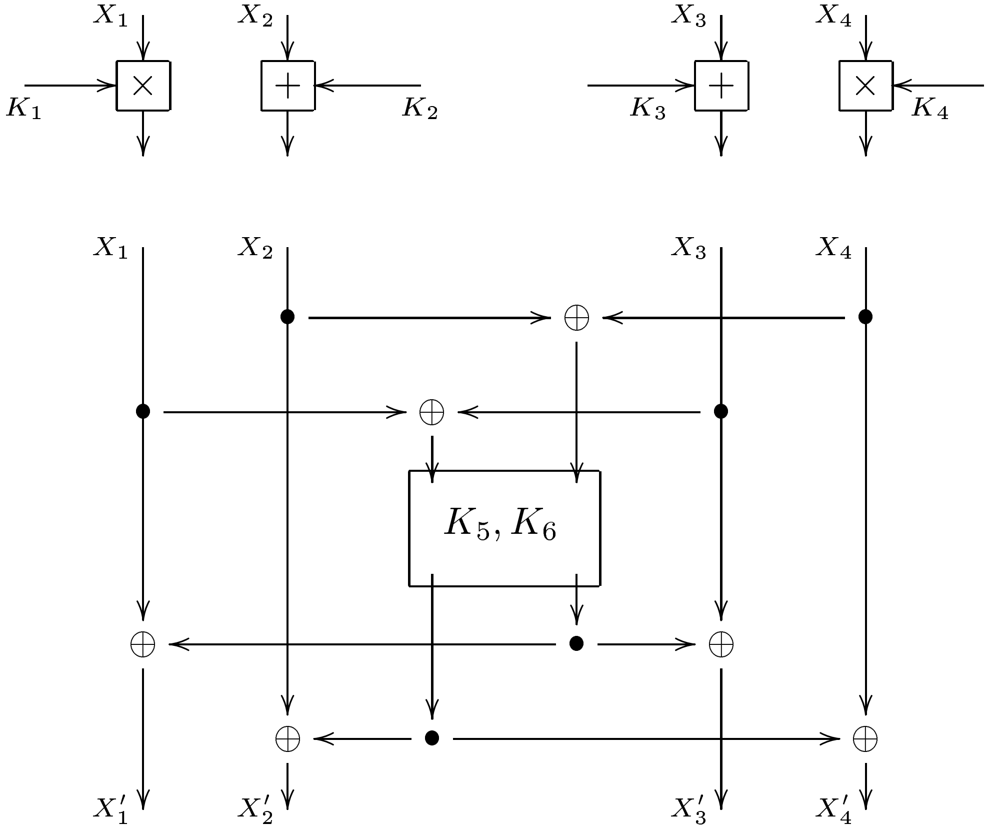
$\mathbf{G}_1$



MA-Box

$$X_1' = X_1 \oplus Y_2 \quad X_3' = X_3 \oplus Y_2$$
$$X_2' = X_2 \oplus Y_1 \quad X_4' = X_4 \oplus Y_1$$

Estrutura das fases H e G



Decifragem e Programador de Chaves

$$\begin{aligned}\mathbf{H}^{-1}(K_1,K_2,K_3,K_4) &= \mathbf{H}(K_1^{-1},-K_2,-K_3,K_4^{-1})\\ \mathbf{G}^{-1}(K_5,K_6) &= \mathbf{G}(K_5,K_6)\end{aligned}$$

A decifragem é igual à cifragem com a sequência de sub-chaves inversa da da cifragem e as sub-chaves da fase difusora substituídas pelos respectivos simétricos no grupo apropriado.

Dada a chave de cifra $\mathbf{k}$ são concatenadas 7 cópias desta chave

$$\mathbf{k}_0 \cdot \mathbf{k}_1 \cdot \ldots \cdot \mathbf{k}_6$$

com $\mathbf{k}_0 = \mathbf{k}$ e $\mathbf{k}_{i+1} = \gg_{25} \mathbf{k}_i$. Do *array* final são cortadas 52 "fatias" de 16 bits $(9 \times 4 + 8 \times 2 = 52)$ que constituem as sub-chaves.

AES

Decorre nesta altura (as propostas forma entregues em 15/Junho/98) um concurso promovido pelo NIST/FIPS para encontrar um *standard* de cifra rápida que suceda ao DES e que vigore, pelo menos, para os próximos 30 anos.

Chamar-se-a **Advanced Encryption Standard** e a decisão final está prevista para o ano 2000.

As características essenciais são:

- Será uma cifra de blocos de 128 bits,
- Deve fornecer a opção de trabalhar com chaves de 128, 192 e 256 bits
- Deve ter implementações eficientes em três linguagens (C de referência, C optimizado e Java) e em três tipos de plataforma: *smartcards* (ex. 6805), 32-bits *low-end* (ex. Pentium MX) e 32/64-bits *high-end* (ex. Pentium II/III, Alpha, ...).
- Deve ter implementações eficientes em *hardware*.

Devido ao tipos de aplicações previstas as implementações em *hardware* e em *software* de baixo custo são as mais importantes.

O algoritmo seleccionado deve ser público e não sujeito a direitos de propriedade.

Deve fornecer a segurança de pelo menos TripleDES

Após a primeira ronda de apresentações foram selecionadas 15 candidatos: CAST-256, Deal , HPC, MARS, RC6, Safer+, Twofish, Crypton, DFC, E2, Frog, LOKI97, Magenta, Rijndael e Serpent.

Referência: *The Block Cipher Lounge - AES*

`http://www.ii.uib.no/~larsr/aes.html`

SAFER+

A cifra Safer+ (*Secure and Fast Encryption Routine*) é classificada, pelos seus autores, como *substitution/linear-transformation cipher*.

É uma cifra de blocos iterada em cada *round* é formado, como usualmente, por uma *fase de mistura*, que aqui é uma substituição controlada por sub-chaves, e uma *fase de difusão*, aqui realizada por uma transformação linear invertível.

A cifra é **orientada ao *byte*** com operações em dois grupos: $\langle\, \mathbb{Z}_2^8 \,,\, \oplus \,\rangle$, com a adição em $\mathbb{Z}_2$ efectuada bit-a-bit, e $\langle\, \mathbb{Z}_{256} \,,\, + \,\rangle$, com a adição módulo 256.

Adicionalmente são usadas duas formas de operações específicas:

Explorando o facto de que 257 é um número primo, o grupo multiplicativo $\mathbb{Z}_{257}^*$ é cíclico e tem 256 elementos.: identificando 256 com 0, $\mathbb{Z}_{257}^*$ é representado por um byte.

O número 45 é **gerador** de $\mathbb{Z}_{257}^*$: i.e., a função $\mathsf{exp}_{45} : \mathbb{Z}_{256} \to \mathbb{Z}_{257}^*$

$$\mathsf{exp}_{45} \;:\; x \;\mapsto\; 45^x \pmod{257}$$

determina um isomorfismo. A sua inversa é representada por log_{45}.

A *Transformação Pseudo-Hadamard* de ordem 2 (2-PHT, em inglês) é uma transformação linear definida pela matriz H_2 caracterizada por

$$\mathsf{H}_2 = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix} \qquad \mathsf{H}_2^{-1} = \begin{bmatrix} 1 & -1 \\ -1 & 2 \end{bmatrix}$$

$$\begin{tikzcd} a \arrow[d, "2"'] \arrow[dr] & b \arrow[d] \arrow[dl] \\ 2a+b & a+b \end{tikzcd}$$

Safer+ usa 8, 12 ou 16 ciclos consoante a chave tem 128, 192 ou 256 bits. Após o último ciclo tem uma pós-cifragem.

Cada *round* ($\equiv$ a ciclo) é formado por uma fase misturadora, não-linear controlada por duas sub-chaves de 16 bytes, e uma fase difusora que é linear, invertível e não depende da chave.

A **Mistura** e **Difusão** são indicados a seguir.

A **pós-cifragem** é igual à primeira parte da mistura (fase das operações de grupo) e consome uma sub-chave de 16 bytes.

Decifragem

(i) Nas operações de grupo $\oplus$ e $+$ as fracções de sub-chaves são substituídas pelas suas simétricas no grupo.

(ii) exp tem por inverso log e vice-versa.

(iii) As transformações 2-PHT são substituídas pelas suas matrizes inversas e a permutação Π_A é substituída pela sua inversa.

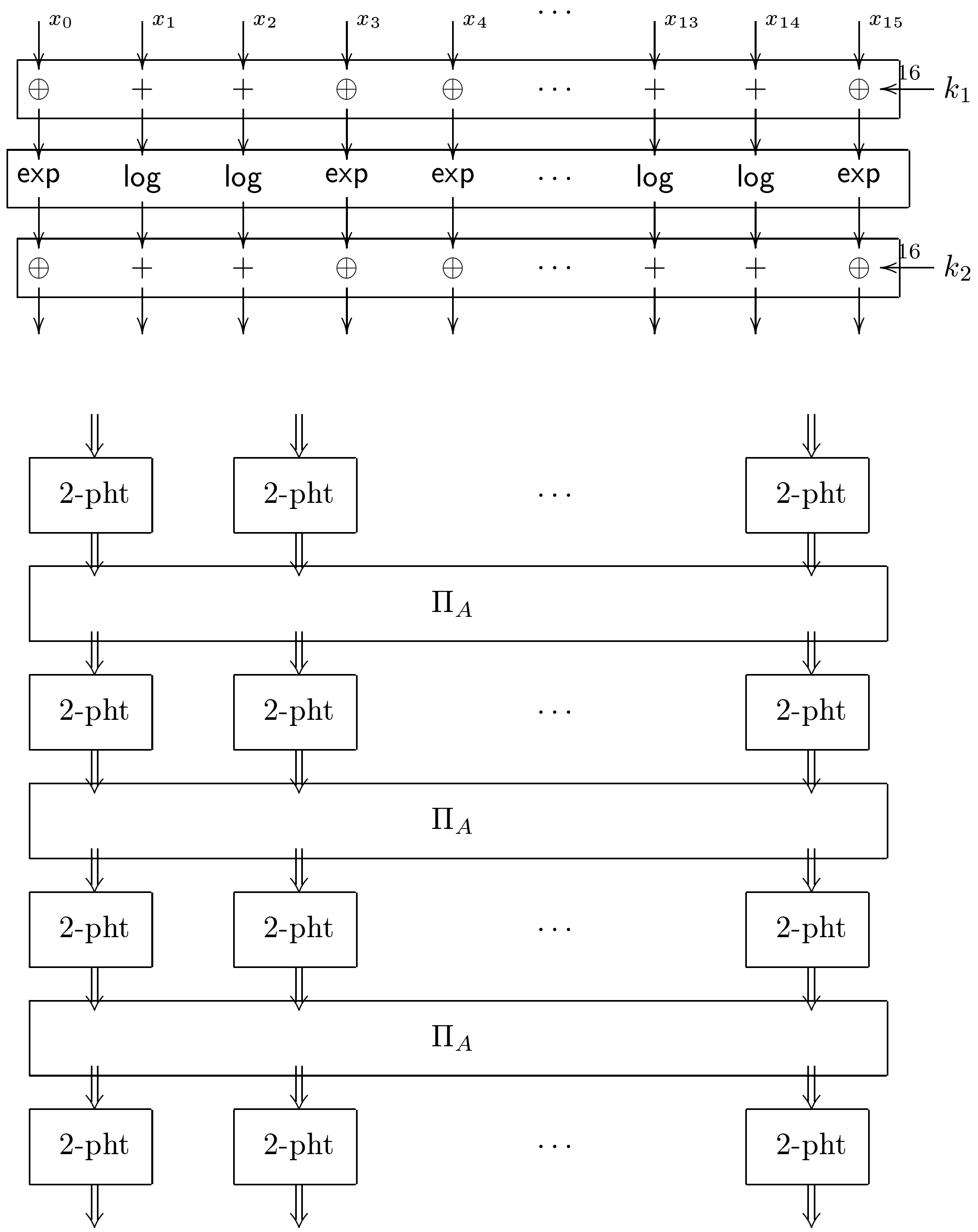
(iv) A sequência de sub-chaves é gerada pela ordem inversa.

Programador de Chaves

Depende do tamanho de chave e do número de *rounds* usados. A estratégia é essencialmente a mesma que no IDEA: rotação da chave de cifra e extracção de uma "fatia" de 16 bytes após cada rotação.

No Safer+ há ainda adição bit-a-bit (XOR) de vectores de *bias* constantes $B_0, B_1, \ldots$ a cada sub-chave.

Apresenta-se, como exemplo, o KS para chaves de 128 bits e 8 *rounds*.



Π_A é a permutação de bytes $\{9, 12, 13, 16, 3, 2, 7, 6, 11, 10, 15, 14, 1, 8, 5, 4\}$

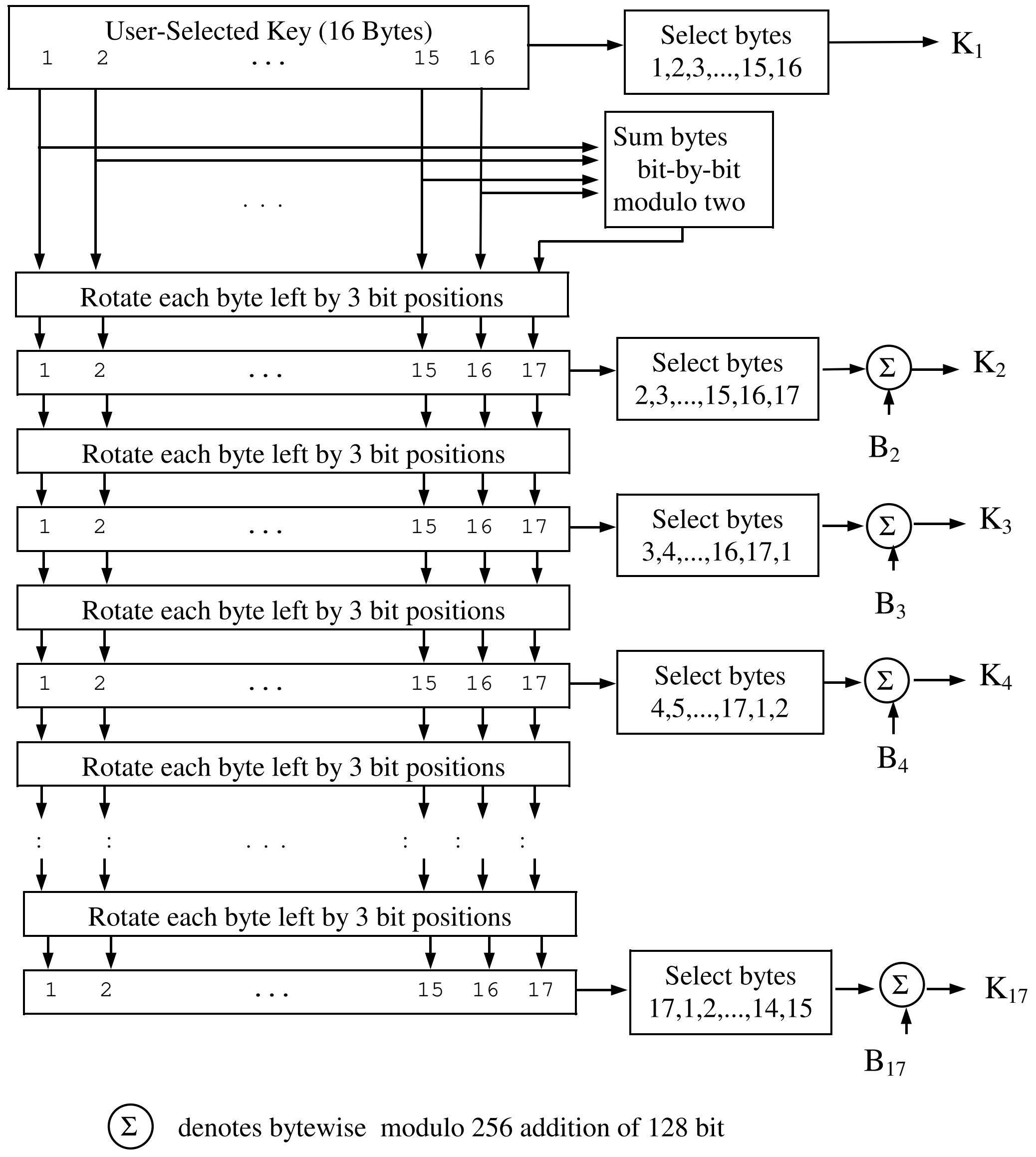


Fig. 4: SAFER+ key schedule for 128 bit key.

`Fonte` *Nomination of Safer+ as candidate for the Advanced Encryption Standard (AES)*, J.L.Massey *et al.*, 1998

Twofish

Outro dos candidatos ao AES, é uma evolução do **Blowfish** (de Schneier) e pretende ser extremamente eficiente em diversas plataformas.

É um circuito de Feistel com 16 *rounds*.

Cada *round* é um circuito de Feistel antecedido com um pré-processamento sobre a forma de "branqueamento" (*withening*) do bloco de entrada por chaves.

A função de Feistel actua sobre dois sub-blocos de 32 bits e é formada, como usualmente, por uma fase misturadora (não controlada por chaves) e uma fase difusora controlada por chaves.

Os dois sub-blocos de 32 bits não sujeitos à função de Feistel são objecto de rotações de 1 bit.

A fase misturadora transforma cada sub-bloco de 32 bits por 4 S-Boxes 8×8. O segundo desses sub-blocos é sujeito a uma rotação de 8 bits.

A fase difusora tem duas componentes:

(a) uma matriz MDS (*maximum distance sparable*) que tem a propriedade de oferecer a máxima "distância" (n^o de bits de diferença) entre saídas correspondentes a pares de entradas que se distingam em 1 bit,

(b) uma 2-PHT juntando os dois sub-blocos,

Existe um pós-processamento de "branqueamento" análogo ao pré-processamento de cada *round*

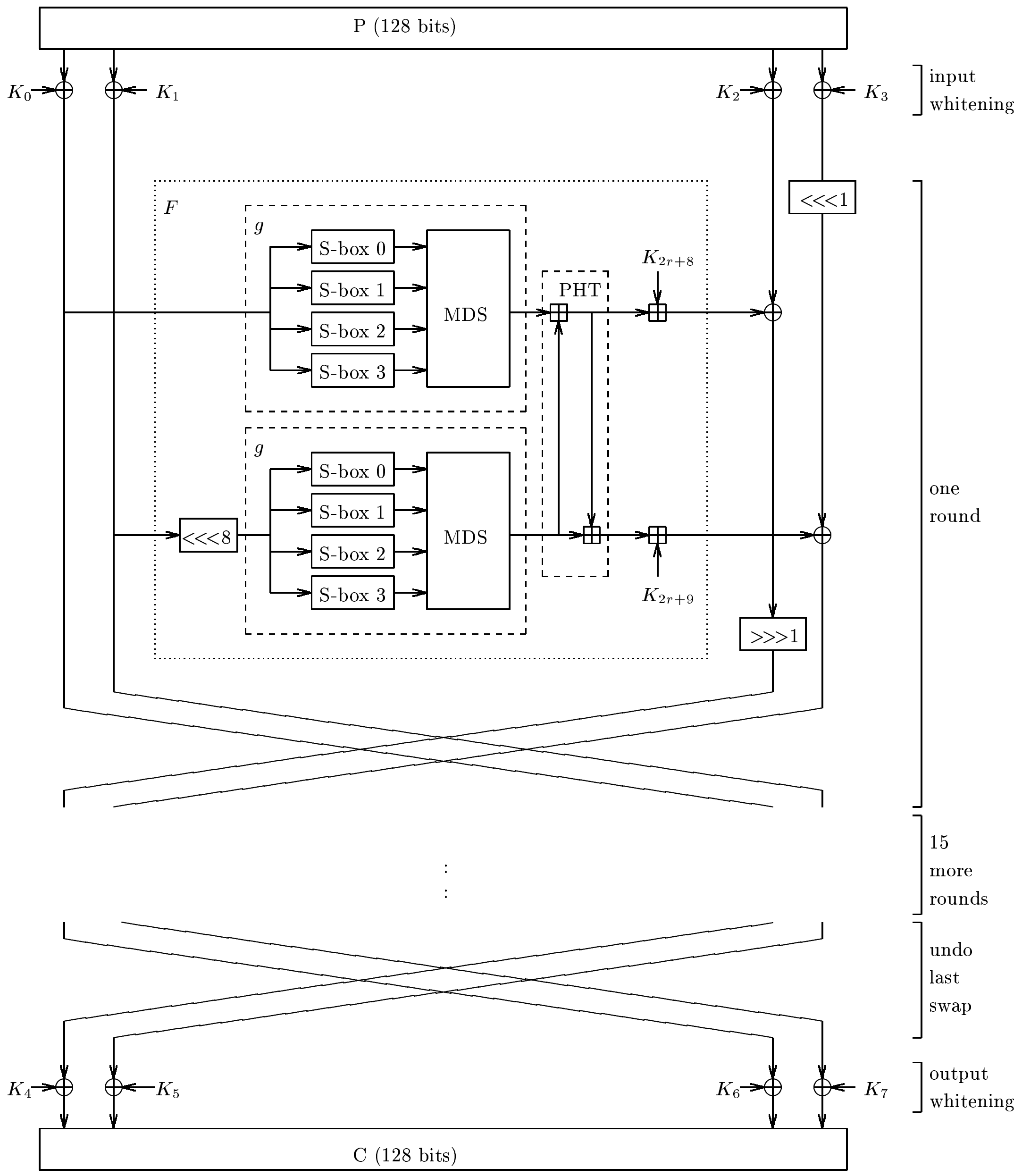


Figure 1: Twofish

Fonte *Twofish: a 128-bit block cipher*, B.Schneier *et al.*, 1998

Técnicas Criptográficas– Módulo III

Introdução à Teoria dos Números

1. Teoremas base da aritmética.
2. Noção geral de grupo cíclico. O grupo cíclico $\mathbb{Z}_p^*$.
3. Corpos finitos: os corpos de Galois $\mathbf{GF}(2^n)$.
4. Curvas Elípticas.
5. Algoritmos base da Teoria dos Números.
6. Teste e geração de números primos.
7. Geração de números pseudo-aleatórios.
8. Factorização de inteiros: factorização de Fermat e de Pollard. Factorização quadrática e por curvas elípticas (breve referência).
9. O problemas do Logaritmo Discreto nos inteiros. Algoritmos triviais, algoritmo de Shank, algoritmo de Pohlig-Hellman e referência ao método do índice. Método **rho** de Pollard.

Teoria dos Números

NOTAÇÃO:

$\mathbb{Z}$ 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

+	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

Exemplo: $\mathbb{Z}_5$

×	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

$\mathbb{Z}_n^*$ o **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 $a \times b = 1 \pmod{n}$.

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

Notas

§- Sendo p primo todos os elementos de $\mathbb{Z}_p$, excepto 0, são invertíveis.

§- $\mathbb{Z}_9^* \equiv \{1, 2, 4, 5, 7, 8\}$ com $2^{-1} = 5$ $4^{-1} = 7$ $8^{-1} = 8$

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

$g^i \pmod{9}$

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

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

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

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

2 Teorema (pequeno de Fermat) *Se* p *é primo então*

$$a^p = a \pmod{p} \qquad \textit{para todo} \quad a \geq 0$$

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

$$\phi(m) \;\doteq\; |\mathbb{Z}_m^*|$$

Propriedades da função de Euler-phi

(1) $\phi(m) \;=\; \prod_{i=1}^{k} \; p_i^{(e_i - 1)} \times (p_i - 1)$ se $m > 1$.

