Criptografia

Módulo I – Terminologia

M. B. Barbosa
mbb@di.uminho.pt

Departamento de Informática
Universidade do Minho

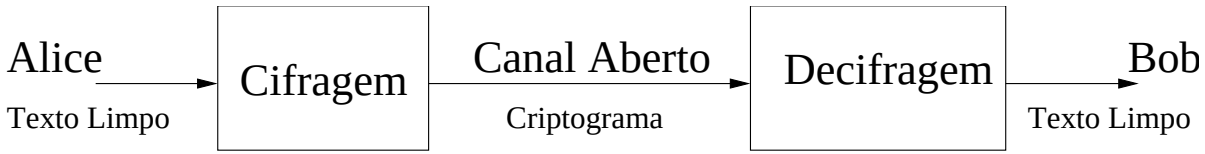
2006/2007

Segurança da Informação

- A criptografia está intimamente ligada ao conceito de **segurança da informação**.
- Dependendo do contexto, a segurança da informação pode implicar diferentes requisitos:
  - privacidade/confidencialidade, integridade e anonimato.
  - autenticação/identificação de entidades ou da origem de mensagens.
  - autorização, validação, controlo de acessos e propriedade.
  - certificação, revogação e *timestamping*.
  - recibos/confirmações da recepção de mensagens ou execuções de serviços.
- O objectivo da criptografia é garantir que os intervenientes na troca de informação tenham garantias de que os requisitos de segurança foram satisfeitos.

- Desde há muito que este tipo de objectivos são satisfeitos através dos mais variados mecanismos e, na maior parte dos casos, sem recorrer a técnicas criptográficas.
- A maneira de representar e armazenar a informação não se alterou muito nos últimos tempos.
- O que mudou dramaticamente foi a facilidade e rapidez com que é possível copiar e alterar informação de forma indetectável e indistinguivel.
- Note-se que a satisfação de muitos destes objectivos exige muito mais do que apenas primitivas criptográficas.
- São também necessários protocolos e estruturas de suporte complexas, bem como legislação adequada.

Comunicação segura entre agentes

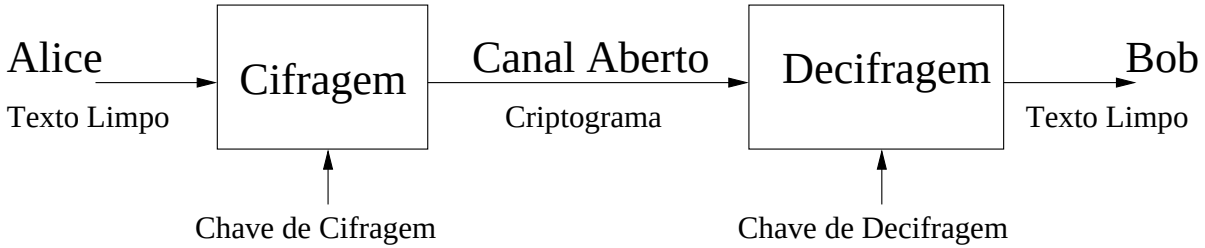


Os conceitos fundamentais neste contexto, além da **Confidencialidade**, são:

- **Autenticação**. B tem a garantia de que a mensagem provém de A.
- **Integridade**. B tem a garantia de que a mensagem que recebeu foi aquela que A enviou, sem alterações.
- **Não repúdio**. O emissor não pode, mais tarde, negar que enviou a mensagem.

Cifras e Chaves

- Uma **cifra** é um algoritmo criptográfico, i.e. uma função matemática injectiva, que efectua as transformações entre o **texto limpo** e o **criptograma**.
- Tradicionalmente, os detalhes das cifras eram secretos: era neste princípio que residia a sua segurança.
- Na criptografia moderna, a segurança de uma cifra não se consegue tornando secreto o seu funcionamento.
- A qualidade de uma cifra mede-se pelo tempo que permanece inquebrável, sendo do conhecimento geral.
- Uma vez que o seu funcionamento interno é do conhecimento geral, o seu comportamento é controlado por uma **chave**.



- No que diz respeito às chaves que utiliza, uma cifra diz-se:
  - **Simétrica**. As chaves de cifragem e decifragem são iguais. Os interlocutores partilham uma mesma chave, que tem de ser previamente acordada e mantida secreta. Neste caso a chave chama-se **chave secreta**.
  - **Assimétrica**. As chaves de cifragem e decifragem são diferentes. Apenas a chave de decifragem precisa de ser secreta, e apenas o receptor a pode conhecer. Um intruso pode conhecer a chave de cifragem, sem que isso comprometa a segurança da cifra.