(2) $a^{\phi(m)} = 1 \pmod{m}$ para todo em $a \in \mathbb{Z}_m^*$ (*Teorema de Euler*)

(3) $i = j \pmod{\phi(m)} \quad \Longrightarrow \quad x^i = x^j \pmod{m}$

para todo $x \geq 0$, se m for um produto de primos distintos.
Em particular (*Teorema do RSA*)

$$a = b^{-1} \pmod{\phi(m)} \quad \Longrightarrow \quad (x^a)^b = x \pmod{m}$$

(4) $m = \sum_{a|m} \phi(a)$

A notação $a|m$ representa a asserção "a é um **divisor** de m"

2 DEFINIÇÃO *A **ordem** de* $a \in \mathbb{Z}_m^*$ *é o menor* $t > 0$ *tal que*

$$a^t = 1 \pmod{m}$$

Se $t \equiv \phi(m)$, a *diz-se **elemento primitivo** de* $\mathbb{Z}_m^*$

2 FACTO

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

Nota *Um grupo multiplicativo* $\langle\, G\,,\,\cdot\,\rangle$ *é **cíclico** se existe* $g \in G$ *(chamado **gerador do grupo**) de tal forma que a exponenciação* $i \mapsto g^i$ *estabelece um isomorfismo entre o grupo aditivo* $\mathbb{Z}_{|G|}$ *e* G*:*

$$G \equiv \{g^0\,,\, g^1\,,\,\cdots\,,\, g^{|G|-1}\} \qquad g^{i+j} = g^i \cdot g^j \qquad g^0 = 1$$

A cardinalidade de G *(representado por* $|G|$*) chama-se a **ordem** de* G*. Se* $|G|$ *é finito qualquer subgrupo de* G *tem uma ordem que é divisor de* $|G|$ *(*Teorema de Lagrange*).*

(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 $\phi(21) = 12$).

As propriedades do grupo cíclico $\mathbb{Z}_p^*$, quando p é um *grande primo* ($\sim 2^{512}$), estão na base das mais recentes técnicas assimétricas (cifras e assinaturas) da família Diffie-Hellman, El Gamal, DSA, etc.

Grupos cíclicos, como $\mathbb{Z}_p^*$ são essenciais à moderna criptologia. Será possível extender o conceito?

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, $\mathbb{Z}_p \setminus \{0\} \equiv \mathbb{Z}_p^*$

3 FACTO *Seja $\mathbf{F}_m$ um corpo finito com m elementos.*

(a) Se $\mathbf{F}_m$ tem característica $p \neq 0$ então p é primo e existe uma dimensão n tal que $m = p^n$ e

$$\mathbf{F}_m \sim (\mathbb{Z}_p)^n$$

(b) O grupo multiplicativo $\mathbf{F}_m^*$, formado pelos elementos invertíveis de $\mathbf{F}_m$, é cíclico e existem $\phi(m-1)$ geradores diferentes deste grupo.

(c) O conjunto $\mathbf{F}_m$ é identificável com o conjunto das raízes do polinómio $X^m - X$

(d) Existe um polinómio primitivo $\mathbf{c}[X]$ de grau n tal que

$$\mathbf{F}_m \sim \mathbb{Z}_p[X]/\mathbf{c}[X]$$

Nota $\mathbb{Z}_p[X]/\mathbf{c}[X]$ é o anel dos polinómios de coeficientes em $\mathbb{Z}_p$ em que a adição e multiplicação de polinómios é efectuada módulo $\mathbf{c}[X]$.

Um polinómio em $\mathbb{Z}_p[X]$, mónico e irreduzível, é **primitivo** em $\mathbf{F}_m$ se divide $X^m - X$ mas não divide nenhum $X^k - X$ para $k < m$.

Corpos de Galois $\mathsf{GF}(2^m)$

São os corpos finitos de característica 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	1001
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

$\mathbf{GF}(2^4)$ determinado pelo polinómio característico

$$\mathbf{c}[X] = (X^4 + X + 1)$$

Admite uma representação polinomial e uma representação binária. A segunda é mais vantajosa para somas e a primeira para multiplicações.

As operações de multiplicação, soma e módulo são facilmente implementadas em *hardware*. Uma das técnicas possíveis para multiplicação e módulo recorre aos LFSR.

O gerador do grupo cíclico $\mathsf{GF}(2^m)^*$ é muito simples de encontrar: é o polinómio X.

Exemplo: codificação dos inteiros $\mathbb{Z}_{16}$ no corpo $\mathsf{GF}(2^4)$.

$$7 + 11 = 0111 + 1011 = 1100 = 12$$

$$7 * 11 = 0111 * 1011 = X^{10} * X^7 = X^{17} = X^2 = 0100 = 4$$

$$(9)^2 = (1001)^2 = (X^{14})^2 = X^{28} = X^{13} = X^3 + X^2 + 1 = 1101 = 13$$

$$9^{-1} = (X^{14})^{-1} = X^{-14} = X^1 = 0010 = 2$$

Curvas Elípticas

3 Definição *Dado um corpo finito* $\mathbf{F}_p$, *sendo* $p > 3$ *um primo, e seja*

$$Q[x] = x^3 + \mu\, x + \upsilon \qquad \text{com} \quad \upsilon, \mu \in \mathbf{F}_p$$

um polinómio sobre $\mathbf{F}_p$ *sem raízes múltiplas. A curva elíptica definida por* $Q[x]$ *é o conjunto de soluções* $\langle\, a\,,\, b\,\rangle \in \mathbf{F}_p \times \mathbf{F}_p$ *da equação*

$$b^2 = Q[a]$$

acrescentado de um ponto especial $\mathcal{O}$ *chamado* ***ponto no infinito***

É possível generalizar a definição para corpos (finitos ou não) com qualquer característica; nomeadamente são importantes as curvas elípticas sobre corpos de Galois $\mathbf{GF}(2^n)$. A forma do polinómio é, no entanto, ligeiramente diferente e por isso as diferentes computações são diferentes das que são aqui apresentadas (ver `Koblitz` *A Course in Number Theory and Cryprograpgy*)

4 Facto *Toda a curva elíptica sobre* $\mathbf{F}_p$ *ddetermina um grupo abeliano finito com adição definida por*

1- $\langle\, a\,,\, b\,\rangle + \mathcal{O} = \mathcal{O} + \langle\, a\,,\, b\,\rangle = \langle\, a\,,\, b\,\rangle$

2- $\langle\, a\,,\, b\,\rangle + \langle\, a\,,\, -b\,\rangle = \mathcal{O}$

3- $\langle\, a\,,\, b\,\rangle + \langle\, a\,,\, b\,\rangle = \langle\, \lambda^2 - 2\,a\,,\, \lambda(3\,a - \lambda^2) - b\,\rangle$ *com* $\lambda = (3\,a^2 + \mu)/(2\,b)$.

4- $\langle\, a\,,\, b\,\rangle + \langle\, a'\,,\, b'\,\rangle = \langle\, \lambda^2 - a - a'\,,\, \lambda(2\,a + a' - \lambda^2) - b\,\rangle$ *com* $\lambda = (b - b')/(a - a')$ *se* $a \neq a'$.

5 FACTO *O número de pontos N da curva elíptica determinada por $Q[x]$ é*

$$N \;=\; (p+1) + \sum_{x \in \mathbf{F}_p} \left(\frac{x}{p}\right)$$

N é aproximado por

$$|N - (p+1)| \leq 2\sqrt{p}$$

O **símbolo de Legendre** $\left(\frac{q}{p}\right)$ é definido como sendo 0 se $q|p$, 1 se existe s tal que $s^2 = q \pmod p$, (q é um **resíduo quadrático** módulo p) e -1 em caso contrário.

6 FACTO *Se N é primo então a curva elíptica é um grupo cíclico isomórfico com $\mathbb{Z}_N$ através da "exponenciação" $i \mapsto i \cdot \alpha$ definido por*

$$i \cdot \alpha \;=\; \underbrace{\alpha + \alpha + \cdots + \alpha}_{i \text{ vezes}}$$

em que o gerador α é qualquer elemento do grupo diferente de $\mathcal{O}$.

Se $n = \gcd(N, p-1) \neq 1$ então existe um subgrupo cíclico, de ordem $N' = N/n$, isomórfico com $\mathbb{Z}_{N'}$ através da mesma "exponenciação".

Todas as técnicas criptográficas baseadas nas propriedades de *grupos cíclicos multiplicativos* podem ser adaptadas a curvas elípticas como *grupos cíclicos aditivos.*

A vantagem destas últimas é a **flexibilidade** na escolha do grupo cíclico: a variedade de curvas elípticas é muito grande. É mais fácil adaptar uma curva a um problema particular do que escolher um corpo finito para a mesma função.

Técnicas baseadas em curvas elípticas são, em geral, mais dificilmente invertíveis do que as baseadas em corpos finitos.

As desvantagens são o esforço computacional e, principalmente, as dificuldades de codificação.

Suponhamos que queremos codificar um texto $x \in X$ num ponto $\langle\, a\,,\, b\,\rangle$ de uma curva elíptica sobre $\mathbf{F}_p$.

Os pares $\langle\, a\,,\, b\,\rangle$ que pertencem à curva distribuem-se em $\mathbf{F}_p \times \mathbf{F}_p$ de modo aproximadamente aleatório (esta é vantagem em termos de segurança); logo não há garantia que uma transformação fixa $X \to \mathbf{F}_p \times \mathbf{F}_p$ produza pares de valores que sejam pontos da curva.

A estratégia está em associar a cada x uma "nuvem" de pares $\langle\, a\,,\, b\,\rangle$ de modo que as nuvens sejam disjuntas e exista pelo menos um elemento em cada nuvem com elevada probabilidade de pertencer à curva elíptica.

(i) Escolhemos um "tamanho de nuvem" P e, a cada x associamos a nuvem de elementos $a_i = x\,P + i$ com $i = 0, 1, \ldots, P-1$. Para se poder codificar todo X em $\mathbf{F}_p$ temos de ter $p > P\,|X|$.

(ii) Sucessivamente calculamos os vários $Q[a_i]$ e calculamos o símbolo de Legendre $\left(\dfrac{Q[a_i]}{p}\right)$ até obtermos um resíduo quadrático.

(iii) Para esse a_i determinamos a raíz quadrada módulo p de $Q[x_i]$; isto é, determinamos b tal que $b^2 = Q[a_i] \pmod p$. O par $\langle\, a_i\,,\, b\,\rangle$ é a codificação pretendida.

Nota Existem algoritmos eficientes para determinar tanto o símbolo de Legendre como a raíz quadrada. Em particular quando $4|(p+1)$, a raíz quadrada de a é dada directamente por $a^{(p+1)/4} \pmod p$ (consequência do teorema de Fermat).

Prova-se que a probabilidade de não encontrar, por este método, um ponto da curva eliptíca é menor do que 2^{-P}. Por isso um valor de P na ordem de 30 a 50 é suficiente para a maioria das aplicações.

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 }
até { 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.

Geração de Números Pseudo-Aleatórios

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 (**gerador linear**)

$$\begin{aligned} s_k &= a \times s_{k-1} + b & \qquad a, b, s_k \in \mathbb{Z}_p, \quad a \neq 0 \\ x_k &= s_k \pmod 2 & \qquad x_k \in \mathbb{Z}_2 \end{aligned}$$

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.

Geradores ad-hoc

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.*.

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 Para $m = 1127843$ temos o valor inicial $\mu = \lceil(\sqrt{m})\rceil = 1062$; como $m - \mu^2 = 1$ é logo um quadrado perfeito, conclui-se $\upsilon = 1$ e $\mu = 1062$. Assim $p = 1063$ e $q = 1061$.

Factorização de Pollard

Gerando uma sequência pseudo-aleatória finita $\rho_0, \rho_1, \ldots \rho_n$ em $\mathbb{Z}_m$, calculam-se os valores $d_i = \gcd(\rho_i, m)$ até se obter algum $d_s > 1$ (como resultado do processo) ou se atingir o limite da sequência dos ρ_i (com indicação de falha).

Método rho Os ρ_i são determinadospor um gerador quadrático

$$x_i = x_{i-1}^2 + 1 \pmod{m} \qquad y_i = (y_{i-1}^2 + 1)^2 + 1 \pmod{m} \qquad (\dagger)$$
$$\rho_i = x_i - y_i \pmod{m}$$

Justificação Se $x_0 = y_0$ então $y_i = x_{2i}$, para todo $i \geq 0$: a solução é atingida quando for $x_{2s} = x_s \pmod{p}$ para algum p (desconhecido) divisor de m.

Todo o eventual p, divisor de m, determina uma sequência $x'_i = x_i \pmod{p}$ que verifica a mesma igualdade ($\dagger$) 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 $(p-1)$

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

7 FACTO *Se se verificar*

$$x^2 = y^2 \pmod{m} \qquad \& \qquad x \neq \pm y \pmod{m} \tag{$\dagger$}$$

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.

Logaritmo Discreto

A segurança de muitas técnicas criptográficas depende crucialmente da característica *one-way* da "exponenciação" em **grupos cíclicos**: a função directa é computacionalmente tratável mas a função inversa deve ser computacionalmente intratável.

A propriedade criptográficamente significante de um grupo cíclico finito $\langle\, G\,,\,\cdot\,\rangle$ é a existência do **isomorfismo de grupos** $\mathbb{Z}_{|G|} \simeq G$ estabelecida pela exponenciação $\mathsf{exp}_g \,:\, i \mapsto g^i$ e pela sua função inversa $\mathsf{log}_g : G \to \mathbb{Z}_{|G|}$ (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

4 Definição *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 $\beta = \alpha^a \pmod p$.*

Algoritmos triviais para LogDisc(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")

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[a] 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

$$a = a_q \pmod{q} \tag{$\dagger$}$$

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 ($\dagger$) 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

$$(\alpha^a)^r = (\alpha^r)^{a_q} \pmod{p} \qquad \text{ou seja} \qquad \beta^r = (\gamma)^{a_q} \pmod{p} \tag{$\ddagger$}$$

Pode-se ver ($\ddagger$) 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 ($\dagger$) temos $\mu|(a - x_1)$; tal como anteriormente, com $r = (p-1)/\mu$, tem-se $a\,r = r\,x_1 \pmod{p-1}$ o que permite, determinar x_1 resolvendo um PLD num sub-grupo de ordem μ e gerador $\gamma = \alpha^r \pmod{p}$.

Tem-se $\mu^2|(a - x_1 - x_2\,\mu)$ donde $(a - x_1)\,(r/\mu) = r\,x_2 \pmod{p-1}$. Daí determina-se x_2 resolvendo um PLD no mesmo sub-grupo.

E assim sucessivamente determinam-se todos os dígitos $x_1, x_2, \ldots x_e$.

Complexidade: *proporcional ao maior divisor primo de* $(p-1)$

[a] Muitas técnicas criptográficas usam primos da forma $p = q\,2^t + 1$, com q primo.

Método ρ-Pollard

Quando se pretende resolver o problema do logaritmos discreto num subgrupo 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.[a]
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

$$x_k = x_{2k} \pmod{p}$$

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* $\sqrt{q}$.

[a] Por exemplo, $S_i \equiv \{\, x \mid x = i \pmod{3} \,\}$ $\quad i = 0, 1, 2$.

Técnicas Criptográficas– Módulo IV

Criptografia de Chave Pública

(Factorização de Inteiros)

1. Técnicas criptográficas baseadas na factorização de inteiros. Condições de seguranca.
2. Técnicas RSA.
   1. Condições de correcção.
   2. Cifra RSA. Descrição, correcção e ataques.
   3. Esquema de assinaturas com apêndice de mensagem (*aam*) RSA.
3. Técnicas Rabin.
   1. O problema da raiz quadrada.
   2. Cifra Rabin.
   3. Assinatura Rabin: primeira noção de assinatura com redundância.
4. Esquemas de Assinaturas com Recuperação de Mensagem (*arm*). Descrição geral e condições de correcção e segurança.
5. RSA e Rabin como esquemas *arm*.
6. Cifras não-deterministas FI: Goldwasser-Micali e Blum-Goldwasser.

Técnicas criptográficas baseadas na Factorização

As técnicas criptográficas de *chave pública* são sempre baseadas em dois factos: um que garante a satisfação das **condições de correcção** e outro que garante a satisfação das **condições de segurança**.

Neste estudo classificam-se as diversas técnicas (cifras, assinaturas, etc.) pelo facto que determina as condições de segurança.

Segurança das técnicas de factorização

Se p e q são primos suficientemente grandes ($\geq 2^{512}$) e suficientemente distintos (p, q e $(p-q)$ têm todos a mesma ordem de grandeza) então a factorização de $p \times q$ é computacionalmente intratável.

Parâmetros gerais das técnicas de factorização

(1) São escolhidos primos p e q que satisfaçam a condição de segurança e é determinado $m = p \times q$ (designado por **módulo**).

(2) São determinados items de informação k e K, designados por **chave pública** e **chave privada** respectivamente, de tal modo que:

(a) O conhecimento do par $\langle p, q \rangle$ é equivalente ao conhecimento da chave privada K.

$$[p] \wedge [q] \Leftrightarrow [K]$$

(b) O conhecimento da chave pública k implica o conhecimento do módulo m.

$$[k] \Rightarrow [m]$$

Nota O operador $[\cdot]$ denota o **conhecimento da informação**: se o *símbolo* x designa um item de informação, então $[x]$ é a *proposição* que representa a asserção "é conhecido x".

Técnicas RSA

Parâmetros específicos

(1) É determinado $\phi = (p-1) \times (q-1)$ e escolhido $a \in \mathbb{Z}_\phi^*$.

(2) É calculado $b = a^{-1} \pmod{\phi}$ e são destruídos os valores p, q e ϕ.

(3) A chave pública é $k = \langle\, m\,,\, a\,\rangle$ e a chave privada é $K = \langle\, m\,,\, b\,\rangle$.

8 Facto *O conhecimento da factorização de m é equivalente ao conhecimento da chave privada K.*

De p e q reconstrói-se ϕ e, com o conhecimento de a (que é público) calcula-se b. Inversamente, conhecido K pode-se calcular $\mu = (a\,b - 1)$. Sabe-se que $x^{\mu/2} = \pm 1 \pmod{m}$ para cerca de metade dos $x \in \mathbb{Z}_m$; assim, por tentativas, pode-se gerar vários valores de x e calcular $\gcd(x^{\mu/2} - 1, m)$ até encontrar um factor não trivial de m.

Correcção

Se m é um produto de primos distintos então, para todo $x \geq 0$

$$a = b^{-1} \pmod{\phi(m)} \quad \Longrightarrow \quad (x^a)^b = x \pmod{m}$$

Notas

(i) O item a da chave pública é escolhido como um pequeno primo da forma $2^t + 1$ que não divida ϕ. O objectivo é o de facilitar o cálculo de exponenciações x^a em $\mathbb{Z}_m$ que, neste caso, podem ser executadas com t quadrados seguidos de uma multiplicação.

(ii) Conhecidos os items p e q a determinação do valor $y^b \pmod{m}$ por exponenciação directa é menos eficiente que a sua determinação, usando o teorema chinês dos restos, a partir de valores parciais $y_p = y^b \pmod{p}$ e $y_q = y^q \pmod{q}$.

Cifra RSA

A forma da função de cifragem é a mesma da função de decifragem mudando apenas a chave usada: pública na cifragem e privada na decifragem.

cifragem	$\mathsf{rsa}(x)$	$= x^a \pmod{m}$	$\forall x, y \in \mathbb{Z}_m$
decifragem	$\mathsf{rsa}^{-1}(y)$	$= y^b \pmod{m}$	

9 Facto *A cifra RSA verifica:*

(a) a condição de correcção de cifras assimétricas,

(b) a condição de não-invertibilidade de chaves,

(c) a condição de não-invertibilidade do criptograma.

Justificação

Para todo $x \in \mathbb{Z}_m$, $\mathsf{rsa}^{-1}(\mathsf{rsa}(x)) = (x^a)^b \pmod{m} = x$; logo o RSA é correcta.

O conhecimento de K é equivalente ao conhecimento da factorização de m que, por construção, é intratável; logo o RSA é não-invertível nas chaves.

A eventual invertibiliade do criptograma está relacionada com os *ataques circulares* baseados na construção da sequências $y_0 = y,\ y_{k+1} = y_k^a \pmod{m}$.

Se se encontrar $n > 0$ tal que $y_n = y$ então será $y_{n-1} = x$. Se se verificar $m > \gcd(y_n - y, m) > 1$, esse valor será um dos factores p ou q; obtém-se a factoriazação de m e, daí, a chave privada b e depois x.

O problema da invertibilidade do criptograma é essencialmente equivalente ao da factorização de m. Pode no entanto existir, além do conhecimento do a, m e y, informação adicional que dê origem a **ataques** que determinem x.

Ataques ao RSA

Baseados na factorização: se m for factorizável, porque é pequeno, ou porque admite uma factorização de Fermat (p e q são próximos) então a chave pública é invertível.

Circulares: baseados na determinação da sequência $y_k = y^{a^k}$; o ataque inverte y ou determina a factorização de m como foi indicado atrás. Como um ataque desta forma está relacionada com os grupos cíclicos de ordem q e q, o ataque é improvável se $p-1$ e $q-1$ tiverem factores primos grandes.

Propriedades multiplicativas

A cifragem e a decifragem no RSA preservam a estrutura multiplicativa da entrada: $(x_1\, x_2)^u = x_1^u\, x_2^u \pmod{m}$ com $u \equiv a$ ou $u \equiv b$. Esta **redundância** pode ser explorada em ataques.

Falso criptograma: se o intruso convencer o titular de b a decifrar $y' = r^a\, y$ (por exemplo, assinando-o que no RSA que é equivalente à decifragem), com um valor de r por si escolhido, e se o resultado for x' então $x' = (y')^b = (r^{a\,b}\, y^b) = r\, x \pmod{m}$; logo $r^{-1}\, x' \pmod{m}$ recupera x.

Módulo comum: sejam $\langle\, a_1\,,\, m\,\rangle$ e $\langle\, a_2\,,\, m\,\rangle$ duas chaves públicas com módulo comum e sejam $y_1 = x^{a_1} \pmod{m}$, $y_2 = x^{a_2} \pmod{m}$ dois criptogramas para a mesma mensagem. Se for $\gcd(a_1, a_2) = 1$ determinam-se r, s tais que $a_1\, r = 1 + a_2\, s$ e daí calcula-se x

$$(x^{a_1})^r = x^1\, (x^{a_2})^s \quad\Longrightarrow\quad y_1^r = x\, y_2^s \quad\Longrightarrow\quad x = y_1^r\, y_2^{-s} \pmod{m}$$

Expoente comum: se a é um primo pequeno (tipicamente $a = 3$) comum a várias chaves públicas $\langle\, a\,,\, m_i\,\rangle$ e se são conhecidos criptogramas $y_i = x^a \pmod{m_i}$ para a mesma mensagem x, calcula-se $M = m_1\, m_2\, \ldots\, m_a$; porque $x^a < M$ com teorema chinês dos restos determina-se $u < M$ tal que $u = x^a \pmod{M}$; se for possível calcular raízes de ordem a em $\mathbb{Z}_M$ temos x determinado.

Esquema de Assinaturas RSA

É um esquema determinístico onde o cálculo da assinatura é equivalente à decifragem RSA e a verificação é equivalente à cifragem RSA.