Criptoanálise de uma Cifra

- A criptoanálise é feita de "ataques" que podem ser classificados como:
  - **Criptograma conhecido**. O intruso conhece um ou vários criptogramas trocados entre emissor e receptor.
  - **Texto limpo conhecido**. O intruso conhece um ou vários criptogramas, e os textos limpos correspondentes, trocados entre emissor e receptor.
  - **Texto limpo escolhido**. Implica convencer o emissor a cifrar um determinado texto limpo ou, no caso de ser **adaptativo** implica acesso à *máquina* de cifragem.
  - **Criptograma escolhido**. Implica acesso à *máquina* de decifragem.
  - **Outros...** Violência, extorsão, etc.
- O objectivo é recuperar o texto limpo para um novo criptograma ou, mais dificilmente, a própria chave.

Criptoanálise de um Protocolo

- Para atacar um protocolo criptográfico mais complexo, é mais fácil tentar subverter as regras do jogo:
  - **Chave conhecida**. O intruso utiliza chaves utilizadas em instâncias anteriores do protocolo.
  - **Repetição**. O intruso armazena mensagens numa execução do protocolo e repete-as mais tarde tentando replicar o processo.
  - **Personificação**. O intruso tenta assumir a identidade de um participante legitimo.
  - **Dicionário**. Geralmente um ataque a *passwords*, consiste em tentar uma lista de palavras prováveis.
  - **Code book**. O intruso constrói uma colecção de pares texto limpo/criptograma que vai crescendo em observações sucessivas do protocolo.

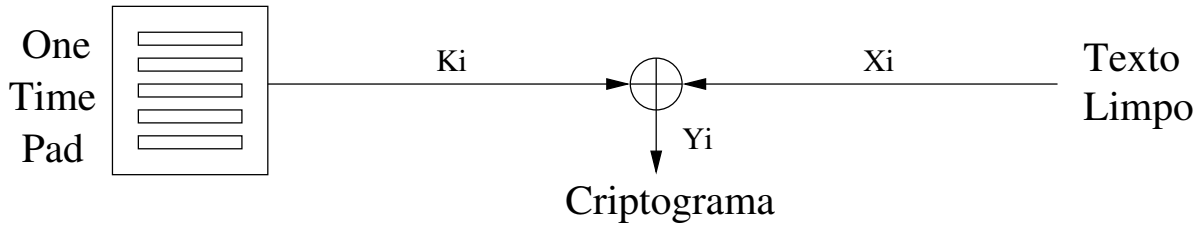
Segurança de uma cifra

- A segurança de uma cifra pode ser avaliada de acordo com a complexidade inerente a um ataque bem sucedido:
  - ao nível da **quantidade de dados** necessária.
  - ao nível do **processamento** necessário.
  - ao nível do **espaço de armazenamento** necessário.
- A complexidade expressa-se em ordens de magnitude e.g. uma complexidade de processamento de 2^{128} significa que é este o número de operações necessárias para quebrar a cifra.
- Note-se que um milhão de computadores a executar um milhão de operações por segundo demorariam um milhar de milhões de vezes a idade do universo a quebrar uma cifra deste tipo.

- Uma cifra diz-se **incondicionalmente segura** se o número de criptogramas conhecidos nunca for suficiente para quebrar a cifra.
- Este é um conceito introduzido por Shannon com base na teoria da informação e descreve uma cifra em que os criptogramas não revelam qualquer informação adicional sobre o texto limpo.
- Esta é uma noção probabilística: a incerteza de um adversário sobre um texto limpo permanece inalterada, mesmo se este observar um ou mais criptogramas.
- Todas as cifras que não tenham esta propriedade são quebráveis por pelo menos um ataque: o **ataque por força bruta**.

- O ataque por força bruta consiste em tentar todas as chaves possíveis até se encontrar a correcta.
- Em geral, um ataque deste tipo não é viável na prática devido ao enorme número de possibilidades para o valor da chave.
- Por exemplo: uma cifra que utiliza uma chave de 128 bits. O número de chaves possíveis é 2^{128}.
- Um ataque por força bruta necessitaria, em média, de 2^{127} decifragens para ser bem sucedido.