Dado o texto x, a assinatura $\mathsf{ass}(x)$ e a verificação $\mathsf{ver}(x,s)$ são

assinatura	$\mathsf{ass}(x) = s$	sse	$s = x^b \pmod{m}$	$\forall x, s \in \mathbb{Z}_m$
verificação	$\mathsf{ver}(x,s)$	sse	$x = s^a \pmod{m}$	

10 Facto (Correcção) $\mathsf{ass}(x) = s$ *sse* $\mathsf{ver}(x,s)$

Justificação Pela condição de correcção do RSA.

11 Facto (Segurança) *São públicos* $\langle\, a \,,\, m \,\rangle$, *o texto* x *e a assinatura* s *tal que* $\mathsf{ver}(x,s)$ *é verdadeiro. Então*

(i) É intratável determinar a chave privada b

(ii) É intratável determinar $x' \neq x$ *tal que* $\mathsf{ver}(x',s)$ *é verdadeiro.*

Justificação (i) o conhecimento simultâneo de a e b é equivalente à factorização de m; (ii) se existisse x' teríamos $\mathsf{ass}(x') = \mathsf{ass}(x)$ e isso não é possível porque a decifragem RSA (equivalente à assinatura) é sempre biunívoca.

A informação adicional que referimos nos ataques às cifras RSA dá origem a ataques análogos às assinaturas.

Técnicas Rabin para criptografia de chave pública

Problema da Raíz Quadrada (PRQ)

Dado $u \in \mathbb{Z}_m$ determinar, caso exista, $a \in \mathbb{Z}_m$ tal que $u = a^2 \pmod{m}$.

12 FACTO *A determinação de raízes quadradas em $\mathbb{Z}_m$ é computacionalmente tratável se m for primo ou se a factorização de m for conhecida.*

Quando a factorização de m é desconhecida o problema da raiz quadrada é equivalente ao problema da factorização.

Justificação Se m é primo a determinação de a é trivial (ver o `Handbook of Applied Criptography`) e existem sempre duas soluções para este problema.

Se $m = p\,q$ determinam-se, pelo processo anterior, $x \in \mathbb{Z}_p$ e $y \in \mathbb{Z}_q$ tais que $x^2 = u \pmod{p}$ e $y^2 = u \pmod{q}$.

Usando o terorema chinês dos restos determinamos um valor $a < p\,q$ que satisfaça $a = x \pmod{p}$ e $a = y \pmod{q}$. As duas soluções para x e as duas soluções para y dão origem a 4 soluções para a.

Cifra Rabin

Parâmetros específicos

A chave pública k é o módulo m e a chave privada K é o par $\langle p\,,\,q\,\rangle$.

cifragem	$\mathsf{rabin}(x)$	$= x^2 \pmod{m}$	$\forall x, u \in \mathbb{Z}_m$
decifragem	$\mathsf{rabin}^{-1}(u)$	$= \{a_1, a_2, a_3, a_4\}$	soluções do **PRQ**

O valor do texto x é uma das 4 soluções a_i do PRQ; o titular de K tem de usar um critério adicional para decidir qual dos a_i coincide com o texto original.

Segurança e Ataques a Rabin

A segurança da cifra Rabin é baseada na equivalência entre o PRQ e o problema da factorização de m.

Porque a cifragem Rabin é essencialmente análoga ao RSA todos os ataques RSA são convertíveis em ataques Rabin.

Assinatura Rabin

Parâmetros específicos

(i) Uma função $r : \mathcal{M} \to \mathcal{Q}_m$ invertível, computacionalmente tratável (assim como a sua inversa), do espaço de mensagens $\mathcal{M}$ para o espaço $\mathcal{Q}_m$ dos resíduos quadráticos em $\mathbb{Z}_m$; i.e., os elementos de $\mathbb{Z}_m$ que têm raiz quadrada.

(ii) Um teste $\mathsf{teste} : \mathbb{Z}_m \to \mathbb{B}$ computacionalmente tratável que verifica

$$\mathsf{teste}(x) = \mathsf{verd} \qquad \text{sse} \qquad x \in r(\mathcal{M})$$

Condição de segurança

A *função de redundância* r é escolhida de modo a ser computacionalmente intratável encontrar $s \in \mathbb{Z}_m$ tal que $s^2 \pmod m$ satisfaça $\mathsf{teste}(\cdot)$.

Esquema de Assinatura

Dado um texto $u \in \mathcal{M}$ a assinatura e verificação são dadas por

assinatura	$\mathsf{ass}(u) = s$	sse	$s^2 = r(u)$
verificação	$\mathsf{ver}(s)$	sse	$\mathsf{teste}(s^2 \pmod m)$

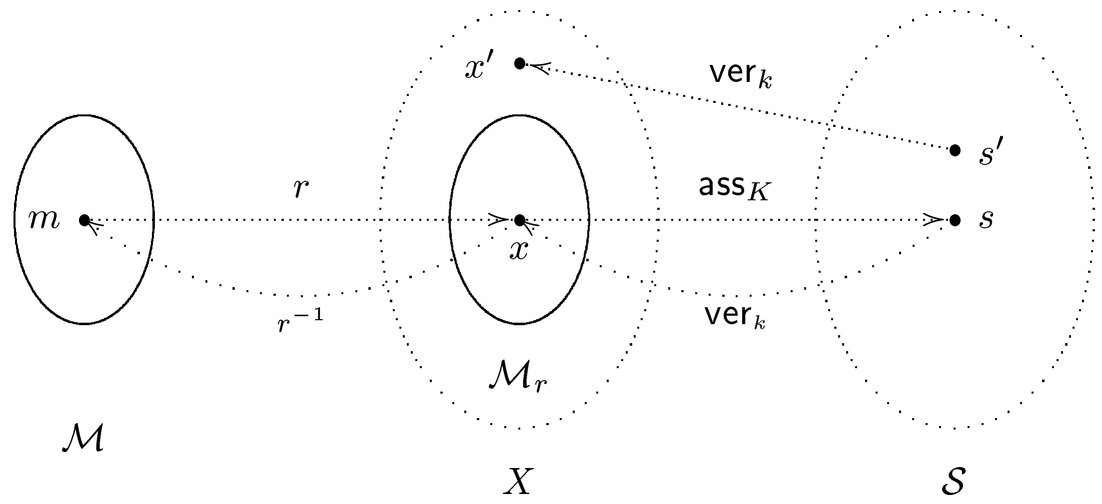
A função de assinatura usa a factorização de m (a chave privada) para calcular e seleccionar uma das quatro raízes quadradas de $r(u)$.

Assinatura com Recuperação de Mensagem (*arm*)

Recentemente apareceram novas técnicas de assinatura que, no acto de verificação, **não** requerem a presença explícita do texto x; chamam-se **assinaturas com recuperação de mensagens**.

Nas assinaturas tradicionais (que requerem x na verificação) chamadas **assinaturas com apêndice de mensagem (*aam*)** tem-se $|\mathcal{M}| \gg |X|$; o seu uso em mensagens genéricas requer um *hash*, $\mathsf{h} : \mathcal{M} \to X$.

Numa *arm* tem-se a situação inversa: o espaço de códigos é muito maior do que o espaço de mensagens e esse aumento de dimensão é representado por uma **função de redundância** $r : \mathcal{M} \to X$ que é <u>injectiva</u> e <u>pública</u>.



$\mathcal{M}$	mensagens	$\mathcal{M}_r \equiv r(\mathcal{M})$	imagem de $\mathcal{M}$
X	códigos	$\mathcal{S}$	assinaturas
r	função de redundância	ass, ver	assinatura, verificação

Esquema geral de uma ARM determinista

Função	**Aridade**	**Propriedades**
redundância	$r : \mathcal{M} \to X$	injectiva, pública e computacionalmente tratável assim como a sua inversa $r^{-1} : \mathcal{M}_r \to \mathcal{M}$
assinatura	$\mathsf{ass}_K : X \to \mathcal{S}$	injectiva e privada (dependente da chave privada K)
verificação	$\mathsf{ver}_k : \mathcal{S} \to X$	inversa de ass mas pública (dependente da chave pública k)
aceitação	$(\cdot \in \mathcal{M}_r) : X \to \mathbb{B}$	computacionalmente tratável

Condição de Correcção
Para todo $x \in X$ e todo $s \in \mathcal{S}$

$$s = \mathsf{ass}_K(x) \quad \text{sse} \quad x = \mathsf{ver}_k(s)$$

Condição de segurança
Dada uma chave pública k, é computacionalmente intratável, sem conhecer a respectiva chave privada K, achar um s tal que $\mathsf{ver}_k(s) \in \mathcal{M}_r$.

Esquema de Verificação
A assinatura $s \in \mathcal{S}$ é **aceite** se e só se

$$\mathsf{ver}_k(s) \in \mathcal{M}_r$$

Esquema de recuperação
Se s é uma assinatura aceite, a mensagem inicial é recuperada por

$$r^{-1}(\mathsf{ver}_k(s))$$

ARM em RSA e Rabin

O esquema de assinaturas Rabin é, à partida, um esquema com recuperação de mensagens.

As cifras assimétricas deterministas, como o RSA, adaptam-se eventualmente a um esquema de assinaturas com recuperação de mensagem:

(i) Escolhendo ass_K como a função de decifragem e ver_k como função de verificação garante-se a correcção da *arm*.

(ii) Escolhendo uma função de redundância apropriada será eventualmente possível satisfazer a condição de segurança determinada pela função ver_k escolhida em (i).

Nas técnicas baseadas em factorização (RSA e Rabin) é possível escolher uma tal função de redundância e existe uma norma (ISO/IEC 9796) que estabelece essa função.

As mensagens em $\mathcal{M}$ são, tipicamente, resultado de uma função de *hash* e, assim, são representáveis com 160 bits; donde $|\mathcal{M}| \simeq 2^{160}$.

Em contrapartida o espaço de códigos $X \equiv \mathbb{Z}_m$ tem m elementos. Porque as condições de segurança das técnicas impõe pelo menos 1024 bits de representação para m, teremos $|X| \simeq 2^{1024}$.

Dada a desproporção entre as cardinalidades de $\mathcal{M}$ e X é relativamente simples construir uma função de redundância $r : \mathcal{M} \rightarrow X$ que verifique a condição de segurança.

A probabilidade de uma assinatura "falsa" s' ser aceite é dada pela probabilidade de $\mathsf{ver}_k(s')$ ir cair dentro $\mathcal{M}_r$. Como $|\mathcal{M}_r|/|X| \simeq 2^{-864}$ essa probabilidade é desprezável.

Cifras Não Deterministas

Cifras não deterministas produzem, para a mesma chave e o mesmo texto, criptogramas distintos em cada invocação. Evitam ataques por repetição, ataques baseados na estrutura do texto e evitam canais sublimares.

CIFRA GOLDWASSER-MICALI

Parâmetros específicos

(i) Selecciona-se a que **não** seja um resíduo quadrático módulo m mas cujo símbolo de Jacobi $\left(\frac{a}{m}\right)$ seja 1; i.e. a é um pseudo resíduo quadrático.

(ii) A chave pública k é o par $\langle a, m \rangle$ e a chave privada K é a factorização $\langle p, q \rangle$ de m.

Esquema de cifragem

(i) É escolhido uma "chave de sessão" aleatória $\omega \in \mathbb{Z}_m^*$.

(ii) Representa-se o texto x pela sua representação binária $b_1 b_2 \cdots b_n$, com $b_i \in \mathbb{Z}_2$.

(iii) Para $i = 1, \cdots, n$ determina-se $y_i = a^{b_i} \omega^2 \pmod{m}$.

(iv) O criptograma é o tuplo $\langle y_1, y_2, \cdots, y_n \rangle \in \mathbb{Z}_m^n$.

Esquema de decifragem

(i) Para cada y_i do criptograma calcula-se $s_i = \left(\frac{y_i}{p}\right)$.

(ii) Será $b_i = 1$ se e só se $s_i \neq 1$.

Correcção da cifra

Se $b_i = 0$ então $y_i = \omega^2 \pmod{m}$ é um resíduo quadrático módulo m e também o é módulo p; logo $\left(\frac{y_i}{p}\right) = 1$.

Se $b_i = 1$ então $y_i = a\,\omega^2 \pmod{m}$ não é um resíduo quadrático módulo m (porque a não o é por construção) e portanto não é resíduo quadrático módulo p; teremos, porque p é primo, $\left(\frac{y_i}{p}\right) \neq 1$.

Segurança da cifra

Cada elemento y_i verifica sempre $\left(\frac{y_i}{m}\right) = 1$ porque é o produto de duas parcelas a^{b_i} e ω^2 que verifica, ambas esta propriedade. O intruso que conheça apenas y_i sem conhecer a factorização de m não tem possibilidade de distinguir se se trata de um resíduo quadrático ou de um peseudo-resíduo quadrático.
Se esta distinção for um problema computacionalmente intratável (como é com a tecnologia actual) a cifra não é invertível sem conhecimento dos factores de m.

Notas

1. Uma desvantagem óbvia deste esquema é a expansão de n bits do texto para $n \times \ln(m)$ bits do criptograma. Um texto de 1 kbyte, com um módulo de 1024 bits, origina um criptograma com 1 Mbyte !!.

2. A chave pública a é construída da seguinte forma: geram-se não resíduos quadráticos c módulo p e d módulo q e usa-se o teorema chinês dos restos para resolver o sistema de equações $\{a = c \pmod{p}\,,\; a = d \pmod{q}\}$.
   Como $\left(\frac{a}{p}\right) = \left(\frac{a}{q}\right) = -1$ temos $\left(\frac{a}{p\,q}\right) = (-1)\,(-1) = 1$.

Cifra Blum-Goldwasser

Cifra não-determinista baseada na factorização que, em termos de eficiência computacional e expansão do criptograma é equivalente ao RSA.

Baseia-se no uso do gerado Blum-Blum-Shub para gerar uma sequência de números pseudo-aleatórios que funcionam como uma chave "*one-time pad*" para o texto. O criptograma é formado pela sequência de bits assim cifrada juntamente com a cifragem da semente do gerador BBS.

Parâmetros específicos

(i) Os primos p e q verificam $p = q = 3 \pmod 4$.

(ii) A chave pública k é o módulo m e a chave privada K é a sua factorização $\langle\, p\,,\, q\,\rangle$.

(iii) Escolhe-se n tal que $m \leq 2^{2^n}$; i.e. cada $x \in \mathbb{Z}_m$ pode ser representado com 2^n bits.

Esquema de cifragem

1. Representa-se o texto a cifrar x como uma *string* de dígitos $x_1\, x_2\, \ldots\, x_k$ com $x_i \in \mathbb{Z}_{2^n}$.
2. Escolhe-se $\omega \in \mathbb{Z}_m^*$ aleatório e determina-se a semente do gerador BBS, como $s_0 = \omega^2 \pmod m$.
3. Geram-se k números pseudo-aleatórios $u_1\, u_2\, \cdots\, u_k$: para $i = 1, \cdots k$
$$s_i = s_{i-1}^2 \pmod m \qquad\qquad u_i = s_i \pmod{2^n}$$
4. O criptograma é um par $\langle\, y\,,\, s_k^2 \pmod m\,\rangle$ em que $y = \langle y_1, \cdots, y_k\rangle$ é o tuplo de componentes $y_i = x_i \oplus u_i$.

Nota

Recordemos que, verificando-se (i), a equação $a^2 = s \pmod p$ tem solução $a = s^{(p+1)/4} \pmod p$.

Este algoritmo pode ser generalizado para soluções da equação $a^{2^t} = s \pmod p$ calculando um expoente $d = (\frac{p+1}{4})^t \pmod{p-1}$ e a solução $a = s^d \pmod p$.

Para calcular a raiz-2^t módulo m de s usa-se o teorema chinês dos restos para determinar esse valor a partir das raízes-2^t módulo p e módulo q, calculadas, por seu turno, pelo algoritmo que acabámos de sugerir.

Esquema de decifragem

1. Dado o criptograma $\langle y_1, y_2, \cdots, y_k, \mu \rangle$, dado que $\mu = s_0^{2^{k+1}} \pmod m$, usa-se o algoritmo anterior para determinar s_0 como raiz-2^{k+1} de μ.
2. Com a semente s_o usa-se o gerador BBS para produzir a sequência $u_1\, u_2\, \cdots u_k$ tal como na cifragem.
3. Determinam-se os dígitos x_i por $x_i = y_i \oplus u_i$

Comentários

(i) Uma vez geradas as "chaves" u_i esta cifra comporta-se como um *one-time pad* e, portanto a sua correcção é óbvia.

(ii) A segurança desta cifra é idêntica à Rabin e está baseada na complexidade da extracção de raízes quadradas módulo m sem ser conhecida a factorização de m. Neste caso a segurança reside na impossibilidade de um intruso conhecer a semente s_0 sem conhecer a factorização de m.

(iii) O criptograma $\langle\, y\, ,\, \mu\, \rangle$ duplica o tamanho do texto claro x. Isto é usual em cifras não deterministas e é o preço a pagar pelas vantagens que esta modalidade de cifragem introduz. De qualquer forma esta expansão é muito inferior à que ocorre no esquema Goldwasser-Micali.

(iv) Comparando o números de operações aritméticas "pesadas" (multiplicações e exponenciações) do RSA e do Blum-Goldwasser conclui-se que a complexidade computacional destas técnicas é semelhante.

Técnicas Criptográficas– Módulo V

Criptografia de Chave Pública

(Logaritmo Discreto)

1. Técnicas criptográficas baseadas no problema LogDisc(p).
2. Protocolo Diffie-Hellman para acordo de chaves.
3. Cifra El Gamal. Ataques à cifra El Gamal.
4. Esquema de assinaturas El Gamal. Ataques.
5. Esquemas de assinaturas com recuperação de mensagem Nyberg-Rueppel.
6. Esquemas de assinaturas DSA (*Digital Signature Standard*).
7. Assinaturas não-repudiáveis de Chaum van-Antwerpen
8. Assinaturas "fail-then-stop"

Técnicas baseadas no problema LogDisc(p)

Um número considerável de técnicas criptográficas é baseada nas propriedades dos grupos cíclicos: a condição de correcção baseia-se no isomorfismo $\mathbb{Z}_{|G|} \simeq G$ estabelecido pela "*exponenciação*" $a \mapsto g^a$, e a segurança baseia-se na intratabilidade computacional da função inversa, o "*logaritmo*".

Nomeadamente são essenciais os grupos cíclicos da forma $\mathbb{Z}_p^*$, sendo p primo.

Parâmetros gerais da Criptografia de Chave Publica em $\mathbb{Z}_p^*$.

1. É dado um primo p ímpar grande ($p \gtrsim 2^{512}$) tal que o maior primo q que divide $(p-1)$ é ainda grande ($q \gtrsim 2^{160}$).
2. É escolhido um pequeno primo aleatório α que seja gerador de $\mathbb{Z}_p^*$.
3. São geradas uma chave pública k e uma chave privada K tais que:
   (i) O conhecimento de k implica o conhecimento de p e α.

13 Facto (Correcção) *A função $x \mapsto \alpha^x \pmod p$ determina um isomorfismo de grupos entre o grupo aditivo $\mathbb{Z}_{p-1}$ e o grupo multiplicativo $\mathbb{Z}_p^*$.*

14 Facto (Segurança) *Dado um $y \neq 1$ em $\mathbb{Z}_p^*$ arbitrário, é computacionalmente intratável determinar o $x \in \mathbb{Z}_{p-1}$ tal que $y = \alpha^x \pmod p$.*

Protocolo de Diffie-Hellman para Acordo de Chaves

Objectivo Dois agentes A e B pretendem acordar num segredo comum κ (uma chave) trocando informação num canal aberto.

$$\begin{tikzcd} A \arrow[r, shift left, "\beta=\alpha^a"] & B \arrow[l, shift left, "\gamma=\alpha^b"] \\ a & b \end{tikzcd}$$

$$\gamma^a = \beta^b = \alpha^{a\,b}$$

(i) A gera um $a \in \mathbb{Z}_{p-1}$ aleatório, calcula $\beta = \alpha^a \pmod{p}$ e envia-o para B.

(ii) B gera um $b \in \mathbb{Z}_{p-1}$ aleatório, calcula $\gamma = \alpha^b \pmod{p}$ e envia-o para A.

(iii) A calcula $\kappa = \gamma^a \pmod{p}$

(iv) B calcula $\kappa = \beta^b \pmod{p}$

Segurança

1. Um intruso I, que intersecte β e γ, não consegue descobrir a ou b (pela complexidade do LogDisc(p)) e portanto não consegue calcular κ pelo método usado por A ou pelo método usado por B.

2. **Porém** se conseguir evitar que β chegue a B e que γ chegue a A pode montar um ataque (conhecido por **homem-no-meio**):

$$\begin{tikzcd} A \arrow[r, "\beta=\alpha^a"] & \times \arrow[d, shift right] \arrow[d, shift left] & B \arrow[l, "\gamma=\alpha^b"'] \\ & I \arrow[ul, "\gamma'=\alpha^c"] \arrow[ur, "\beta'=\alpha^c"'] & \end{tikzcd}$$

– todas as operações feitas $\pmod{p}$ –

(i) I gera c e envia um falso β para B e um falso γ para A ambos iguais a α^c.

(ii) I acorda um falso κ com A igual a $\alpha^{a\,c}$ e acorda um falso κ com B igual a $\alpha^{b\,c}$

Assim A julga ter acordado uma chave com B e realmente acordou-a com I; o mesmo se passa com B: o intruso I intersecta toda a comunicação entre A e B.

Conclusão Em qualquer acordo de chaves os intervenientes devem usar um protocolo de identificação para se assegurarem da identidade mútua.

PRINCÍPIO DAS CIFRAS NÃO DETERMINISTAS NUM GRUPO ABELIANO G

O texto $x \in G$ é cifrado numa aplicação da operação de grupo $(\kappa \cdot x)$, sendo κ uma *chave de sessão* aleatória gerada a partir da chave pública.

Juntamente com $(\kappa{\cdot}x)$, o criptograma contém um *indicador de redundância* γ que permite reconstruir κ a partir da chave privada.

Cifra El Gamal

O grupo abeliano é aqui o grupo multiplicativo $\mathbb{Z}_p^*$.

Parâmetros específicos

1. É gerado $a \neq 0$, aleatório em $\mathbb{Z}_{p-1}$, e calculado $\beta = \alpha^a \pmod p$.
2. A chave pública é $k \equiv \langle p, \alpha, \beta \rangle$ e a chave privada é $K \equiv a$

Esquema de Cifragem

(i) O emissor gera $\omega \neq 0$ aleatório em $\mathbb{Z}_{p-1}$ e calcula $\gamma = \alpha^\omega \pmod p$

(ii) O emissor determina a chave de sessão $\kappa = \beta^\omega \pmod p$ e o valor $y = (\kappa\, x) \pmod p$.

(iii) O criptograma é o par $\langle\, y\,,\, \gamma\,\rangle$.

Esquema de decifragem

(i) O receptor recebe $\langle\, y\,,\, \gamma\,\rangle$ e recupera a chave de sessão κ como $\gamma^a \pmod p$

(ii) O receptor determina x como $(\kappa^{-1}\, y) \pmod p$.

Correcção A chave de sessão β^ω calculada pelo emissor coincide com a calculada pelo receptor, γ^a, porque, atendendo às definições de β e γ, ambos os valores são iguais a $\alpha^{a\,\omega}$.

Segurança O conhecimento de γ não permite determinar ω e o conhecimento de β não permite determinar a. Ambos os processos de calcular a chave de sessão estão interditos a um intruso.

Ataques à cifra El Gammal

Substituição da chave pública

A cifra El Gammal estabelece um protocolo Diffie-Hellman para acordo da chave κ tendo em atenção que os intervenientes já se assumem como mutuamente identificados. Aqui a substituição fraudulenta da chave pública equivale ao ataque *homem-no-meio*; não é um problema específico desta cifra e a sua resolução está na **certificação de chaves públicas**.

Ataque ao logaritmo discreto

Se o intruso fosse capaz de resolver o problema do logaritmo discreto invertia a chave e o criptograma. De facto, conhecendo p, α e β, determinaria a (a chave privada) e, conhecendo γ, determinaria ω, depois κ e depois x.

Ataque por ω comum

Se o mesmo ω for usado em dois textos diferentes, x_1 e x_2, a chave de sessão κ (que só depende de ω) é a mesma em ambas criptogramas, $\langle\, y_1\,,\, \gamma_1 \,\rangle$ e $\langle\, y_2\,,\, \gamma_2 \,\rangle$. Teremos $y_1 = \kappa\, x_1$, $y_2 = \kappa\, x_2$ e $\gamma_1 = \gamma_2$.

Se adicionalmente um dos x_i for conhecido, o outro é determinado bem como a chave κ.

Ataque por canal subliminar

Um implementador malicioso do esquema de cifragem pode aproveitar a redundância introduzida pelo não-determinismo para fazer passar informação privada do emissário (por exemplo, a sua chave privada).

Por exemplo, pode escolher um primo grande $r < q$ que funciona como chave do canal; para enviar um bit de informação b_i, na geração de ω faz tentativas suficientes até obter um γ tal que $\left(\frac{\gamma}{r}\right) = 1$, caso seja $b_i = 1$, ou $\left(\frac{\gamma}{r}\right) \neq 1$, caso seja $b_i = 0$.

Lendo γ na saída, conhecendo r e calculando $\left(\frac{\gamma}{r}\right)$ o intruso conhece b_i. Bit a bit pode, assim, ser enviada a chave toda.

Assinatura Digital El Gamal

Parâmetros específicos

Os mesmos da cifra: a chave pública é $\langle p, \alpha, \beta \rangle$ e a privada é a.

Esquema de geração da assinatura

Dado um texto a assinar $x \in \mathbb{Z}_{p-1}$

(i) O signatário gera $\omega \in \mathbb{Z}^*_{p-1}$ aleatório e calcula $\gamma = \alpha^\omega \pmod{p}$.

(ii) A assinatura é o par $\langle\, s \,,\, \gamma \,\rangle$ com $s \in \mathbb{Z}_{p-1}$ solução de

$$\omega\, s + a\, \gamma = x \pmod{p-1} \qquad (\dagger)$$

Esquema de verificação da assinatura

Dado uma assinatura $\langle\, s \,,\, \gamma \,\rangle$ e um texto x, aceitar a assinatura sse

$$\gamma^s \cdot \beta^\gamma = \alpha^x \pmod{p} \qquad (\ddagger)$$

Correcção

O isomorfismo, dado pela exponenciação, entre os grupos $\mathbb{Z}_{p-1}$ e $\mathbb{Z}^*_p$ produz

$$\begin{aligned} \omega\, s + a\, \gamma = x \pmod{p-1} \quad &\Leftrightarrow \quad (\alpha^\omega)^s \cdot (\alpha^a)^\gamma = \alpha^x \pmod{p} \\ &\equiv \quad \gamma^s\, \beta^\gamma = \alpha^x \pmod{p} \end{aligned}$$

Segurança

A complexidade de $\textsc{LogDisc}(p)$ não permite determinar a a partir de β ou ω a partir de γ.

Encontrar um $x' \neq x$, em $\mathbb{Z}_{p-1}$ e com a mesma assinatura $\langle\, s \,,\, \gamma \,\rangle$, é impossível porque tem de verificar ($\ddagger$) e portanto $\alpha^x = \alpha^{x'} \pmod{p}$. O isomorfismo definido pela exponenciação não o permite.

Dado x, determinar a sua assinatura $\langle\, s \,,\, \gamma \,\rangle$ sem conhecer a, é resolver ($\ddagger$) em ordem a s e γ. A complexidade de $\textsc{LogDisc}(p)$ não o permite.

Porém ...

Dado que x, γ e s são conhecidos, se um intruso conhecer um dos valores a ou ω (através de um ataque) a equação ($\dagger$) permite-lhe determinar o outro.

Ataques à assinatura El Gamal

Ataque por ω comum

Duas assinaturas, $\langle s_1 , \gamma \rangle$ e $\langle s_2 , \gamma \rangle$, com o mesmo valor de ω, produzidas para dois textos x_1 e x_2 verificam (em $\mathbb{Z}_{p-1}$) $\omega\, s_1 + a\,\gamma = x_1$ e $\omega\, s_2 + a\,\gamma = x_2$. Ou seja $\omega\,(s_1 - s_2) = (x_1 - x_2) \pmod{p-1}$.

Se $d = \gcd(s_1 - s_2, p-1) = 1$ então determina-se trivialmente $\omega = (s_1 - s_2)^{-1}(x_1 - x_2) \pmod{p-1}$. Mesmo que seja $d > 1$ existe um processo um pouco mais complicado (ver `Stinson`) para calcular ω.

Ataques por geração simultânea de texto e assinatura

(**1**) Escolhe-se um par $\mu, \upsilon \in \mathbb{Z}_{p-1}$ com $\gcd(\upsilon, p-1) = 1$. Calcula-se $\gamma = \alpha^{\mu}\,\beta^{\upsilon} \pmod{p}$, $s = -\gamma\,\upsilon^{-1} \pmod{p-1}$ e $x = s\,\mu \pmod{p-1}$. Verifica-se facilmente que $\langle s , \gamma \rangle$ é assinatura aceite de x.

(**2**) Se fosse admissível $\gamma > p$, um intruso que conheça uma assinatura $\langle s , \gamma \rangle$ de um texto x pode determinar, para texto qualquer $x' = \upsilon\, x$, valores γ' e s' tais que:

$$\gamma' = \gamma \pmod{p} \qquad \gamma' = \gamma\,\upsilon \pmod{p-1} \qquad s' = s\,\upsilon \pmod{p-1}$$

Verifica-se que $\langle s' , \gamma' \rangle$ é assinatura aceite de x'.

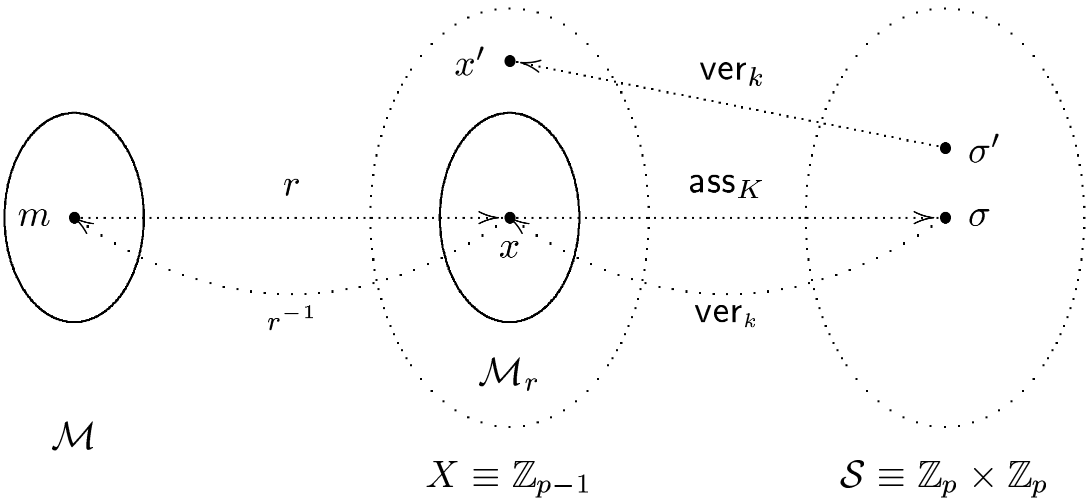
Isto significa que a apresentação de um texto e de uma assinatura não é suficiente para identificar o titular da assinatura. A exigência de o texto passar por uma função de *hash* resolve imediatamente 1ª questão; a exigência de que seja sempre $\gamma \in \mathbb{Z}_{p-1}$ resolve a 2ª questão.

Ataque por canal subliminar

Tal como na cifra é possível, a um implementador malicioso, definir um canal subliminar que faz passar bits de informação para cada invocação da assinatura. Para isso escolhe um primo $r < p$ ímpar e grande e gera falsos pseudo-aleatórios ω até que $\left(\frac{\gamma}{r}\right)$ represente o bit desejado.

Esquema de assinaturas Nyberg-Rueppel

É um esquema com recuperação de mensagens variante do El Gamal.



Parâmetros específicos
Os mesmos do El Gamal. A chave pública é $k \equiv \langle p, \alpha, \beta \rangle$ e a chave privada é $K \equiv a$ tal que $\beta = \alpha^a \pmod{p}$.

Uma escolha segura da função de redundância $r : \mathcal{M} \to \mathbb{Z}_{p-1}$.

Geração da assinatura Dado $x \in \mathbb{Z}_{p-1}$

(i) É gerado $\omega \in \mathbb{Z}^*_{p-1}$ aleatório e calculado $\gamma = \alpha^{-\omega} \pmod{p}$.

(ii) Determina-se $y = x\,\gamma \pmod{p}$ e $s = a\,y + \omega \pmod{p-1}$.

(iii) A assinatura é $\mathsf{ass}_K(x) = \langle\, s\,,\, y\,\rangle$.

Verificação da assinatura Dado $\sigma = \langle\, s\,,\, y\,\rangle \in \mathbb{Z}_p \times \mathbb{Z}_p$

(i) Calcula-se $v = \alpha^s\,\beta^{-y} \pmod{p}$ e $\mathsf{ver}_k(\sigma) = v\,y \pmod{p}$

Correcção Temos $s - a\,y = \omega$ em $\mathbb{Z}_{p-1}$ sse $\alpha^s\,(\alpha^a)^{-y} = \alpha^\omega$ em $\mathbb{Z}^*_p$. Donde $v = \gamma^{-1} \pmod{p}$; como $y = \gamma\,x$ teremos $v\,y = x$.

"Digital Signature Algorithm" (DSA)

Em 1994 foi adoptado como norma FIPS[a] para esquema de assinaturas. Juntamente com uma função de *hash* designada por SHA [b], actualmente numa versão designada por SHA-1, e com outras funções auxiliares (como o gerador de números aleatórios) constitui o **Digital Signature Standard** (DSS) descrito globalmente na norma FIPS 186.

Contexto

É desejável que a geração de assinaturas seja efectuada com muito poucos recursos computacionais (*smart cards*, *buttons*, etc.). A verificação da assinatura não tem estas limitações e pode ser feita em qualquer máquina.

A segurança do esquema de assinaturas El Gamal exige um primo p de ordem $\simeq 2^{1024}$; o espaço de assinaturas é $\mathbb{Z}_p \times \mathbb{Z}_p$ (o que exige 2048 bits de representação) e as operações "pesadas" (multiplicações, exponenciações, módulos, ...) são efectuadas em $\mathbb{Z}_p^*$ e exigem, por isso, aritmética de 1024 bits.

Estes requisitos tornam inviável uma implementação da geração de assinaturas El Gamal nos dispositivos referidos já que este dispõe, tipicamente, de 1 ou 2 kylobytes de memória RAM.

Pretende-se um esquema de assinaturas tão seguro quanto o El Gamal (use o mesmo p) mas onde a geração de assinaturas exija aritmética num espaço $\mathbb{Z}_q$, com q substancialmente menor do que p.

O DSA usa $q \simeq 2^{160}$, a aritmética necessária à geração da assinatura é realizada em $\mathbb{Z}_q$ e a assinatura é um elemento de $\mathbb{Z}_q \times \mathbb{Z}_q$. Portanto a assinatura requer aritmética de 160 bits e ocupa apenas 320 bits. [c]

[a] Federal Information Processing Standard
[b] Secure Hash Algorithm
[c] Em oposição aos 2048 bits que a assinatura ElGamal necessita.

DSA (cont.)

Parâmetros Específicos

(i) q é um primo representável com 160 bits e $p = 2^i\, q + 1$ é um primo representável com $64 \times t$ bits, $t = 8, 9, ...16$.

(ii) α é um gerador do sub-grupo cíclico de ordem q em $\mathbb{Z}_p^*$. [a]

(iii) a é aleatório em $\mathbb{Z}_q^*$ e $\beta = \alpha^a \pmod{p}$.

(iv) $k \equiv \langle p, q, \alpha, \beta \rangle$ é a chave pública e $K \equiv a$ é a chave privada.

Esquema de Geração da Assinatura

Determinar $\mathsf{ass}_K(x)$ dado um texto $x \in \mathbb{Z}_q$.

(i) ω é aleatório em $\mathbb{Z}_q^*$ e $\gamma = (\alpha^\omega \mod p) \pmod{q}$

(ii) s é solução de $s\,\omega = x + a\,\gamma \pmod{q}$.

(iii) $\mathsf{ass}_K(x)$ é definido pelo par $\langle\, s\,,\, \gamma \,\rangle$.

Esquema de Verificação da assinatura

Dado $x \in \mathbb{Z}_q$ e $\sigma = \langle\, s\,,\, \gamma \,\rangle \in \mathbb{Z}_q \times \mathbb{Z}_q$, verificar $\mathsf{ver}_k(x, \sigma)$

(i) Determinar $\mu = x\, s^{-1} \pmod{q}$ e $\upsilon = \gamma\, s^{-1} \pmod{q}$.

(ii) Determinar $\gamma' = \alpha^\mu\, \beta^\upsilon \pmod{p}$.

(iii) Calcular $\mathsf{ver}_k(x, \sigma) \;\equiv\; (\gamma' = \gamma \mod q)$

[a] Tipicamente calculado como $g^{-2^i} \pmod{p}$ sendo g um gerador de $\mathbb{Z}_p^*$

Correcção

Pela definição de s, μ e v tem-se $\omega = \mu + a\,v \pmod{q}$.
Donde $\alpha^\omega = \alpha^\mu\,(\alpha^a)^v \pmod{p}$. Atendendo à definição de γ conclui-se $\gamma = (\alpha^\mu\,\beta^v \mod p) \pmod{q}$

Segurança

A complexidade de LogDisc(p) impede a inversão da chave pública e a obtenção de ω conhecido γ.

Impede também que se determine μ ou v, conhecido *sigma*, impedindo assim a inversão da assinatura.

Notas

1. Toda a geração da assinatura $\langle\, s\,,\,\gamma\,\rangle$ é feita em $\mathbb{Z}_q$: necessita aritmética de apenas 160 bits e a própria assinatura ocupa só 320 bits.
   A verificação da assinatura recorre a exponenciações em $\mathbb{Z}_p$: necessita de 1024 ou mais bits.
   Para a utilização "geração $\Rightarrow$ *smart-card*" e "verificação $\Rightarrow$ *host*", esta situação é ideal. Em aplicações onde a verificação é feita em *smart-cards* o DSA é, claramente, não aconselhável.

2. A versão inicial do DSA requeria p representável com 512 bits (correspondente a $t = 8$ na definição); posteriormente foram aceites maiores primos dentro do *standard* da forma como são apresentados na definição.

3. Os ataques ao DSA são semelhantes aos do El Gamal. Por isso nunca o mesmo ω pode ser usado para assinar dois textos nem a assinatura deve ser usada sem uma função de *hash*

Variante Nyberg-Rueppel do DSA

(assinaturas com recuperação de mensagem)

Esquema em tudo idêntico ao Nyberg-Rueppel para El Gamal substituindo todas as operações em $\mathbb{Z}_{p-1}$ pelas mesmas operações em $\mathbb{Z}_q$.

Assinatura não-repudiável de Chaum van-Antwerpen

Objectivo

Protocolo *desafio-resposta* de assinaturas no qual o acto de verificação requer a vontade combinada do signatário e do destinatário.

Exemplo de aplicações

(a) A quer ter acesso a uma área controlada por B (ex. A é um cliente de um banco B e quer ter acesso ao seu cofre); para isso B exige uma assinatura não repudiável de A sobre um documento que estipula data e hora de realização do acto. Após o que:

- (i) B não pode provar a terceiros a realização do acesso sem a colaboração de A.
- (ii) A não pode negar, perante terceiros, ter realizado o acesso.

(b) A cria um programa de software, assina-o com assinaturas não-repudiáveis e vende-o a B que, por seu lado, decide fazer uma cópia e revendê-la a C.

- (i) C não tem possibilidade de verificar a autoria de A pela sua assinatura sem a colaboração de A; a fraude de B é detectada.
- (ii) Na eventualidade de danos A não pode negar ser o autor do programa.

Parâmetros específicos

(i) q é um grande primo tal que $p = 2\,q + 1$ é primo.

(ii) $\alpha \in \mathbb{Z}_p^*$ é um elemento de ordem q. [a]

(iii) $a \in \mathbb{Z}_q$ é aleatório e $\beta = \alpha^a \pmod{p}$.

(iv) $k \equiv \langle p, \alpha, \beta \rangle$ é a chave pública e $K \equiv a$ é a chave privada.

[a] α é um qualquer valor da forma $y^2 \pmod{p}$, com $y \in \mathbb{Z}_p^*$, desde que $\neq 1$. O sub-grupo multiplicativo de ordem q gerado por α é formado por todos os resíduos quadráticos de $\mathbb{Z}_p^*$.

Função de assinatura

Dado o texto $x \in \mathbb{Z}_q$ a assinatura é definida por $\mathsf{ass}_K(x) = x^a \pmod{p}$.

Protocolo de verificação

A é o signatário da assinatura $s = \mathsf{ass}_K(x)$ do texto x ; B é o destinatário. Todas as operações são efectuadas $\pmod{p}$.

$$A \xleftarrow{\;y = s^\mu\,\beta^\upsilon\;} B \qquad \mu, \upsilon$$

B gera números aleatórios $\mu, \upsilon \in \mathbb{Z}_q$, calcula $y = s^\mu\,\beta^\upsilon$ e envia-o a A; y constitui o *desafio*.

$$\underset{b = a^{-1} \pmod{q}}{A} \xrightarrow{\;z = y^b\;} \underset{z = x^\mu\,\alpha^\upsilon}{B}$$

A conhece b inverso de a em $\mathbb{Z}_q$, calcula $z = y^b$ e envia-o a B; z é a *resposta*.

B aceita a assinatura sse $z = x^\mu\,\alpha^\upsilon$.

Justificação

Verifica-se que $y = (x^\mu\,\alpha^\upsilon)^a$; donde $z = y^b = (x^\mu\,\alpha^\upsilon)^{a\,b} = (x^\mu\,\alpha^\upsilon)^1$.

A segurança é garantida pela complexidade de LogDisc(p): é impossível determinar a a partir do conhecimento de x e s, como é impossível determinar μ e υ a partir do conhecimento de y e determinar b a partir do conhecimento de z.

Possibilidades de repúdio por A

Assumindo que A participa no protocolo de verificação, A pode **repudiar** a assinatura numa das seguintes situações:

(i) A assinatura é válida, o protocolo falha e A participa no protocolo fraudulentamente.

(ii) A assinatura não é válida, o protocolo falha e A participou de boa-fé na verificação.

(iii) A assinatura é válida e o protocolo tem sucesso.

É necessário que um *agente de confiança* TA (um juiz) decida qual das três situações ocorre. Para isso estabelece com A o chamado **protocolo de repúdio** onde, essencialmente, executa o protocolo de verificação duas vezes.

Protocolo de Repúdio

TA escolhe números aleatórios μ, v e desenvolve o protocolo de verificação com A recolhendo os valores y e z.
Se for $z = x^{\mu}\,\alpha^{v} \pmod p$, o TA declara a assinatura válida e para.

TA escolhe novos números aleatorios μ', v' e desenvolve o protocolo de verificação com A recolhendo valores y' e z'.
Se for $z' = x^{\mu'}\,\alpha^{v'} \pmod p$, TA declara a assinatura válida e para.

TA calcula $\delta = (z\,\alpha^{-v})^{\mu'} \pmod p$ e $\delta' = (z' x\,\alpha^{-v'})^{\mu} \pmod p$.
Se for $\delta = \delta'$ então a assinatura s é fraudulenta e A tem razão em repudiá-la; em caso contrário a assinatura s é legítima e A está a tentar repudiá-la fraudulentamente.

Justificação

1. Se TA e A executaram o protocolo correctamente então tanto δ como δ' são iguais a $s^{b\,\mu\,\mu'}$. Portanto só pode ocorrer $z \neq x^{\mu}\,\alpha^{v} \pmod p$ e $z' \neq x^{\mu'}\,\alpha^{v'} \pmod p$ se for $s \neq x^{a} \pmod p$; isto é, s é uma falsificação.
2. Se ocorrer $z \neq x^{\mu}\,\alpha^{v} \pmod p$ e $z' \neq x^{\mu'}\,\alpha^{v'} \pmod p$ então A não tem possibilidade (devido à complexidade de LogDisc(p)) de intervir fraudulentamente no protocolo de modo a que se verifique $\delta = \delta'$; qualquer intervenção fraudulenta de A no protocolo conduzirá a $\delta \neq \delta'$.

Assinaturas "fail-then-stop"

Assinaturas onde o signatário consegue provar que uma assinatura que lhe é atribuída é falsa.

Se por acaso um intruso, munido de meios de cálculo poderosos, consegue gerar uma falsa assinatura que satisfaz o processo de verificação então esse facto é detectado e o mecanismo de assinaturas pode ser abandonado.

Parâmetros específicos A geração de chaves é dividida entre o signatário A e um *trusted agent* TA.

(i) O TA gera os parâmetros p, q, α e $\beta = \alpha^a \pmod p$ como no DSA. O valor a é privado do TA e os valores $\langle p, q, \alpha, \beta \rangle$ são públicos.

(ii) O signatário A gera números aleatórios $a_1, b_1, a_2, b_2 \in \mathbb{Z}_q$ e calcula $\beta_1 = \alpha^{a_1}\, \beta^{b_1} \pmod p$ e $\beta_2 = \alpha^{a_2}\, \beta^{b_2} \pmod p$

(iii) A chave pública de A é $k \equiv \langle \beta_1, \beta_2 \rangle$ e a chave privada de A é $K \equiv \langle a_1, b_1, a_2, b_2 \rangle$.

Geração da assinatura para uma <u>única</u> mensagem $x \in \mathbb{Z}_q$

(i) Calcula-se $\gamma = a_1 + x\, a_2 \pmod q$ e $\delta = b_1 + x\, b_2 \pmod q$.

(ii) A assinatura é o par $s = \langle \gamma , \delta \rangle$.

Verificação da assinatura

Dado $s = \langle \gamma , \delta \rangle$ e x

(i) Calcula-se $\mu = \beta_1\, \beta_2^x \pmod p$ e $\upsilon = \alpha^\gamma\, \beta^\delta \pmod p$.

(ii) A assinatura é aceite sse $\mu = \upsilon$.

Justificação
Com todas as exponenciações determinadas $\pmod p$ temos $\beta_1 = \alpha^{a_1 + a\, b_1}$ e $\beta_2 = \alpha^{a_2 + a\, b_2}$. Então $\beta_1\, \beta_2^x$ e $\alpha^\gamma\, \beta^\delta$ são ambos iguais a $\alpha^{a_1 + a_2\, x + a\, (b_1 + b_2\, x)}$.