One-Time-Pad



- A chave no one-time-pad é uma sequência de símbolos aleatória, e do mesmo tamanho do texto limpo.
- Cada símbolo do texto limpo é combinado com um símbolo da chave gerando um símbolo do criptograma.
- É **incondicionalmente segura**. Porquê?
- A chave só é utilizada uma vez, até porque é recuperável por um ataque de texto limpo conhecido.
- Problemas práticos: geração e distribuição da chave.

Mistura e Difusão

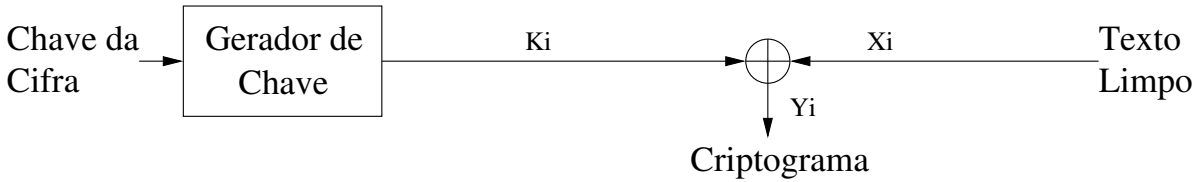
- Segundo Shannon, as duas técnicas básicas que permitem tornar um texto limpo num criptograma são as seguintes:
  - **Mistura (Confusion)**. Consiste em obscurecer a relação que existe entre o texto limpo, o criptograma e a chave. Na sua forma mais simples, implementa-se através de uma substituição.
  - **Difusão**. É a técnica de espalhar as redundâncias do texto limpo por todo o criptograma. Na sua forma mais simples isto é feito através de uma permutação.

Construção de Cifras

- Os *inputs* (e os *outputs*) das cifras são invariavelmente sequências de símbolos de um determinado alfabeto: letras, números ou sequências de bits.
- Evolução das transformações de cifragem:
  - **Monoalfabética**. Cada símbolo do texto limpo é transformado da mesma forma e.g. cifra de César. São susceptíveis a ataques por análise de frequências.
  - **Polialfabética**. Numa tentativa de mascarar os padrões, são aplicadas transformações distintas a símbolos sucessivos e.g. cifra de Vigenere.
  - **Poligrama**. Os símbolos são agrupados, e vários símbolos de texto limpo influenciam cada símbolo do criptograma.
- Idealmente alterando um símbolo do texto limpo altera com igual probabilidade todos os símbolos do criptograma.

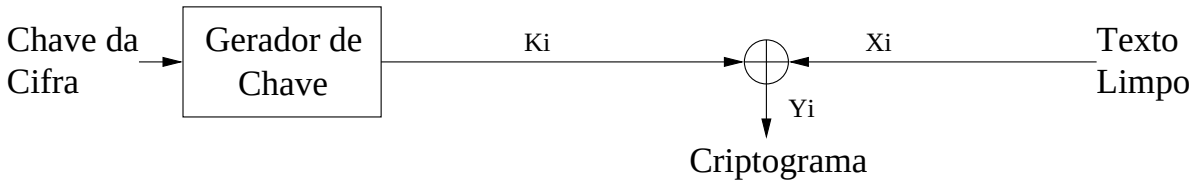
- Cumprir à risca este princípio é inviável na prática:
  - Em muitas aplicações não se dispõe de todo o texto limpo na altura em que se pretende iniciar a cifragem.
  - Mesmo que isto aconteça, impor este critério sobre um número arbitrário de bits de texto limpo é impraticável devido à complexidade computacional que isso implicaria.
- As cifras modernas podem ser vistas como aproximações a dois conceitos ideais:
  - Cifras Sequenciais – aproximações ao One-Time-Pad em que a sequência de chaves é apenas pseudo-aleatória.
  - Cifras por Blocos – poligramas optimizados para blocos de tamanho fixo, por exemplo 128 bits ou 256 bits.