Nota Este algoritmo só serve para gerar uma única assinatura já que duas assinaturas com o mesmo K permitem reconstruir os quatro números $\langle a_1, b_1, a_2, b_2\rangle$.

Prova de falsificação

Suponhamos que um intruso conseguia, sem conhecer a chave privada $\langle a_1, b_1, a_2, b_2\rangle$, usar um qualquer processo que lhe permitia calcular uma assinatura $s' = \langle\, \gamma'\,,\, \delta'\,\rangle$ que satisfaz a condição de verificação:

$$\beta_1\, \beta_2^x = \alpha^{\gamma'}\, \beta^{\delta'} \pmod p \qquad (\dagger)$$

Se s' for uma falsificação, A prova-o calculando a tal que $\beta = \alpha^a \pmod p$; i.e. resolvendo o problema do logaritmo discreto.

(i) A calcula a assinatura $s = \langle\, \gamma\,,\, \delta\,\rangle$ de x determinando $\gamma = a_1 + x\, a_2 \pmod q$ e $\delta = b_1 + x\, b_2 \pmod q$.

(ii) Verificando $s \neq s'$, A calcula

$$a \;=\; (\gamma - \gamma')\, (\delta - \delta')^{-1} \pmod q \qquad (\ddagger)$$

Justificação
De $\alpha^\gamma\, \beta^\delta \;=\; \alpha^{\gamma'}\, \beta^{\delta'} \pmod p$ concluímos $\gamma + a\, \delta \;=\; \gamma' + a\, \delta' \pmod q$ o que permite calcular a por ($\ddagger$).

Técnicas Criptográficas– Módulo VI

Criptografia de Chave Pública

(Protocolos Certificados)

1. Certificação. Árvores de certificação.
2. Protocolos de Indentificação.
   1. Protocolo de Schnorr
   2. Protocolos de Conhecimento Zero: Fiat-Shamir, Guillou-Quisquater e variantes.
3. Protocolos de Acordo de Chaves.
   1. Diffie-Hellman e El Gamal.
   2. Protocolos MTI e suas variantes.
   3. Protocolo STS (*Station To Station*)
   4. Chaves implicitamente certificadas (Günther) e chaves auto-certificadas (Girault).

Certificação

Um **certificado** é um documento sobre um outro documento x, apresentado como **prova** de que x satisfaz uma dada propriedade P.

A aceitação da prova baseia-se na **autoria** e na **integridade** do certificado: a prova é aceite porque provêm de um agente de confiança (*trusted agent*, ou TA) e existe a certeza que não foi alterada desde que foi criada. Ambas as propriedades são asseguradas pela presença de uma assinatura do TA sobre o certificado.

Notação

O certificado-documento é representado por $\|x\|^P$. A assinatura do TA sobe esse documento é $(\|x\|^P)_{\mathsf{TA}}$. Um certificado justificado é assim um par formado pelo certificado-documento e pela assinatura e representa-se por

$$[\![x]\!]^P \equiv \langle\, \|x\|^P ,\, (\|x\|^P)_{\mathsf{TA}} \,\rangle$$

No que se segue todos os certificados consideram-se justificados.

Exemplos de Certificados

Chaves Públicas Os documentos certificados são pares $\langle\, A ,\, k \,\rangle$, em que A identifica um agente e k uma sua chave pública. A propriedade certificada é a titularidade de k por A ($A\ \mathsf{titula}\ k$). O certificado é representado por $[\![A, k]\!]^T$.

Segurança Os documentos certificados [a] são programas p (representados pelas suas *fontes*). A propriedade certificada é "p cumpre os requisitos de segurança, correcção, etc"; os certificados são $[\![p]\!]^S$.

Estampilhas de tempo Os documentos certificados são pares $\langle\, x ,\, t \,\rangle$ formados por um documento x e por um *tempo* t. A propriedade certificada é "x está criado no instante t" e o certificado é $[\![x, t]\!]^K$ [b]

[a] Todos os "grandes documentos" ocorrem nos certificados, via funções de *hash*.
[b] "T" para *titularidade*, "S" para *safety* e "K" para *knowledge*.

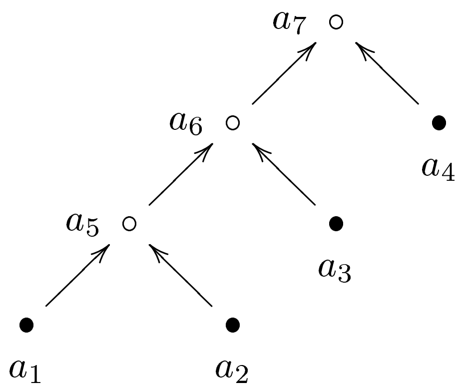
Árvores de Certificação

O texto x, normalmente, ocorre no certificado na forma *hash* $\overline{x}$. Pode-se aproveitar isto para incluir <u>vários</u> documentos no mesmo certificado.

Uma **árvore de certificação** é uma árvore binária cujos nodos são valores de *hash* construídos segundo as regras:

(i) As folhas $\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n}$ são valores de *hash* dos textos a certificar,

(ii) Um nodo com descendentes imediatos a e b tem valor $\overline{a \| b}$. [a]



$$a_i = \overline{x_i} \qquad i = 1 \cdots 4$$
$$a_5 = \overline{a_1 \| a_2}$$
$$a_6 = \overline{a_5 \| a_3}$$
$$a_7 = \overline{a_6 \| a_4}$$

O certificado contém apenas a **raiz** da árvore (neste exemplo, a_7).

Para verificar a certificação de um texto público (por exemplo, x_1), um agente precisa de conhecer o certificado e conhecer o *caminho* desde a folha do texto até à raiz; para tal, neste exemplo, é-lhe fornecido x_1, a sequência $\langle a_2, a_3, a_4 \rangle$.

O verificador calcula sucessivamente

$$u_0 = \overline{x_1} \qquad u_1 = \overline{u_0 \| a_2} \qquad u_2 = \overline{u_1 \| a_3} \qquad u_3 = \overline{u_2 \| a_4}$$

e compara o resultado final u_3 com o conteúdo do certificado.

[a] "$\|$" denota a concatenação de *strings*.

Identificação

Um agente B (o **verificador**) pretende estabelecer a identidade de um agente A (o **identificado**) de tal modo que B (ou qualquer outro intruso) não recebe informação que lhe permita identificar-se como A perante um terceiro agente C.

Genericamente o mecanismo de identificação deve ser um **protocolo de conhecimento zero**: i.e., um processo pelo qual uma agente A demonstra a validade de uma asserção P sem revelar qualquer informação sobre a asserção (além do facto de ser válida).

O mecanismo "*login-password*" é uma aproximação muito má deste ideal; num canal aberto um intruso captura toda a informação necessária à identificação; num canal fechado B recebe toda a informação que lhe permite fazer-se passar por A.

Uma abordagem melhor são os protocolos "*desafio-resposta*" baseados nos princípios do "*conhecimento zero*" que aproximam a situação ideal tanto quanto o esforço computacional investido no processo.

Um protocolo de conhecimento zero diz-se

Completo quando na presença de um verificador e de um identificado honestos o protocolo tem sucesso a menos de uma probabilidade arbitrariamente pequena.

Correcto [a] quando a probabilidade de B usar a informação obtida de A para se identificar como A perante terceiros é equivalente à probabilidade de recuperação da informação privada de A.

[a] Tradução da designação "*sound*".

Protocolo de Schnorr

Protocolo *desafio-resposta* baseado no problema do logaritmo discreto, recorrendo a certificados emitidos por um TA.

Está implícito que todos os agentes possuem meios para verificar os certificados emitidos pelo TA.

Inicialização

1. O TA gera um conjunto de parâmetros DSS $\langle p, q, \alpha \rangle$ e um número $t \geq 40$ (chamado *nível de segurança*) tal que $2^t \leq q$.
2. Cada agente A obtém do TA um certificado $[\![p, q, \alpha]\!]$ dos parâmetros gerais, gera um valor aleatório a e calcula $\beta = \alpha^{-a} \pmod{p}$.
3. Após identificação de A perante o TA, este emite o certificado de titularidade $\mathsf{cert}_A = [\![A, \beta]\!]^T$

Protocolo de Identificação de A perante B [a]

1. A gera $\omega \in \mathbb{Z}_q^*$, calcula $\gamma = \alpha^\omega$ e envia-o para B juntamente com cert_A (*intensão*).
2. B aceita cert_A (recolhendo A, β e verificando a validade do certificado), gera $0 < x < 2^t$ e envia-o a A (*desafio*).
3. A calcula $y = a\,x + \omega$ e envia-o para B (*resposta*).
4. B verifica $\gamma = \alpha^y\,\beta^x$ (*aceitação*).

$$A \xrightarrow{\gamma=\alpha^\omega,\quad \mathsf{cert}_A} B$$

ω

$$A \xleftarrow{\quad x \quad} B$$

A,β,x

$$A \xrightarrow{y=a\,x+\omega} B$$

$\gamma=\alpha^y\,\beta^x$

[a] Todas os produtos e exponenciações são feitas módulo p e todas as somas são feitas módulo q.

Completude

Temos $\omega = y - a\,x \pmod q$ se e só se $\alpha^\omega = \alpha^y\,(\alpha^{-a})^x \pmod p$; atendendo às construções de β e γ conclui-se

$$y = a\,x + \omega \pmod q \quad \text{sse} \quad \gamma = \alpha^y\,\beta^x \pmod p$$

Correcção

Se um intruso I conseguir, com probabilidade maior do que 2^{-t} prever respostas y que sejam aceites por B para vários desafios possíveis x, então, para dois desafios diferentes $x_1, x_2 \leq 2^t$ usa os respectivos y_1, y_2 que consegue prever para resolver o sistema de equações $\{y_1 = a\,x_1 + \omega\ ,\ y_2 = a\,x_2 + \omega\}$ e determinar a.

Segurança

(i) Para reconstruir um y que verifique $\gamma = \alpha^y\,\beta^x \pmod p$, um intruso que conheça x, γ mas não conheça a e ω, tem de resolver o problema do logaritmo discreto (PLD).

(ii) Para se fazer identificar como A perante terceiros o verificador B apenas conhece informação pública β, γ e y. De β e γ não consegue recuperar a ou ω porque isso seria resolver o PLD.

(iii) Se A usar duas vezes o mesmo ω então do par de equações módulo q, $\{y_1 = a\,x_1 + \omega\ ,\ y_2 = a\,x_2 + \omega\}$, é possível recuperar a.

(iv) Um intruso que consiga prever o desafio x pode-se fazer passar por A escolhendo previamente o y (que irá enviar como resposta ao desafio) e inicia o protocolo enviando γ que calcula como sendo $\alpha^y\,\beta^x \pmod p$.

Ao receber o desafio x envia o y escolhido anteriormente. Dado o modo como γ foi calculado, B aceita a identificação porque a condição $\gamma = \alpha^y\,\beta^x$ é verificada.

Se x for criptográficamente seguro a probabilidade de um tal ataque é limitada a 2^{-t}.

A nota (iv) ilustra porque é que o protocolo **não é** de "conhecimento zero"; B passa a ter uma solução $\langle \gamma, y, x \rangle$ da equação $\gamma = \alpha^y\,\beta^x \pmod p$ que não conheceria se o protocolo não se realizasse.

Protocolos de Conhecimento Zero

A identifica-se perante B provando-lhe que conhece um "segredo" v sem revelar a B (ou qualquer intruso) qualquer informação sobre o segredo que não possa ser obtida sem realizar o procolo.

Protocolo Fiat-Shamir

Baseado no problema da factorização de inteiros (PFI) equivalente ao problema do cálculo de raízes quadradas.

Inicialização

(i) O TA estabelece um módulo RSA $m = p\,q$ e divulga $[\![m]\!]$.

(ii) Cada agente A gera um $v \in \mathbb{Z}_m^*$, calcula $\mu = v^2 \pmod{m}$ e regista μ junto ao TA como chave pública obtendo $\mathsf{cert}_A = [\![A, \mu]\!]^T$.

Assume-se que todos os agentes têm acesso a todos os certificados e dispõem de meios para os validar.

Protocolo: São executados t iterações de: [a]

(i) A gera $\omega \in \mathbb{Z}_m^*$, calcula $\gamma = \omega^2$ e envia-o a B (*intensão*).

(ii) B gera um bit $x \in \mathbb{Z}_2$ e envia-o a A (*desafio*).

(iii) A envia $y = \omega\, v^x$ a B (*resposta*).

(iv) B rejeita a identificação se $y = 0$ ou $y^2 \neq \gamma\, \mu^x$.

$$A \xrightarrow{\gamma=\omega^2} B$$

ω

$$A \xleftarrow{x\in\{0,1\}} B$$

μ, x

$$A \xrightarrow{y=\omega\, v^x} B$$

$y^2 = \gamma\, \mu^x$

Se no final das t iterações a identificação não tiver sido rejeitada é aceite.

[a] Todas as operações são efectuadas módulo m.

Completude

Como ω, υ são invertíveis em $\mathbb{Z}_m^*$ nunca pode ser $y = \omega\,\upsilon^x = 0$. Se $y \neq 0$ tem-se $y = \omega\,\upsilon^x$ se só se $y^2 = \omega^2\,(\upsilon^2)^x = \gamma\,\mu^x$.

Conhecimento Zero

A resposta $y = \omega$ (correspondente ao desafio $x = 0$) é independente de υ. A resposta $y = \omega\,\upsilon$ (que corresponde a $x = 1$) não fornece informação sobre υ porque, neste caso, B desconhece ω.

Os triplos $\langle \gamma\,,\,x\,,\,y \rangle$, conhecidos de B, podem ser obtidos sem o protocolo gerando valores aleatórios y e calculando $\gamma = \mu^{-1}\,y^2$ ou $\gamma = y^2$ consoante o bit x é 1 ou 0.

Correcção

Um intruso I (que não conheça υ) tem duas estratégias possíveis para se tentar fazer passar por A:

(1) Escolhe um ω e envia $\gamma = \omega^2$. Se o desafio que recebe for $x = 0$ então ele envia a resposta correcta $y = \omega$ e não é detectado. Se o desafio for porém $x = 1$ ele só pode enviar a resposta correcta y se conseguir resolver a equação $y^2 = \gamma\,\mu = \omega^2\,\mu$; isto é equivalente a determinar a raiz quadrada de μ e, assim, calcular υ.

Assumindo a intratabilidade do cálculo das raízes quadradas, a resposta a $x = 1$ só pode ser um valor aleatório y. Globalmente a probabilidade de não ser detectado é $\simeq 2^{-1}$

(2) Escolhe uma resposta y aleatória e calcula $\gamma = \mu^{-1}\,y^2$. Se o desafio for $x = 1$ envia a resposta correcta y. Porém se o desafio for $x = 0$ obter a resposta correcta y_0 seria equivalente a resolver a equação $y_0^2 = \gamma = \mu^{-1}\,y^2$ o que equivaleria a determinar a raiz quadrada de μ e, assim, calcular υ.

Assumindo de novo a intratabilidade do PFI, a probabilidade de não ser detectada é de novo $\simeq 2^{-1}$

Repetindo a experiência t vezes a probabilidade de nunca ser detectado é $\simeq 2^{-t}$ que é desprezável para valores de t suficientemente grandes.

Protocolo GQ (Guillou-Quisquater)

O protocolo de Fiat-Shamir requer $3\,t$ mensagens e uma chave privada v da ordem de m (ou seja, 1024 ou mais bits). Isto torna-o pouco aceitável em implementações com *smart-cards* que são as mais interessantes nos protocolos de identificação. O protocolo GQ extende o Fiat-Shamir reduzindo estes requisitos computacionais.

Inicialização

(i) O TA estabelece um módulo RSA $m = p\,q$ e um par de expoentes $a, b \in \mathbb{Z}^*_{\phi(m)}$ com $a\,b = 1 \pmod{\phi(m)}$. Torna públicos m, b sob a forma de um certificado $[\![m, b]\!]$.

(ii) É pública um função injectiva r que associa a cada agente A um *identificador de redundância* $i = r(A)$, com $i \in \mathbb{Z}^*_{\phi(m)}$. TA envia a A <u>através de um canal seguro</u> o "segredo" $v = i^{-a} \pmod{m}$.

Protocolo

São executadas t iterações do protocolo seguinte: [a]

(i) A gera $\omega \in \mathbb{Z}_m$ não-zero, calcula $\gamma = \omega^b$ e envia $\langle A\,,\gamma \rangle$ a B (*intensão*).

$$A \xrightarrow{A\,,\,\gamma=\omega^b} B$$
$$\omega$$

(ii) B gera um $0 < x \leq b$ e envia-o a A (*desafio*).

$$A \xleftarrow{\;x\;} B$$
$$x$$

(iii) A calcula $y = \omega\, v^x$ e envia-o para B (*resposta*).

$$A \xrightarrow{y=\omega\, v^x} B$$
$$y^b\, r(A)^x = \gamma$$

(iv) B calcula $z = r(A)^x\, y^b$ e rejeita a identificação se $z = 0$ ou $z \neq \gamma$.

Se B não rejeita a identificação ao fim das t iterações, aceita-a.

[a] Todas as exponenciações e multiplicações efectuadas módulo m.

Completude

Em $\mathbb{Z}_m$, sendo $y = \omega\, v^x$ e $i = r(A)$, tem-se $z = y^b\, i^x = \omega^b\, v^{b\,x}\, i^x$. Como $\gamma = \omega^b$ e $v = i^{-a}$ temos $z = \gamma\,(i^{-x})^{a\,b}\, i^x = \gamma\, i^{-x}\, i^x = \gamma$. [a]

Conhecimento Zero

O protocolo GQ é análogo ao Flat-Shamir em que o desafio x, em vez de tomar apenas dois valores $x = 0$ ou $x = 1$, assume um qualquer valor no intervalo $1 \ldots b$.

Os argumentos para a satisfação da propriedade de "conhecimento zero" são, portanto, os mesmos que foram usados no protocolo FS.

Correcção

Usando de novo a analogia entre GQ e FS podemos ver um intruso com b hipóteses de dar a resposta correcta, uma para cada um dos $x \in 1 \ldots b$.

Assumindo que o desafio é x, o intruso pode escolher previamente y e calcular $\gamma = y^b\, r(A)^x$. Se o desafio recebido for exactamente x, o intruso pode dar a resposta certa y; porém se não for não tem possibilidade de calcular essa resposta sem conhecer v.

A probabilidade de não ser detectado é, em cada hipótese, $\simeq b^{-1}$. Ao fim de t tentativas a probabilidade de não ser detectado é $\simeq b^{-t}$. Como normalmente $b \gg 2$, para obter uma segurança idêntica à do protocolo FS é necessário um número de iterações t muito menor.

Complexidade Computacional

Para um valor de b suficientemente grande basta um número de iterações $t = 1$ ou $t = 2$.

A chave privada $v = r(A)^{-a} \pmod m$ é ainda da ordem de 1024 ou mais bits.

Reduções maiores no tamanho da chave requerem, normalmente, o protocolo de Schnorr

[a] Pela propriedade fundamental do RSA, $u^{a\,b} = u \pmod m$.

Protocolos de Acordo e Distribuição de Chaves

O problema da distribuição de segredos sobre canais abertos é um dos mais vastos em Criptografia para o qual têm vindo a ser desenvolvidas muitas técnicas. Neste módulo apenas se irão apresentar técnicas baseadas em criptografia de chave pública.

O objectivo de todos os protocolos é a **geração** e **partilha** de um segredo K pelos agentes A e B, usando canais abertos e restringindo o conhecimento do segredo a esses dois agentes (ou um TA).

Características dos Protocolos

Chaves Tipicamente K é usado como chave para uma única comunicação entre A e B (uma **chave de sessão**) evitando assim:

- (i) A necessidade de armazenar muitas chaves para comunicações numa comunidade.
- (ii) A quantidade de informação associada a uma mesma chave diminuindo a probabilidade de ataques.
- (iii) Um longo tempo de exposição a ataques por uma chave.

Natureza Todos os protocolos aqui referidos pertencem à categoria descrita como **acordo de chaves** o que significa que K é determinado pela vontade conjunta dos dois agentes.

Confirmação da chave Quando um agente se assegura que o outro interveniente no protocolo tem conhecimento da chave acordada.

Autenticação A **autenticação de chaves** assegura a cada agente que apenas ele e o outro interveniente têm acesso à chave acordada. A **autenticação de entidade** assegura-lhe a identidade de todos os agentes intervenientes.

Frescura e controlo A noção de *frescura* coloca-se quando um dos intervenientes quer se assegurar que a chave é nova e não é uma chave antiga que está a ser reutilizada por iniciativa do outro interveniente ou de um intruso. A noção de *controlo* põe-se quando um dos intervenientes pode impor uma chave determinada.

Protocolos baseados no PLD

O ponto de referência é o protocolo de Diffie-Hellman.

$$\begin{array}{ccc} A & \underset{\gamma=\alpha^b}{\overset{\beta=\alpha^a}{\rightleftarrows}} & B \\ a & & b \end{array}$$

$$\gamma^a = \beta^b = \alpha^{a\,b}$$

DH é vulnerável a ataques "*man-in-the-middle*" que resultam da falta de identificação mútua dos agentes. Como consequência todos os protocolos desta família complementam DH com um **esquema de identificação**.

Protocolo El Gamal

Protocolo de um passo em que a chave pública do agente B deve ser previamente conhecida pelo agente A.

Inicialização (*executa uma só vez*)

(i) B gera parâmetros El Gamal $\langle p, \alpha\rangle$, uma chave privada $b \in \mathbb{Z}^*_{p-1}$ e calcula $\gamma = \alpha^b \pmod{p}$.

(ii) B identifica-se perante o TA e obtém $\mathsf{cert}_B = [\![B, p, \alpha, \gamma]\!]$.

Protocolo (*executa uma vez para cada chave K*)

(i) A obtém cert_B, gera $\omega \in \mathbb{Z}^*_{p-1}$ aleatório e envia a B o valor $h = \alpha^\omega \pmod{p}$.

(ii) A calcula a chave $K = \gamma^\omega \pmod{p}$ e B calcula a mesma chave como $h^b \pmod{p}$.

$$\begin{array}{ccc} A & \xrightarrow{h=\alpha^\omega} & B \\ \omega & & b \end{array}$$

$$\gamma^\omega = h^b$$

Segurança O agente B não identifica o outro interveniente no protocolo. Nenhum interveniente tem garantias de *frescura* da chave. O protocolo não contém nenhuma forma de identificação ou de confirmação de chave.

Protocolos MTI (*Matsumoto, Takashima e Imai*)

Protocolos de dois passos, variante de DH, com identificação mútua e sem requerer assinaturas dos intervenientes.

Parâmetros e Inicialização

(i) O TA gera um primo p para o qual o PLD é intratável e um gerador α de $\mathbb{Z}_p^*$ e publicita o certificado $[\![p, \alpha]\!]$.

(ii) A gera $a \neq 0 \in \mathbb{Z}_{p-1}\,1$, calcula $\beta = \alpha^a \pmod{p}$, identifica-se perante o TA e obtém o certificado $[\![A, \beta]\!]$. Análogamente B gera b, calcula $\gamma = \alpha^b \pmod{p}$ e obtém o certificado $[\![B, \gamma]\!]$.

Protocolos

Assume-se que todos os intervenientes têm acesso a todos os certificados e têm meios para os validar.[a]

São definidos 4 protocolos (identificados por A0, B0, C0 e C1) que se distinguem pelas mensagens M e N trocadas e pela forma como as chaves K e J são determinada por cada um dos agentes. Os valores de M, N, I e J são definidos na tabela 1.

1. A gera $\mu \neq 0 \in \mathbb{Z}_{p-1}$, calcula $M(\mu)$ e envia-o a B.
2. B gera $\upsilon \neq 0 \in \mathbb{Z}_{p-1}$, calcula $N(\upsilon)$, e envia-o a A.
3. A calcula $I(a, \mu, N)$ e B calcula $J(b, \upsilon, M)$; ambos os valores coincidem com a chave acordada K.

$$\begin{array}{ccc} A & \begin{array}{c} \xrightarrow{M(\mu)} \\ \xleftarrow[N(\upsilon)]{} \end{array} & B \\ \mu & & \upsilon \end{array}$$

$$\begin{array}{cc} A & B \\ I(a,\mu,N) & J(b,\upsilon,M) \end{array}$$

[a]Nomeadamente, os certificados são públicos e a chave pública do TA é conhecida e tida como legítima.

Protocolo	M	N	I	J	$K =$
A0	α^{μ}	α^{v}	$N^{a}\,\gamma^{\mu}$	$M^{b}\,\beta^{v}$	$\alpha^{a\,v+b\,\mu}$
B0	γ^{μ}	β^{v}	$N^{a^{-1}}\,\alpha^{\mu}$	$M^{b^{-1}}\,\alpha^{v}$	$\alpha^{\mu+v}$
C0	γ^{μ}	β^{v}	$N^{a^{-1}\,\mu}$	$M^{b^{-1}\,v}$	$\alpha^{\mu\,v}$
C1	$\gamma^{\mu\,a}$	$\beta^{v\,b}$	N^{μ}	M^{v}	$\alpha^{a\,\mu\,b\,v}$

Table 1: Protocolos MTI

Notas

1. Os protocolos MTI fornecem autenticação mútua das chaves mas não fornecem confirmação das chaves nem autenticação dos agentes.

2. No primeiro protocolo A ou B não requerem informação *à priori* sobre o outro agente; pode cada um iniciar o protocolo e enviar a sua chave pública (β ou γ) num certificado junto com as mensagens M ou N . Nos restantes protocolos cada agente tem de conhecer, antes de iniciar a troca de mensagens, a informação pública certificada do outro.

3. Normalmente, depois de B (ou, por simetria, A) conhecer a chave pública de A deve-se assegurar que A conhece a respectiva chave privada.

   Caso contrário um intruso I pode fazer um **ataque de substituição de fonte** da seguinte forma:

   (i) I obtém o certificado de A e identifica-se perante o TA reclamando β como a sua chave pública: obtém o certificado $\mathsf{cert}_I \equiv [\![I, \beta]\!]$.

   (ii) Quando A envia a mensagem $M(\mu)$ juntamente com o seu certificado, o intruso substitui cert_A por cert_I . A mensagem $N(v)$ é reencaminhada para A.

   (iii) O agente B julga que as mensagens que recebe cifradas com a chave acordada K provêm de I quando, realmente, elas provêm de A. O intruso não influencia o conteúdo dessas mensagens mas influencia a percepção que B tem sobre a fonte das mensagens.

Protocolo STS (*Station To Station*)

Protocolo de três passos com mútua autenticação de identidade usando um esquema de assinaturas e uma cifra simétrica.

Parâmetros

(i) O TA gera parâmetros DH $\langle\, p\,,\, \alpha\,\rangle$ e emite o certificado $[\![p, \alpha]\!]$.

(ii) É escolhido um esquema de assinaturas. Perante identificação, TA emite certificados de chave pública $\mathsf{cert}_A = [\![A, k_A]\!]$ e $\mathsf{cert}_B = [\![A, k_B]\!]$.

(iii) É escolhida uma cifra simétrica segura susceptível de usar a chave acordada K .

Protocolo

Assume-se que cada agente tem conhecimento dos certificados públicos e meios para verificar a sua validade.

(1) A gera $\mu \neq 0 \in \mathbb{Z}_{p-1}$, calcula $\beta = \alpha^\mu$ e envia-o a B.[a]

(2) B gera $\upsilon \neq 0 \in \mathbb{Z}_{p-1}$, calcula a resposta $\gamma = \alpha^\upsilon$, o segredo $K = \beta^\upsilon$ e a assinatura $s = (\gamma, \beta)_B$. Envia γ juntamente com $\{s\}_K$ (i.e. a assinatura s cifrada com K).

(3) A recebe $\langle\, \gamma\,,\, \{s\}_K\,\rangle$ e determina $K = \gamma^\mu$. Usa K para obter s . Com os valores $\langle\, \gamma\,,\, \beta\,\rangle$ e a chave pública k_B (recolhida de cert_B) verifica a assinatura s .
Aceitando a assinatura, A gera $s' = (\beta, \gamma)_A$ e envia $\{s'\}_K$ a B.

(4) B recebe $\{s'\}_K$, usa K para determinar $s' = (\beta, \gamma)_A$ e, com a chave pública k_A extraída de cert_A e o par $\langle\, \beta\,,\, \gamma\,\rangle$, verifica s' .
Se tiver sucesso B aceita que K é conhecida de A.

[a] Todas as exponenciações são realizadas módulo p.

$$A \xrightarrow{\quad\beta = \alpha^{\mu}\quad} B$$

$$\mu \qquad\qquad\qquad\qquad v\,,\ \gamma = \alpha^{v}$$

$$A \xleftarrow{\quad\gamma\,,\ \{s\}_K\quad} B$$

$$K = \gamma^{\mu}\,,\ s \qquad\qquad\qquad s = (\gamma,\beta)_B, K = \beta^{v}$$

$$A \xrightarrow{\quad\{s'\}_K\quad} B$$

$$\mathsf{ver}_B(s)\,,\ s' = (\beta,\gamma)_A \qquad\qquad\qquad s'\,,\ \mathsf{ver}_A(s')$$

Notas

1. O protocolo STS gera a chave K exactamente do mesmo modo que o DH mas usa o esquema de assinaturas e a cifra para fornecer autenticação mútua de identidade e confirmação da chave.
   Nem o ataque *man-in-the-middle* (característico do DH) nem o ataque de substituição da fonte (característico do MTI) são aqui possíveis.

2. A autenticação mútua de identidade dos agentes é assegurada nos passos onde se realiza a verificação das assinaturas s e s'.

3. A confirmação mútua é assegurada porque o desenvolvimento do protocolo exige, a partir do momento que cada agente tem informação para gerar K, que todas as mensagens sejam cifradas com essa chave.

4. Os certificados cert_A e cert_B podem ser previamente distribuídos (e nesse caso o anonimato dos agentes é preservado perante um intruso que escute as comunicações) ou ser enviado nas duas primeiras mensagens.

Chaves implicitamente certificadas de Günther

Todos os protocolos baseados no PLD assentam na geração, por um agente A, de informação privada a e na divulgação de expoentes $\beta = \alpha^a$. O mecanismo de chaves públicas implicitamente certificadas permite a um agente B gerar β (sem conhecer obviamente a) tendo garantias de que este é a exponenciação de a. Essa garantia é obtida sem ter de recorrer à certificação explícita da chave pública β.

Parâmetros genéricos O par $\langle\, p\,,\,\alpha\,\rangle$ do protocolo DH.

Objectivo TA gera distribui informação privada a a um agente A, gera informação pública α_A (chamada *chave pública reconstruída*) de tal modo que qualquer agente B pode determinar $\beta_A = \alpha_A^a \pmod p$ a partir de informação pública certificada.

Distribuição da informação privada[a]

(i) O TA selecciona, como chave privada $t \in \mathbb{Z}^*_{p-1}$ e calcula $\delta = \alpha^t$.

(ii) Para cada agente A, o TA gera $\omega \in \mathbb{Z}^*_{p-1}$ e calcula $\alpha_A = \alpha^\omega$.

(iii) O TA calcula ainda $a = \omega^{-1}(\bar{A} - t\,\alpha_A)$ e envia $\langle\,\alpha_A\,,\,a\,\rangle$ a A por canal seguro.

Nota O par $\langle\,\alpha_A\,,\,a\,\rangle$ não é mais do que a assinatura El Gamal de $\bar{A}$ com t como chave privada do TA.

Distribuição da informação pública e geração de chaves públicas

1. O TA publicita $\langle p, \alpha, \delta, A, \alpha_A\rangle$.
2. Cada agente B determina $\beta_A = \alpha_A^a$ calculando $\alpha^{\bar{A}}\,\delta^{-\alpha_A}$.

[a] Todas as exponenciações e multiplicações são efectuadas módulo p. Todas as somas são efectuadas módulo $(p-1)$.

Chaves auto-certificadas de Girault

Um mecanismo com objectivos análogos ao anterior só que as chaves são certificadas pelo próprio detentor da chave privada já que não são conhecidas pelo TA.

Parâmetros

O TA gera um módulo RSA $m = pq$, um par $\mu, \upsilon \in \mathbb{Z}^*_{\phi(m)}$ tal que $\mu\,\upsilon = 1 \pmod{\phi(m)}$ e $\alpha \in \mathbb{Z}^*_m$ de ordem máxima.

Os parâmetros $\langle m, \mu, \alpha \rangle$ são públicos e υ é privado do TA.

Objectivo[a]

Gerar informação privada a de A e uma chave pública reconstruída α_A de tal forma que seja público reconstruir $\beta = \alpha^a$ sem conhecer a.

Geração de chaves

(1) Cada agente gera um inteiro a e $\beta = \alpha^a$. Identifica-se perante TA e envia-lhe β <u>por canal seguro</u>.

(2) Após A demonstrar ao TA[b] o conhecimento do expoente a este determina $\alpha_A = (\beta - \bar{A})^\upsilon$ e publicita este valor.

Geração da chave pública

Um agente B calcula β como $\alpha_A^\mu + \bar{A}$.

[a] Todas as exponenciações e somas são efectuadas módulo m.

[b] Por exemplo, aceitando um desafio $\gamma = \alpha^x$ do TA e calculando γ^a. O TA compara esse valor com β^x.

DH com chaves implicitamente certificadas

Como exemplo de utilização de chaves implicitamente certificadas ou auto-certificadas podemos considerar uma simples variante ao protocolo básico de Diffie-Hellman usando chaves implicitamente certificadas

Inicialização

O TA estabelece, usando o mecanismo de Günther, pares $\langle\, \alpha_A \,,\, a \,\rangle$ e $\langle\, \alpha_B \,,\, b \,\rangle$ fornecidos por canais seguros a A e B respectivamente.

Protocolo

(1) A envia a B o par $\langle\, A \,,\, \alpha_A \,\rangle$.

(2) B determina $\beta_A = \alpha_A^a$ pelo mecanismo de Günther; gera $x \neq 0 \in \mathbb{Z}_{p-1}$ e envia a A o valor α_A^x juntamente com $\langle\, B \,,\, \alpha_B \,\rangle$.

(3) A determina $\beta_B = \alpha_B^b$ pelo mecanismo de Günther; gera $y \neq 0 \in \mathbb{Z}_{p-1}$ e envia a B o valor α_B^y.

(4) A determina $K = (\alpha_A^x)^a\, \beta_B^y$.
B determina $K = (\alpha_B^y)^b\, \beta_A^x$.
Ambos valores coincidem com $\alpha_A^{a\,x}\, \alpha_B^{b\,y}$.

$$\begin{array}{ccc} A & \xrightarrow{\quad A\,,\,\alpha_A\quad} & B \\ & & \beta_A \\[1em] A & \xleftarrow{\quad B\,,\,\alpha_B\,,\,\alpha_A^x\quad} & B \\ \beta_B & & x \\[1em] A & \xrightarrow{\quad \alpha_B^y\quad} & B \\ y & & \end{array}$$

Nota *Este mecanismo básico pode ser adaptado a vários protocolos substituindo certificados explícitos por informação pública implicitamente certificada ou auto-certificada.*

Técnicas Criptográficas– Módulo VII

Criptografia sobre Curvas Elípticas

1. Curvas Elípticas: conceitos base.
2. Motivação. O *standard* IEEE P1363
3. Aritmética sobre Curvas Elípticas
4. Aritmética sobre $\mathsf{GF}(2^m)$: base polinomial, bases normais e base normais.
5. Escolha de curvas elípticas e geradores.
6. Técnicas criptográficas baseadas no PLDCE:
   (a) Cifras de chave pública: El Gamal e ECES (*Elliptic Curve Encyption Scheme*)
   (b) Protocolos de acordo de chaves: MQV (*Menezes-Qu-Vanstone*) e MO (*Massey-Omura*).
   (c) Assinaturas digitais: Byberg-Rueppel e ECDSA (*Elliptic Curve Digital Signature Algoritm*).

Curvas Elípticas: Conceitos Base

A *teoria curvas elípticas e dos integrais elípticos* nasceu de Weierstraß e Gauss para, entre outros objectivos, determinar propriedades geométricas de elipses.

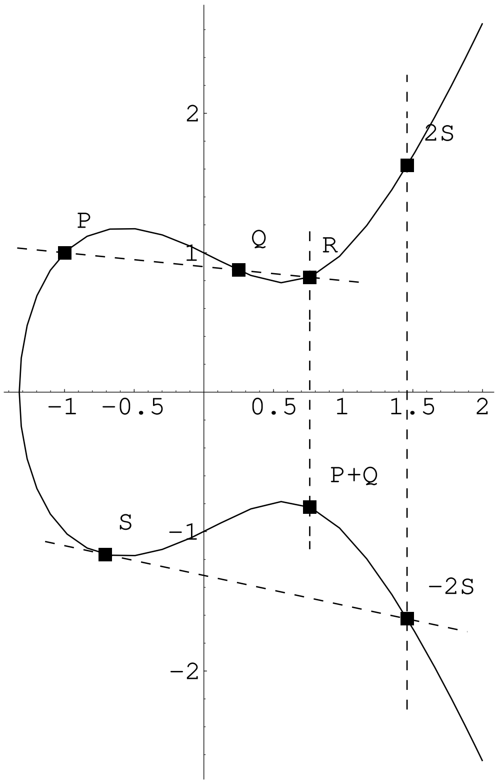
A sua aplicação a Criptografia, porém, nada tem a haver com elipses ou mesmo com curvas; a aplicação deriva do facto de os pontos sob a curva constituem um grupo abeliano com propriedades de "confusão" interessantes mantendo sempre as boas propriedades algébricas.

Historicamente as curvas elípticas sobre o corpo dos racionais são conjuntos de pontos $\langle x , y \rangle \in \mathbb{Q}^2$ que satisfaziam uma equação da seguinte forma (ou de forma análoga)

$$y^2 = x^3 + a\,x + b$$

Com a adição de um ponto especial $\mathcal{O}$ (o *ponto no infinito*) esses pontos constituiam um grupo abeliano com a operação de adição definida de modo que três pontos colineares somem sempre $\mathcal{O}$.

$$P + Q + R = \mathcal{O}$$
$$S + S + (-2\,S) = \mathcal{O}$$



Exemplo A curva sobre $\mathbb{Q}$

$$y^2 = x^3 - x + 1 \qquad P = \langle -1 , 1 \rangle \quad Q = \langle \frac{1}{4} , \frac{7}{8} \rangle \quad P + Q = \langle \frac{19}{25} , -\frac{103}{125} \rangle$$

Um conjunto de pontos no plano $\mathbb{R}^2$ desta forma, não tem as propriedades de "confusão" que são essenciais em criptografia.

Para "aumentar a confusão" sem perder as boas propriedades algébricas das CE, basta notar que as coordenadas $\langle x, y \rangle$ são objecto de apenas duas operações: *somas* e *multiplicações*.

Por isso basta que x e y sejam elementos de um *corpo* e, como em comunicações lidamos sempre com espaços de códigos finitos, bastam-nos os corpos finitos.

Existem duas classes de corpos finitos que são interessantes em criptografia e para os quais está desenvolvida uma teoria de curvas elípticas:

(i) Corpos finitos $\mathbf{F}_q$, de característica q, sendo q um primo grande.

(ii) Corpos finitos $\mathbf{F}_{2^m}$, de característica 2 (*Corpos de Galois*).

Para que as técnicas criptográficas baseadas em tais CE tenham um nível de segurança equivalente aos das técnicas "standard" teremos de ter, no primeiro caso, $q > 2^{160}$ e $m \geq 160$, no segundo caso.

Às CE com coordenadas da classe (i), assumido como sendo $\mathbb{Z}_q^*$, é costume chamar[a] *curvas de grande característica* ou "*prime*".

As CE com coordenadas da classe (ii), assumida como um corpo de Galois $\mathsf{GF}(2^m)$, costumam-se designar por *curvas elípticas binárias.*

Nota

As técnicas numéricas são ligeiramente diferentes nestas duas classes de curvas mas o nível de segurança é o mesmo. A tendência actual (uma vez resolvidos alguns problemas de eficiência de algoritmos) vai no sentido da utilização preferencial das curvas binárias devido à maior flexibilidade das codificações binárias em relação às representações em corpos finitos da forma $\mathbb{Z}_q^*$.

[a]De forma algo inapropriada.

Como exemplo de curva elíptica sobre um corpo finito vamos tomar a curva determinada pela equação

$$y^2 + x\,y = x^3 + x^2 + 1 \tag{$\dagger$}$$

definida sobre o corpo de Galois de *dimensão* 7, $\mathsf{GF}(2^7)$ caracterizado pelo polinómio $X^7 + X + 1$.

A *ordem* (i.e. o número de pontos) da curva é 142. Destes, além do ponto infinito, é de salientar o chamado *ponto singular* $S \equiv \langle\, 0\,,\, 1\,\rangle$ determinado pela coordenada $x = 0$. Os restantes 140 pontos agrupam-se em pares determinados, para cada $x \neq 0$, pelas duas "raízes" de ($\dagger$) vista como equação quadrática em y.

Para representar num plano a curva, as coordenadas $x, y \in \mathsf{GF}(2^7)$ são codificadas nos inteiros em $\mathbb{Z}_{128}$ que têm a mesma representação binária.

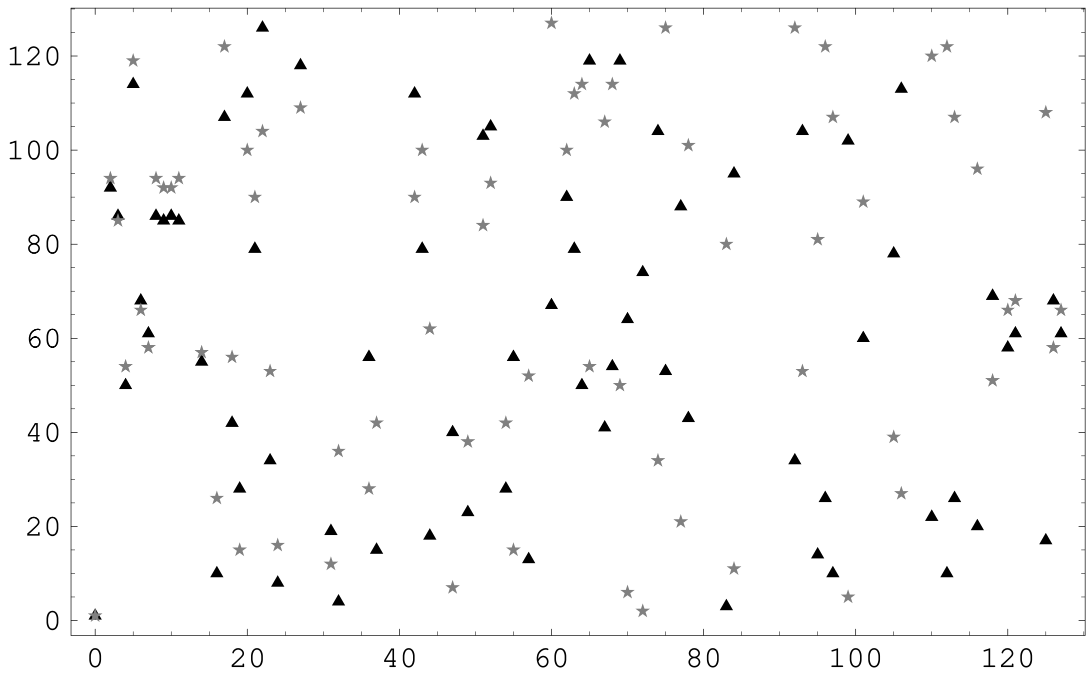


Figure 1: Curva elíptica $y^2 + xy = x^3 + x^2 + 1$ sobre $\mathsf{GF}(2^7)$

Sob o ponto de vista criptográfico é crucial entender como são gerados os pontos desta curva.

A operação algébrica essencial é a **adição** de pontos $P + Q$ que não tem aqui uma interpretação geométrica simples mas é algébricamente calculada de modo análogo ao usado em $\mathbb{R}$.

Com a adição define-se a **multiplicação escalar**

$$n\,P \;= \overbrace{P + P + \cdots + P}^{n \text{ vezes}}$$

Neste exemplo particular os pontos finitos da curva têm todos a forma

$$P = i\,S + n\,G \qquad i \in 0,1 \quad n \in \{1 \cdots 70\}$$

para um **gerador** apropriado G, e tem-se em atenção que $S = 71\,G$.

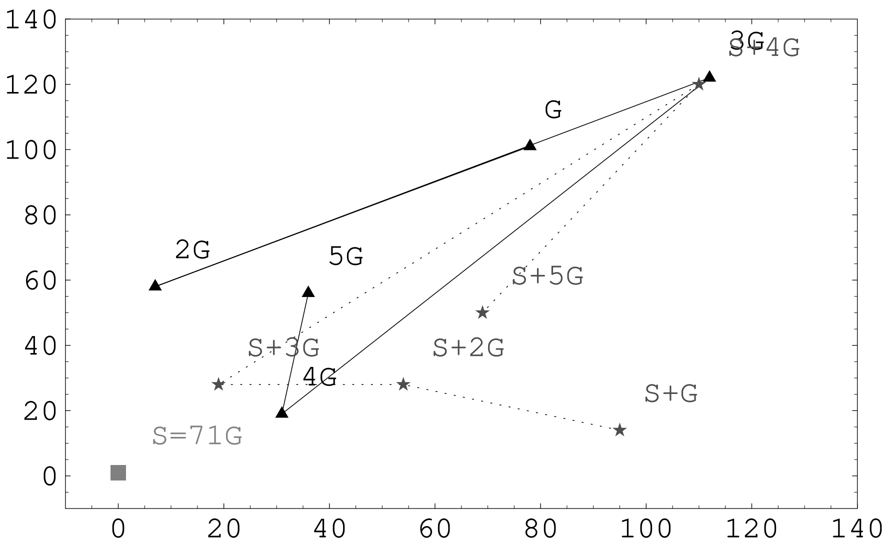


Figure 2: Pontos gerados por $G = \langle\, 78\,,\; 101 \,\rangle$

É notória a forma como a multiplicação escalar "espalha" os pontos no plano criando a necessária "confusão".

A caracterização mais importante da "confusão" reside na complexidade computacional da determinação do chamado **logaritmo discreto**:

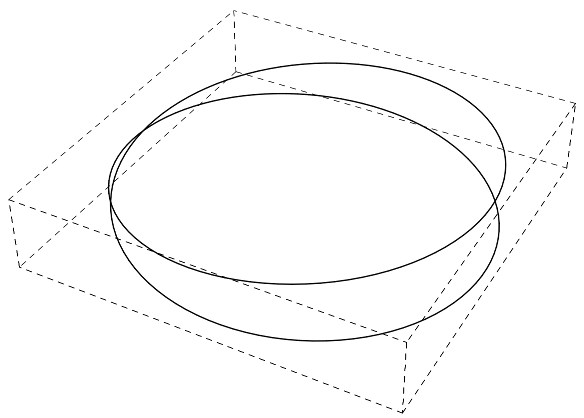
dado um ponto P *calcular o escalar* n *tal que* $n\,G = P$

Se a curva elíptica for convenientemente escolhida não existe solução para este problema melhor do que a procura exaustiva.

Esta será a propriedade essencial sobre a qual se vão basear as técnicas criptográficas sobre CE.

Note-se que os pontos se agrupam em dois "ciclos" de tamanho 71 que partilham o ponto singular e o ponto no infinito.

Nas figuras 1 marcá-mos com símbolos distintos (estrela e triângulo) os pontos dos dois ciclos distintos e na figura 2 estão indicados 5 pontos de cada "ciclo".



A dupla estrutura cíclica pode ser representada num diagrama (como o que se indica nesta figura) que nos dá uma visão geométrica alternativa da curva.

Como se faz Criptografia com Curvas Elípticas?

Curvas elípticas (*ver módulo III*) podem originar técnicas criptográficas análogas às que se baseiam nas propriedades do grupo cíclico $\mathbb{Z}_p^*$; para tal usam-se as analogias

Grupo cíclico $\mathbb{Z}_p^*$	**Curvas Elípticas**
multiplicação de inteiros $x \times y$ exponenciação $\alpha^a \pmod p$	soma de pontos $P + Q$ multiplicação escalar $k\,G$
unidade 1 elemento inverso $x^{-1} \pmod p$	ponto no infinito $\mathcal{O}$ ponto simétrico $-P$
ordem de α: $d(\alpha)$ é o menor $n > 0$ tal que $\alpha^n = 1 \pmod p$ ordem do grupo: i.e. número de elementos do grupo	ordem de G: $\#G$ é o menor $n > 0$ tal que $n\,G = \mathcal{O}$ ordem da curva: i.e. número de pontos sobre a curva
problema do logaritmo discreto dados α e β, achar $a < d(\alpha)$ tal que $\beta = \alpha^a \pmod p$	*problema do logaritmo em CE* dados G e P, achar $k < \#G$ tal que $P = k\,G$

Exemplo: *Protocolo Diffie-Hellman sobre Curvas Elípticas*

1. O TA escolhe uma curva $\mathcal{C}$ e um ponto G sobre a curva cuja ordem $\#G$ seja um primo q grande ($q > 2^{120}$).
2. O agente A gera informação privada $k \in \mathbb{Z}_q$ e envia $P = k\,G$ a B.
3. O agente B gera informação privada $l \in \mathbb{Z}_q$ e envia $Q = l\,G$ a A.
4. O agente A calcula $k\,Q$ e o agente B calcula $l\,P$; ambos os valores determinam o ponto $(k\,l\,G)$ que constitui o segredo partilhado por ambos.
5. Uma chave K pode ser obtida deste ponto usando, por exemplo, o valor de uma das suas coordenadas.

Complexidade das técnicas de CE

Dado que as entidades *curvas elípticas* são matematicamente muito mais complexas do que os inteiros, porque é que tem interesse desenvolver técnicas criptográficas sobre curvas elípticas?

1. Em termos computacionais a complexidade adicional das curvas elípticas é mais aparente do que real.

   De facto no início dos anos 90 as implementações existentes sugeriam desempenhos computacionais que não se podiam comparar com os algoritmos então conhecidos para os inteiros.

   Hoje em dia, atendendo à economia na representação binária (*ver ponto (2.)*) e à melhoria das técnicas numéricas, muitas técnicas criptográficas sobre curvas elípticas são computacionalmente mais eficientes do que as técnicas análogas sobre inteiros.

2. Para o mesmo nível de segurança, os objectos usados nas curvas elípticas exigem representações binárias muito mais curtas do que as representações de objectos análogos nas técnicas com inteiros.

   O tamanho n do corpo finito $\mathbf{F_n}$ sobre o qual se realiza uma curva elíptica desempenha um papel análogo ao tamanho do módulo m, nas técnicas PFI, ou ao tamanho do primo p nas técnicas PLD.

   É consensual julgar que uma dimensão de 160 bits para n é equivalente (em termos de segurança) a 1024 bits para m ou p.

Isto significa que as técnicas criptográficas baseadas em curvas elípticas ocupam muito menos memória e menos recursos computacionais. São, por isso, particularmente apropriadas a pequenos dispositivos móveis (*smart-cards*, *buttons*, telefones celulares, *palm tops*, etc.).

Viabilidade das técnicas de criptografia sobre CE

As técnicas criptográficas sobre CE realizam, essencialmente, todos os objectivos das técnicas sobre os grupos cíclicos de inteiros de forma tão ou mais eficiente. Será seguro, viável e prudente usá-las?

$\oplus$ O **problema do logaritmo sobre curvas elípticas (PLCE)** é computacionalmente tão ou mais intratável do que os problemas do logaritmo discreto ou da factorização.

As técnicas de resolução do PLCE mais eficientes são baseadas no método de ρ-Pollard e têm complexidade proporcional a $\sqrt{\#G}$. Compara-se o seu desempenho com o da mais eficiente técnica para o PFI (o *number sieve*) no seguinte quadro: [a]

Factorização, $m \sim 2^{1024}$	3×10^{11} MIPS ano
Logaritmo CE, $n \sim 2^{163}$	9.6×10^{11} MIPS ano

$\oplus$ A criptografia em CE não está sujeita a quaisquer restrições derivadas de patentes.[b]

De momento existem, publicamente disponíveis, várias bibliotecas com implementações de criptografia sobre CE que podem ser directamente usadas sem receio de se violar quaisquer direitos de propriedade.

$\ominus$ As técnicas matemáticas requeridas pela criptografia sobre CE são mais complexas e difíceis de dominar do que as análogas em inteiros.

No âmbito da nova proposta de *standard* para criptografia de chave pública, desenvolvida pelo IEEE e designada por IEEE P1363, as técnicas criptográficas em CE têm o papel central porque se acredita que vão constituir o núcleo essencial das técnicas criptográficas de chave pública do futuro imediato.

[a] 1MIPS ano $\equiv$ esforço computacional equivalente à execução de 10^6 instruções por segundo durante um ano.

[b] Existem alguns pedidos de patentes mas dificilmente poderão alguma vez cobrir as técnicas criptográficas essenciais.

Aritmética sobre Curvas Elípticas

Vamos considerar apenas curvas elípticas sobre corpos finitos $\mathbf{F}_n$ de característica 2. A sua forma mais geral é gerida pela equação[a]

$$y^2 + x\,y = x^3 + \mu\,x^2 + \upsilon \qquad \mu, \upsilon \in \mathbf{F}_n \quad \upsilon \neq 0 \tag{$*$}$$

O conjunto de pares $\langle\, x\,,\, y\,\rangle \in \mathbf{F}_n \times \mathbf{F}_n$ que satisfazem $(*)$ conjuntamente com o *ponto de infinito* $\mathcal{O}$, representado por $\langle\, 0\,,\, 0\,\rangle$, forma um grupo aditivo abeliano com as seguintes regras:

Complemento, duplicação, adição

$$-\langle\, x\,,\, y\,\rangle \doteq \langle\, x\,,\, x+y\,\rangle \qquad \langle\, x\,,\, y\,\rangle + \langle\, 0\,,\, 0\,\rangle \doteq \langle\, x\,,\, y\,\rangle$$

$$\langle\, x\,,\, y\,\rangle + \langle\, x\,,\, y\,\rangle \doteq \begin{cases} \langle\, 0\,,\, 0\,\rangle & \text{se} \quad x = 0 \wedge y \neq 0 \\ \langle\, \theta\,,\, x^2 + \theta(\lambda+1)\,\rangle & \text{se} \quad x \neq 0 \end{cases}$$

$$\text{com} \quad \lambda \equiv x^{-1}\,y + x \quad \text{e} \quad \theta \equiv \lambda^2 + \lambda + \mu$$

$$\langle\, x\,,\, y\,\rangle + \langle\, x'\,,\, y'\,\rangle \doteq$$

$$\begin{cases} \langle\, 0\,,\, 0\,\rangle & \text{se} \quad x = x' \wedge y \neq y' \\ \langle\, x + x' + \mu\,,\, y\,\rangle & \text{se} \quad x \neq x' \wedge y = y' \\ \langle\, x + x' + \theta\,,\, y + \lambda\,(\theta + x')\,\rangle & \text{se} \quad x \neq x' \wedge y \neq y' \end{cases}$$

$$\text{com} \quad \lambda \equiv (x+x')^{-1}(y+y') \quad \text{e} \quad \theta \equiv \lambda^2 + \lambda + \mu$$

[a] Existe uma forma alternativa $y^2 + y = x^3 + \mu\,x + \upsilon$, conhecida por *curva supersingular*, que tem a vantagem de ter uma aritmética mais simples mas a grande desvantagem de conduzir a PLCE que é criptográficamente inseguro. Por isso essa curva nunca deve ser usada em criptografia.

Notas

1. Como $\mathbf{F}_n$ tem característica 2, será $n = 2^m$ para um $m > 0$ apropriado. Portanto $\mathbf{F}_n$ é sempre o *Corpo de Galois* $\mathsf{GF}(2^m)$

2. Cada curva $\mathcal{C}$ é completamente determinada por três valores $\langle m, \mu, \upsilon \rangle$; o inteiro m é o número de bits necessários para representar as coordenadas $\langle x \, , \, y \rangle$ dos pontos da curva; $\mu, \upsilon \in \mathsf{GF}(2^m)$ são os parâmetros que determinam a equação dessa curva.

3. Se $P \equiv \langle x \, , \, x\, z \rangle$ é um ponto da curva, existe apenas um ponto, distinto de P, que tem a mesma coordenada x: esse ponto é $-P \equiv \langle x \, , \, x\,(z+1) \rangle$.

   Portanto, para determinar um ponto basta a coordenada x e um único bit da representação da segunda coordenada que permita escolher entre z e $z+1$. Como estes dois valores têm bits menos significativos distintos o "bit de escolha" será esse bit menos significativo.

   Note-se que, para $x \neq 0$ e fazendo $c = x + \mu + \upsilon\, x^{-2}$, temos $z = x^{-1}\, y$ como solução da **equação quadrática**

   $$z^2 + z = c \qquad (*)$$

   Assim, para recuperar a coordenada y a partir de x e do bit menos significativo, calcula-se c, determinam-se as duas soluções z da equação $(*)$, usa-se o bit para escolher a solução correcta e determina-se $y = x\, z$.

4. O cálculo da soma $P + Q$ ou da duplicação $2\,P$ de pontos em $\mathcal{C}$ exige, no pior dos casos, um cálculo de inversa, duas multiplicações e um ou dois cálculos de quadrados em $\mathsf{GF}(2^m)$.

   A soma em $\mathsf{GF}(2^m)$ é trivial mas a multiplicação, o quadrado e a inversa não o são. Se acrescentar-mos ainda o cálculo das soluções da equação $(*)$, que pode ser vista como análogo à determinação da raiz quadrada, temos as 4 operações aritméticas computacionalmente críticas em $\mathsf{GF}(2^m)$.

   A falta de algoritmos eficientes para computar o resultado destas operações limitou, durante bastante tempo, a prática das técnicas criptográficas sobre CE. Foi a descoberta de tais algoritmos que veio tornar viável o uso destas técnicas criptográficas.

Multiplicação escalar

Dado $P \equiv \langle\, x\,,\, y\,\rangle$ e um inteiro k, calcular $k\,P$.

Se $k = 0$ temos $k\,P = \mathcal{O}$; senão usa-se a representação binária de k

$$k \equiv b_0\, 2^N + b_1\, 2^{N-1} + \cdots + b_N, \qquad b_i \in \{0, 1\}$$
$$k\,P = 2\,(\cdots 2\,(2\,(2\,(b_0\,P) + (b_1\,P)) + (b_2\,P)) + \cdots + (b_N\,P)$$

Temos N duplicações e tantas somas $(b_i\,P) = P$ quantos os bits $b_i = 1$. Este algoritmo pode ainda ser melhorado de misturar-mos adições de P com adições de $-P$: por exemplo, $(2^N - 1)\,P = 2^N\,P - P$ tem N duplicações e uma subtracção em vez das $N - 1$ somas do método inicial.

Escolha de um ponto $P \not\equiv \mathcal{O}$ sobre a curva $\mathcal{C}$

Pretende-se gerar um ponto aleatório $\langle\, x\,,\, y\,\rangle$ sobre a curva $\mathcal{C} \equiv \langle m, \mu, \upsilon\rangle$.

(i) Gerar um $x \in \mathsf{GF}(2^m)$ aleatório.

(ii) Se for $x = 0$ escolhe-se $y = \upsilon^{2^{m-1}}$.[a]

(iii) Se for $x \neq 0$, calcula-se $c = x + \mu + \upsilon\, x^{-2}$ e procura-se uma solução da equação quadrática $z^2 + z = c$; se não existir voltar a (i).

(iv) Se existir uma solução z escolhe-se um dos dois valores $y = x\,z$ ou $y = x\,(z + 1)$.

Escolha de um ponto de ordem q sobre a curva $\mathcal{C}$

Supõe-se conhecida a ordem N da curva e r tal que $N = r\,q$.

(i) Escolhe-se P de acordo com o algoritmo anterior.

(ii) Se for $r\,P = \mathcal{O}$ voltar a (i). Senão usar P.

[a] Note-se que em $\mathsf{GF}(2^m)$ tem-se sempre $\upsilon^{2^m} = \upsilon$; donde a equação $y^2 = \upsilon$ tem sempre a solução $y = \upsilon^{2^{m-1}}$.

O cálculo de quadrados, multiplicações e inversas e a resolução da equação quadrática são os problemas essenciais de $\mathsf{GF}(2^m)$. Para os resolver é preciso começar por uma representação adequada.

Bases em $\mathsf{GF}(2^m)$

Base conjunto de m elementos $\{\beta_m, \cdots, \beta_2, \beta_1\}$ linearmente independentes; i.e. $a_1\,\beta_1 + \cdots + a_m\,\beta_m = 0$ se e só se $a_i = 0$ para todo $i \in 1..m$.

Escolhida a base, cada $x \in \mathsf{GF}(2^m)$ tem aí <u>uma única</u> representação sob a forma de uma *string* de bits $\bar{x} \equiv x_m \cdots x_2\,x_1$ tal que

$$x \;=\; x_m\,\beta_m + x_{m-1}\,\beta_{m-1} + \cdots + x_2\,\beta_2 + x_1\,\beta_1 \qquad \text{com} \quad x_i \in \mathbb{Z}_2$$

Seja $\mathbf{c}[X] \in \mathbb{Z}_2[X]$ um **polinómio primitivo**[a] de grau m

$$\mathbf{c}[X] \;=\; X^m + c_{m-1}\,X^{m-1} + \cdots + c_1\,X + 1 \qquad c_i \in \mathbb{Z}_2$$

e seja $\alpha \in \mathsf{GF}(2^m)$ uma raiz desse polinómio; i.e. $\mathbf{c}[\alpha] = 0$.

Para $i \in 1 \cdots m$ podem-se definir as seguintes classes de bases:

Base polinomial 0:	β_i	$=$	α^{i-1}
Base polinomial 1:	β_i	$=$	α^{i}
Base normal:	β_i	$=$	$\alpha^{2^{i-1}}$

Nas bases polinomiais os elementos α^i são sempre linearmente independentes. O mesmo já não é possível afirmar das bases normais.

Pode-se verificar que, se existir a base normal, <u>todos</u> os β_i são raízes do polinómio característico $\mathbf{c}[X]$; portanto existe uma base normal se e só se o polinómio característico tem m raízes linearmente independentes.

[a] Polinómio mónico que divide $X^{2^m} + X$ mas não divide qualquer outro polinómio $X^k + X$, com $k < 2^m$.

Exemplos de bases e representações em $\mathsf{GF}(2^4)$.

Consideremos $m = 4$ e o polinómio pleno $X^4+X^3+X^2+X+1$ que, para esta ordem 4, é primitivo e pode ser usado como polinómio característico.

Seja α, uma raíz do polinómio característico; i.e. $\alpha^4+\alpha^3+\alpha^2+\alpha+1 = 0$

Os elementos $\{\alpha^3, \alpha^2, \alpha, 1\}$ formam uma base polinomial 0. O elemento $x = \alpha^3 + 1$ tem a representação 1001 nesta base.

Os elementos $\{\alpha^4, \alpha^3, \alpha^2, \alpha\}$ formam uma base polinomial 1. Devido ao termo 1 o elemento x não parece representável na base; no entanto, como $1 = \alpha^4 + \alpha^3 + \alpha^3 + \alpha$, podemos escrever $x = \alpha^3 + (\alpha^4 + \alpha^3 + \alpha^2 + \alpha) = \alpha^4 + \alpha^2 + \alpha$. Logo x tem a representação 1011 nesta base.

Os elementos $\{\alpha^8, \alpha^4, \alpha^2, \alpha\}$ formam uma base normal. Facilmente se constrói a pequena tabela

1	$=$	$\alpha^8 + \alpha^4 + \alpha^2 + \alpha$
α^3	$=$	α^8

que permite converter de representações polinomiais para normais e vice-versa.

Por exemplo $x = \alpha^3 + 1 = \alpha^8 + (\alpha^8 + \alpha^4 + \alpha^2 + \alpha) = \alpha^4 + \alpha^2 + \alpha$. A representação de x na base normal será então 0111.

Cada classe de base tem as suas vantagens (que procuraremos ver em seguida) e existem sempre algoritmos para converter a representação de um elemento x numa das bases na sua representação noutra base.

Bases normais, quando existem, são particularmente adequadas para lidar com quadrados; note-se que, num corpo de característica 2 verifica-se $(a+b)^2 = a^2 + b^2$. Como consequência:

(1) *O cálculo de x^2 é muito simples*

Temos $x^2 = x_m\,\beta_m^2 + \cdots + x_1\,\beta_1^2$. Temos também $\beta_i^2 = \alpha^{2\times 2^{i-1}} = \alpha^{2^i}$. Se $i < m$ temos $\beta_i^2 = \beta_{i+1}$; se for $i = m$ temos $\beta_m^2 = \alpha^{2^m} = \alpha = \beta_1$. Donde

$$x^2 = x_{m-1}\,\beta_m + \cdots + x_1\,\beta_2 + x_m\,\beta_1$$

Pode-se calcular a representação de x^2 fazendo um *shift left rotate* na representação de x.

(2) *Os diversos elementos β_i da base normal são raízes distintas de $\mathbf{c}[X]$*

Este facto verifica-se por indução no índice i: $\beta_1 = \alpha$ que, por hipótese, é raiz de $\mathbf{c}[X]$. Temos $\mathbf{c}[\beta_i]^2 = \mathbf{c}[\beta_i^2] = \mathbf{c}[\beta_{i+1}]$; logo, se β_i é raiz também β_{i+1} é raiz.

(3) *A resolução da equação $z^2 + z + a = 0$ é muito simples.*

Esta equação tem soluções se e só o chamado **traço** de a, definido como $\mathsf{Tr}(a) = a + a^2 + a^4 + \cdots + a^{2^{m-1}}$ tem o valor 1. Numa base normal sabe-se que este traço se calcula facilmente $\mathsf{Tr}(a) = a_1 \oplus a_2 \oplus \cdots \oplus a_m$.

Do calculo dos quadrados temos $z_{i-1} \oplus z_i \oplus a_i = 0$ para $i > 1$. Daqui se constrói uma relação de recorrência que determina os diversos bits z_i. Consoante a sequência é iniciada com $z_1 = 0$ ou $z_1 = 1$ determina-se uma das duas raízes.

$$z_1 = 0 \quad \text{ou} \quad z_1 = 1 \qquad z_i = a_i \oplus z_{i-1} \quad i \in 2\cdots m$$

Operações em $\mathsf{GF}(2^m)$

Soma $x+y$

Em qualquer base, a representação de $x+y$ obtém-se somando, em $\mathbb{Z}_2$, as representações de x e y bit a bit.

$$\overline{x+y} = \overline{x} \oplus \overline{y}$$

Nota
Esta é uma operação muito eficiente que pode ser efectuada palavra a palavra, num único ciclo. Todas as restantes operações são efectuadas à custa de instruções básicas semelhantes e, nomedamente, operações de *shift*.

Símbolo	*Resultado*	*Descrição*
$\overline{x}$	$x_m \cdots x_2\, x_1$	–
$\uparrow\overline{x}$	$0\, x_m \cdots x_2$	*shift right*
$\Rsh\overline{x}$	$x_1\, x_m \cdots x_2$	*shift right rotate*
$\overline{x}\uparrow$	$x_m \vert x_{m-1} \cdots x_1\, 0$	*shift left carry*
$\overline{x}\Lsh$	$x_{m-1} \cdots x_1\, x_m$	*shift left rotate*

Conversão de base

Note-se que somar $\mathbf{c}[\alpha]$ a x não altera o valor de x (já que $\mathbf{c}[\alpha] = 0$) mas altera o arranjo de bits. Isto é aproveitado para as operações de conversão entre as duas formas de base polinomial.

Exemplo
Seja $m = 4$ e $\mathbf{c}[X] = X^4 + X + 1$; temos $\overline{\mathbf{c}[\alpha]} \equiv 10011$.

Se $x = \alpha^2 + 1$ então 0101 é a representação de x na base polinomial 0. Como $x + \mathbf{c}[\alpha] = \alpha^4 + \alpha^2 + \alpha$ a representação de x na base polinomial 1 será 1011.

Base polinomial 1 para base polinomial 0: $\overline{x}\uparrow \oplus x_m \cdot \overline{\mathbf{c}[\alpha]}$

Base polinomial 0 para base polinomial 1: $\uparrow(\overline{x} \oplus x_1 \cdot \overline{\mathbf{c}[\alpha]})$

Base normal óptima de Tipo I

Quando $m+1$ é primo e 2 é um gerador do grupo cíclico $\mathbb{Z}^*_{m+1}$ então $\mathsf{GF}(2^m)$ admite uma chamada **base normal óptima de Tipo I**.

15 FACTO *Nestas condições*

(i) O polinómio pleno $X^m + \cdots + X^2 + X + 1$ *é característico de* $\mathsf{GF}(2^m)$.

(ii) Para todo elemento β *da base* $\beta^k = \beta^l$ *se e só se* $k = l \pmod{m+1}$.

(iii) $\beta_m + \cdots + \beta_2 + \beta_1 = 1$

Temos $\alpha^{2^{i-1}} = \alpha^j$ se for $j = 2^{i-1} \pmod{m+1}$. Como 2 é gerador de $\mathbb{Z}^*_{m+1}$ esta igualdade estabelece uma *permutação de índices* $\pi : \{1 \cdots m\} \to \{1 \cdots m\}$ tal que $\pi(i) \equiv 2^{i-1} \pmod{m+1}$ e tal que o elemento da base normal $\alpha^{2^{i-1}}$ coincide com o elemento da base polinomial $\alpha^{\pi(i)}$.

Se os elementos da base são os mesmos (apenas apresentados por ordem diferente) a representação polinomial e a representação normal são permutações uma da outra.

Conversão da base normal para base polinomial 1:
O bit $x_{\pi(i)}$ da base polinomial é construído com o valor do bit x_i da base normal.

Conversão da base polinomial 1 para base normal:
O bit x_i da base normal tem o valor do bit $x_{\pi(i)}$ da base polinomial.

O modo simples como se realiza a mudança de base é aproveitado para o cálculo da multiplicação e dos inversos em bases normais $\beta_i = \alpha^{2^{i-1}}$.

16 FACTO *Sendo* $k = 2^{i-1} + 2^{j-1} \pmod{m+1}$ *tem-se*

$$\beta_i \times \beta_j = \begin{cases} \beta_{\pi^{-1}(k)} & se \quad k \neq 0 \\ \beta_m + \cdots + \beta_2 + \beta_1 & se \quad k = 0 \end{cases}$$

Base Normal Óptima de Tipo II

$\mathsf{GF}(2^m)$ aceita uma **base normal óptima de tipo II** quando $2\,m+1$ é primo e ocorre uma de duas situações:

(i) 2 é gerador de $\mathbb{Z}^*_{2m+1}$

(ii) $4|2(m+1)$ e 2 é gerador dos resíduos quadráticos em $\mathbb{Z}_{2m+1}$

17 FACTO *Se γ é qualquer elemento de ordem $2m+1$ em $\mathsf{GF}(2^{2m})$ então, para $i=0\cdots m-1$, os*

$$\beta_i = \gamma^{2^i} + \gamma^{-2^i}$$

formam uma base normal. Adicionalmente, para $i,j<m$,

(i) Se $2^i+2^j \neq 0 \pmod{2m+1}$ existe $k<m$ tal que $2^i+2^j = \pm 2^k \pmod{2m+1}$.

(ii) Se $2^i-2^j \neq 0 \pmod{2m+1}$ existe $l<m$ tal que $2^i-2^j = \pm 2^l \pmod{2m+1}$.

(iii) $\beta_i \times \beta_j = \begin{cases} \beta_k + \beta_l & \textit{se} \quad 2^i \pm 2^j \neq 0 \pmod{2m+1} \\ \beta_k & \textit{se} \quad 2^i - 2^j = 0 \pmod{2m+1} \\ \beta_l & \textit{se} \quad 2^i + 2^j = 0 \pmod{2m+1} \end{cases}$

Multiplicação na base polinomial 0

Se $z = \alpha^{m-1} z_m + \cdots + \alpha z_2 + z_1$ temos $\alpha \times z = \alpha^m z_m + \cdots + \alpha^1 z_1$: a representação polinomial 1 de $\alpha \times z$ coincide com a representação polinomial 0 de z. Usando esse facto e a conversão, $\overline{\alpha \times z} = \bar{z}\!\uparrow \oplus z_m \cdot \overline{\mathbf{c}[\alpha]}$, calcula-se $z = x \times y$ através da expansão

$$z = x \times y = x \times ((\alpha \cdots (\alpha y_m + y_{m-1}) + \cdots y_2) + y_1)$$

o que dá origem ao algoritmo seguinte

$z = y_m \cdot x$
para $i = m - 1$ até 1 fazer $\{z = \alpha \times z \; ; \; z = z + y_i \cdot x\,\}$
devolver z

Este processo pode ser optimizado pré-computando a expressão $F(\overline{x}, \overline{y})$ que determina o bit 1 de z em função das representações de x e y.

$u = \overline{x} \; ; \; v = \overline{y}$
para $i = 1$ até m fazer $\{z_i = F(u, v) \; ; \; u = u\!\curvearrowleft \; ; \; v = v\!\curvearrowleft\,\}$
devolver z

Multiplicação numa base normal óptima (Tipo I ou II)

Sendo $x = x_m \beta_m + \cdots + x_1 \beta_1$ e $y = y_m \beta_m + \cdots + y_1 \beta_1$ temos

$$z = x \times y \; = \; \sum_{i,j} x_i \cdot y_j \cdot (\beta_i \times \beta_j)$$

Usando $\beta_i \times \beta_j$ determinados no facto 16 (no Tipo I) ou no facto 17 (no Tipo II) calcula-se uma matriz $\mathbf{L}$ que determina o bit 1 de z, a partir das representações $\overline{x}$ e $\overline{y}$ como $z_1 = \overline{x}^T \mathbf{L}\, \overline{y}$. Daí constrói-se o algoritmo

$u = \overline{x} \; ; \; v = \overline{y}$
para $i = 1$ até m fazer $\{z_i = \overline{u}^T \mathbf{L}\, \overline{v} \; ; \; u = u\!\curvearrowleft \; ; \; v = v\!\curvearrowleft\,\}$
devolver z

Cálculo de a^{-1} em $\mathsf{GF}(2^m)$

O cálculo do elemento inverso é uma das operações críticas em $\mathsf{GF}(2^m)$. A melhor técnica genérica é baseada na relação $a^{2^m} = a$ que conduz a

$$a^{-1} = a^{2^m-2} = (a^2)^{(2^{m-1}-1)}$$

Este problema insere-se num outro problema mais geral: a determinação de

$$f(y,x,n) = y\,x^{2^n-1}$$

Conhecido um algoritmo que calcule $f(y,x,n)$, tem-se

$$a^{-1} = f(1,a^2,m-1) \qquad \text{e} \qquad a^{-1}\,b = f(b,a^2,m-1)$$

$f(y,x,n)$ pode ser calculado com o algoritmo usual da exponenciação; isso requer cerca de n quadrados e outras tantas multiplicações. Em alternativa o seguinte algoritmo requer $\log_2(n)$ operações.

1º caso: $n>0$ *é par.*
Temos $2^n-1 = (2^{n/2}+1)\,(2^{n/2}-1)$. Logo $x^{2^n-1} = (x^{2^{n/2}+1})^{2^{n/2}-1}$

$$f(y,x,n) = f(y,x\,x^{2^{n/2}},n/2)$$

Isto implica uma rotação[a] e uma multiplicação acompanhada de uma redução para metade do último argumento.

2º caso: $n>1$ *é ímpar.*
Temos $2^n-1 = (2^{(n+1)/2}+2)\,(2^{(n-1)/2}-1)+1$. Donde

$$f(x,y,n) = f(x\,y,x^2\,x^{2^{(n+1)/2}},(n-1)/2)$$

Isto implica duas rotações e duas multiplicações acompanhada de uma redução de n para metade.

3º caso: $n=0$ ou $n=1$

$$f(y,x,0) = y \qquad\qquad f(y,x,1) = y\,x$$

[a]Tomando como referência a representação normal, o cálculo de x^{2^n} é essencialmente independente de n: é feito com uma operação *left shift rotate* que toma n como argumento e comanda o número de posições rodadas.

Escolha de Curva Elíptica e de Gerador

Em qualquer técnica criptográfica sobre CE o problema inicial é sempre a selecção da curva $\mathcal{C}$, escolhendo o corpo $\mathsf{GF}(2^m)$ e os parâmetros μ e υ, e a determinação de um ponto gerador $G \in \mathcal{C}$ com as propriedades de segurança pretendidas.

A propriedade essencial é a **ordem da curva** $N_{\mathcal{C}}$ (o número de pontos da curva) e a **ordem do gerador** $\#G$; para efeitos de segurança ambos têm de ser tão grandes quanto possível.

Factos

1. $N_{\mathcal{C}}$ é par e é múltiplo de todo $\#G$.
2. Verifica-se $(N_{\mathcal{C}} - (2^m + 1))^2 \leq 2^{m+2}$ (*limites de Hasse*).
3. Se m é pequeno a ordem da curva é calculada directamente por
   (i) Seja $s = (-1)^{\mathsf{Tr}(\mu)}$ e $\sigma(x) = \mathsf{Tr}(x + \upsilon\, x^{-2})$.
   (ii) $N_{\mathcal{C}} = 2^m + 1 + s\, \sum_{x \neq 0} (-1)^{\sigma(x)}$
4. Se $m = k\, n$ e a ordem da curva N em $\mathsf{GF}(2^n)$ é conhecida então a ordem $N_{\mathcal{C}}$ em $\mathsf{GF}(2^m)$ é o elemento V_k da sequência determinada pela relação de recorrência

$$V_0 = 2 \qquad V_1 = 2^n + 1 - N \qquad V_i = V_1\, V_{i-1} - 2^n\, V_{i-2} \quad i \geq 2$$

Se m é divisível por um inteiro n suficientemente pequeno para ser viável usar (3.) então (4.) determina a ordem da curva em $\mathsf{GF}(2^m)$. Pode-se usar este processo para escolher uma curva $\mathcal{C}$ (os parâmetros μ e υ) de modo que a ordem seja *"quase"* primo; i.e. seja da forma $t\, q$ em que q é primo e t é um pequeno par.

Multiplicação Complexa

Recentemente apareceram as chamadas técnicas de *Multiplicação Complexa* que permitem escolher curvas com uma ordem dentro de limites pré-estabelecidos. A complexidade matemática destas técnicas não permite que sejam apresentadas neste estudo básico.

Os algoritmos básicos, no entanto, são apresentados no anexo A da norma IEEE 1363.

Ordem de um Gerador

Se a ordem da curva $N_{\mathcal{C}}$ é conhecida e se q é o maior factor primo dessa ordem é simples determinar um gerador G de ordem q.

(i) Determina-se o *co-factor* de q como $t = N_{\mathcal{C}}/q$ e gera-se um ponto aleatório P que verifique $t\,P \neq \mathcal{O}$.

(ii) Faz-se $G = t\,P$.

A determinação da ordem de uma CE é, assim, um processo computacionalmente complexo o que conduz a uma de duas estratégias na escolha de uma curva:

(I) Estão pré-computadas um determinado número de curvas para as quais foi pré-determinado a ordem. O utilizador escolhe uma dessas curvas e o seu grau de liberdade resume-se à escolha do gerador.

m	μ	v	**base**	**ordem**
101	1	1	polin	$2 \times q$
109	1	1	polin	$2 \times q$
113	1	1	bno-II	$2 \times q$
163	1	1	polin	$2 \times q$
131	1	1	bno-II	$4 \times q$
134	0	γ	bno-II	$4 \times q$
158	0	γ	bno-II	$4 \times q$
$\gamma(\gamma+1) = 1$				
148	γ^*	1	bno-I	$18 \times q$
$\gamma \in \mathbb{Z}_2^4 \quad \mathsf{Tr}(\gamma) = 1$				

Esta tabela resume algumas curvas vulgarmente usadas indicando a forma da sua ordem e o tipo de base que aceitam; q denota o maior factor primo da ordem.

Apenas a última aceita uma base normal óptima de tipo I. As dimensões 113, 131, 134 e 158 admitem bases normais óptimas de tipo II. Para $m = 148$ μ tem a forma de uma repetição 37 vezes de um padrão de 4 bits $\gamma \in \mathbb{Z}_2^4$ de paridade 1.

$m = 131$ e $m = 163$ são casos importantes porque originam curvas seguras (melhores que ataques exaustivos a 64 e 80 bits, respectivamente) que cobrem os dois valores típicos de tamanho de *hash*: 128 e 160 bits.

(II) É possível escolher uma curva aleatória (gerando parâmetros aleatórios μ e v) e escolher uma dimensão de corpo que admita, por exemplo, uma base normal óptima de tipo I.

Pode-se usar heurísticas para estimar a ordem de um ponto dessa curva e, se essas heurísticas conseguirem prever ordens criptográficamente seguras, será possível usar estes geradores.

Porém, nesse, não há garantia de qual é a ordem da curva e/ou a ordem do ponto. Algumas técnicas criptográficas exigem o conhecimento explícito desses valores o que inviabiliza este tipo de escolha de curva.

Técnicas criptográficas baseadas no PLDCE

Dada a analogia entre o problema do logaritmo discreto em $\mathbb{Z}_p^*$ e o mesmo problema em curvas elípticas, quase todas as técnicas criptográficas baseadas o PLD dão origem a uma técnica análoga em CE.

Existem no entanto algumas diferenças e algumas técnicas novas. Procuraremos indicar essas analogias e essas diferenças.

Cifras assimétricas

El Gamal

A versão EC da cifra El Gamal não requer o conhecimento *a priori* da ordem da curva elíptica e, por isso, pode ser usado com curvas aleatórias.

Parâmetros públicos: uma curva elíptica aleatória $\mathcal{C}(m, \mu, v)$ criptográficamente segura e um ponto $G \in \mathcal{C}$ de ordem q, sendo q um primo suficientemente grande..

Chave privada de B: um inteiro aleatório $0 < k < q$.

Chave pública de B: $P_B = k\,G$

Cifragem: o agente A realiza

(i) Codifica a mensagem x num ponto da curva P_x

(ii) Gera um $0 < \omega < q$ aleatório e calcula $H = \omega\, G$ e $Q = P_x + \omega\, P_B$

(iii) Envia o par $\langle\, H\,,\, Q\,\rangle$ para B.

Decifragem: o agente B recupera P_x calculando $Q - k\,H$.

Correcção note-se que $k\,H = k\,(\omega\,G) = \omega\,(k\,G) = \omega\,P_B$. Portanto $Q - k\,H = (P_x + \omega\,P_B) - \omega\,P_B = P_x$.

ECES (*Elliptic Curve Encryption Scheme*)

Uma cifra muito simples que recorre às EC apenas para um acordo de chaves DH.

Parâmetros públicos: os mesmos que na cifra El Gamal em CE.

Chave privada e pública de B: as mesmas que na cifra El Gamal.

Cifragem: Para cifrar uma mensagem x, o agente A realiza

(i) Gera $0 < \omega < q$ e calcula $H = \omega\,G$.

(ii) Calcula $S = \omega\,P_B$ e extrai, das coordenada desse ponto, uma *string* de bits s.

(iii) Usa s como chave de uma cifra simétrica para obter $y = \{x\}_s$ e envia $\langle\, H\,,\, y\,\rangle$ a B.

Decifragem: O agente B recupera S calculando $k\,H$. Do ponto S extrai a *string* de bits s e, com o criptograma $\{x\}_s$, recupera x.

Notas

(1) A correcção desta cifra é análoga à do protocolo DH: i.e., o ponto S é calculado por B como $k\,H = k\,\omega\,G = \omega(k\,G) = \omega\,P_B = S$.

(2) Uma cifra frequentemente utilizada é simplesmente $y = s \oplus x$. Também é frequente usar-se a string de bits s como chave de uma cifra sequencial: se o tamanho da mensagem x for comparável com o tamanho de s a cifra sequencial pode ser relativamente simples.

Protocolos de Acordo de Chaves

Para além do protocolo Diffie-Hellman já referido, as curvas elípticas facilitam o desenvolvimento de protocolos multi-passo, com autenticação dos agentes (de modo a contrariar os ataques de *man-in-the-middle*), que exploram as suas propriedades específicas.

Protocolo MQV (*Menezes-Qu-Vanstone*).

Parâmetros Públicos Genéricos: uma curva elíptica $\mathcal{C}(m, \mu, v)$ criptográficamente segura e de ordem N conhecida. Um gerador G de ordem $\#G = q$ sendo q um divisor primo de N suficientemente grande.

Chaves Privadas: A e B têm, respectivamente, chaves privada $k \in \mathbb{Z}_q$ e $k' \in \mathbb{Z}_q$.

Chaves Públicas: A tem chave pública $P = k\,G$ e B tem chave pública $P' = k'\,G$.

Designa-se por a a primeira coordenada do ponto P e por a' a primeira coordenada do ponto P' .

Protocolo:

(1) A gera $\omega \in \mathbb{Z}_q$, calcula $Q = \omega\,G$ e envia-o a B . Identifica-se por x a primeira coordenada de Q .

(2) B gera $\omega' \in \mathbb{Z}_q$, calcula $Q' = \omega'\,G$ e envia-o a B . Identifica-se por x' a primeira coordenada de Q' .

(3) A calcula

$$s = \omega + x\,a\,k \pmod N \qquad U = Q' + x'\,a'\,P' \qquad W = s\,U$$

(4) B calcula

$$s' = \omega' + x'\,a'\,k' \pmod N \qquad U' = Q + x\,a\,P \qquad W' = s'\,U'$$

Resultado: verifica-se $W = W'$ que é o segredo partilhado por A e B.

Notas

(1) Neste como em muitos outros protocolos existem dois tipos de chave pública. Os pontos P e P' são **chaves públicas permanentes** (sobrevivem à execução do protocolo) enquanto que Q e Q' são **chaves públicas efémeras** (o seu tempo de vida é uma execução do protocolo).

(2) **Correcção**
Pela definição de s, tem-se $s\,G = (\omega\,G) + x\,a\,(k\,G) = Q + x\,a\,P = U'$. Do mesmo modo se conclui $s'\,G = U$. Portanto

$$W = s\,U = s\,(s'\,G) \qquad W' = s'\,U' = s'\,(s\,G) \quad \Rightarrow \quad W = W'$$

(3) **Segurança**
Devido às chaves privadas efémeras ω e ω' um intruso não pode reutilizar uma chave acordada numa comunicação anterior: mesmo que consiga obter uma dessas chaves privadas não a pode reutilizar. O valor s é uma *chave privada de sessão* que só pode ser construída conhecendo a chave privada efémera ω mas também a chave privada permanente k. Este facto impede uma substituição de chave pública e, por isso, o ataque de *man-in-the middle.*

Protocolo MO (*Massey-Omura*)

Este é um protocolo de distribuição de chaves em que A escolhe um segredo e no final do protocolo B conhece o segredo.

O protocolo não é sensível a ataques de intrusos porque usa extensivamente a confirmação de comunicações.

Neste caso não existe propriamente informação privada permanente; toda a informação privada é efémera.

Parâmetros Públicos Genéricos: os mesmos que no protocolo MQV.

Protocolo:

(1) A gera $a \in \mathbb{Z}_N^*$ e codifica o segredo x num ponto P_x. Calcula $Q_1 = a\,P_x$ e envia-o a B.

(2) B gera $b \in \mathbb{Z}_N^*$, calcula $H_1 = b\,Q_1$ e envia-o a A.

(3) A calcula $Q_2 = a^{-1}\,H_1$ e envia-o a B

(4) B calcula $H_2 = b^{-1}\,Q_2$.

Resultado: Verifica-se $H_2 = P_x$; portanto, B recupera x de H_2.

Notas

1. **Correcção**
   Tem-se
   $$H_2 = b^{-1}\,Q_2 = b^{-1}(a^{-1}\,H_1) = b^{-1}\,a^{-1}\,(b\,Q_1) = b^{-1}\,a^{-1}\,b\,(a\,P_x) = P_x$$
   devido à comutatividade em $\mathbb{Z}_N^*$.

2. Neste protocolo B não consegue identificar a proveniência do segredo x. No uma vez iniciado o protocolo por um agente A não é possível a um intruso I substituir o segredo x por um "falso segredo" x' já que para tal teria de conhecer a^{-1} e, portanto, o valor de a que iniciou o protocolo.

Assinaturas Digitais em Curvas Elípticas

O *standard* IEEE define dois esquemas de assinaturas digitais baseados em curvas elípticas e que são análogos a dois esquemas bem conhecidos nos inteiros: o DSA e o esquema de Nyberg-Rueppel.

Os esquemas são semelhantes e baseiam-se ambos na mistura de operações em CE com operações em inteiros que representam coordenadas de pontos nessas CE.

Assinaturas com recuperação de mensagem Nyberg-Rueppel

Parâmetros públicos genéricos: Uma curva elíptica $\mathcal{C}(m,\mu,\upsilon)$, criptográficamente segura, de ordem N conhecida. Um ponto gerador G cuja ordem $\#G = q$ é um primo suficientemente grande.

Chaves privada e pública de A: a chave privada é um inteiro $k \in \mathbb{Z}_q$ e a chave pública é $P = k\,G$.

Geração de assinatura: para assinar um texto x o agente A realiza:

(1) Gera $\omega \in \mathbb{Z}_q$, calcula $H = \omega\, G$ e determina h como a 1ª componente de H convertida num inteiro.

(2) Calcula $\gamma = h + x \pmod N$ e $s = \omega - k\,\gamma \pmod N$.

(3) A assinatura é o par $\langle\, \gamma\,,\, s\,\rangle$.

Verificação da assinatura: o agente B realiza

(1) Calcula $H' = s\,G + \gamma\,P$ e recupera h' como a primeira componente de H' convertida em inteiro.

(2) O texto x é recuperado por $x = \gamma - h' \pmod N$.

Aceitação: Como em todos os esquemas *arm* a assinatura é aceite se o valor recuperado de x coincide com o do texto x previamente conhecido.

Notas

1. Para verificar-mos a correcção deste esquema note-se que, com $s = \omega - k\,\gamma \pmod{N}$, o ponto H' calculado por B é

$$\begin{aligned} H' &= s\,G + \gamma\,P \;=\; (\omega - k\,\gamma)\,G + \gamma\,P \;=\; \omega\,G - \gamma\,(k\,G) + \gamma\,P \;= \\ &= H - \gamma\,P + \gamma\,P \;=\; H \end{aligned}$$

   Portanto as primeiras componentes h e h' (de H e H', respectivamente) são iguais e teremos

$$\gamma - h' \pmod{N} \;=\; \gamma - h \pmod{N} \;=\; x$$

2. Como é usual nos esquemas de assinatura baseadad no PLD, a geração da assinatura requer o conhecimento da chave privada permanente k conjuntamente com a chave privada efémera ω. O conhecimento de uma por um intruso facilmente lhe permite determinar a outra.

3. Um ponto importante em todos os esquemas de assinaturas é o tamanho dos dados manipulados na geração de assinatura e a complecidade computacional que envolve.

   A assinatura é um par $\langle \gamma\,,\,s \rangle$ em que cada uma das componentes é da ordem de grandeza da ordem da curva N. Como esta é da mesma ordem de grandeza que a dimensão m do corpo de finito de suporte pode-se concluir que γ e s têm a dimensão semelhante a m: ~ 160 bits.

   A geração da assinatura envolve uma multiplicação escalar e operações de adição e multiplicação sobre inteiros de 160 bits. São, assim, operações computacionalmente simples

Esquema ECDSA (*Elliptic Curve Digital Signature Algorithm*)

Parâmetros públicos e chave privada: os mesmos que no esquema de Nyberg-Rueppel.

Geração da assinatura: Para assinar um texto x o agente A realiza:

(1) Gera $\omega \in \mathbb{Z}_N^*$, calcula $H = \omega\, G$ e identifica h como a 1ª componente de H convertida num inteiro.

(2) Calcula $\gamma = h$ e $s = \omega^{-1}(x + k\, h) \pmod{N}$.

(3) Se $\gcd(s, N) \neq 1$ volta a (1); senão a assinatura é o par $\langle\, \gamma\, ,\, s\, \rangle$.

Verificação da assinatura: O agente B determina:

(1) Calcula $\eta_1 = s^{-1}\, x \pmod{N}$ e $\eta_2 = s^{-1}\, \gamma \pmod{N}$.

(2) Determina $H' = \eta_1\, G + \eta_2\, P$ e identifica h' como a primeira componente de H' convertida em inteiro.

(3) A assinatura é aceite se $h = h' \pmod{N}$.

Notas

1. Para analisarmos a correcção deste esquema basta notar que

$$H' \;=\; \eta_1\, G + \eta_2\, P \;=\; (\eta_1 + k\, \eta_2)\, G \;=\; s^{-1}(x + k\, \gamma)\, G \;=\; \omega\, G \;=\; H$$

   Sendo $H = H'$ teremos forçosamente $h = h'$.

2. A dimensão dos dos dados processados na geração da assinatura é análoga ao esquema Nyberg-Rueppele e ao DSA ($\sim$ 160 bits). No entanto a computação é mais complexa não só porque envolve o cálculo de uma inversa como, porque a verificação da assinatura exige um s invertível, pode ser necessário repetir várias vezes o cálculo de H , h e s até se achar um s invertível.