Cifras Sequenciais



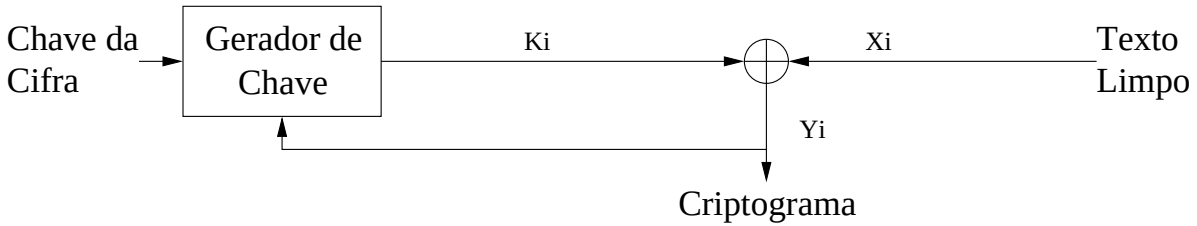
- Estas cifras operam sobre *streams* de texto limpo, um bit ou um byte de cada vez, combinando-o com uma stream de chaves gerada internamente.
- O esquema de decifragem é idêntico ao de cifragem. A sequência de chaves tem de ser reproduzida exactamente.
- A chave da cifra funciona como semente do gerador.
- A segurança da cifra reside na dificuldade de prever a sequência de valores gerados sem saber a chave da cifra.
- O período do gerador deverá ser o **maior possível**

Cifras Sequenciais Síncronas



- O stream de chaves é gerado independentemente dos dados cifrados.
- Robusto, uma vez que não há propagação de erros quando os bits são recebidos de forma errada. Porém, um atacante que saiba quais os bits que pretende alterar, pode não ser detectado.
- Problemas de sincronização quando se perdem ou inserem bits. No entanto, permite detectar ataques grosseiros de repetição ou supressão de bits.

Cifras Sequenciais Auto-sincronizáveis



- O stream de chaves é calculado com base no criptograma já gerado: permite recuperar perdas sincronismo.
- Tipicamente, o texto-limpo é precedido por uma sequência aleatória que garante que o sincronismo já foi alcançado no início da transmissão.
- Estas cifras têm o problema de propagarem erros que ocorram no criptograma, mas são menos susceptíveis a ataques activos. Em certas condições é possível a realização de ataques por repetição.

Exemplo: RC4

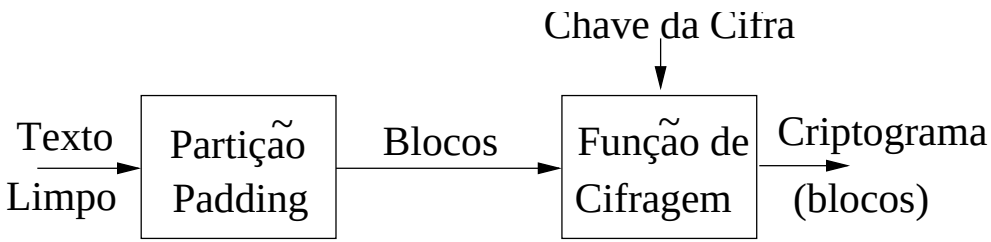
- Cifra de chave de tamanho variável de desenvolvida em 1987 para a empresa RSA , detentora da sua patente.
- Inicialmente foi mantida secreta através de acordos de sigilo. Em 1994 “alguém” publicou os detalhes da cifra na internet . . .
- O algoritmo trabalha de forma síncrona e gera um byte K_{ij} utilizando dois contadores i e j, da seguinte forma:
  $i = i + 1 \pmod{256}$; $j = j + S_i \pmod{256}$;
  $swap(S_i, S_j)$; $t = S_i + S_j \pmod{256}$;
  $K_{ij} = S_t$;
- Os valores S_i representam um array (*S-box*) de 256 posições, com uma permutação dos valores de 0 a 255, permutação essa gerada a partir da chave da cifra.

- O preenchimento da S-box começa pela geração de dois arrays de bytes S e K, de 256 bytes cada um:
  - S é preenchido sequencialmente com valores de 0 a 256.
  - K é preenchido com os bytes da chave, repetida as vezes necessárias.
- Depois executa-se o seguinte algoritmo de "randomize":

  $j = 0$;
  for i=0 **to** 255
  $j = (j + S_i + K_i) \pmod{256}$
  $swap(S_i, S_j)$

- Uma utilização pouco cuidada desta cifra levou à vulnerabilidade descoberta no standard WEP.

Cifras por Blocos



- A função de cifragem é um poligrama, e trabalha sobre blocos de tamanho fixo (tradicionalmente 64 bits, mas hoje em dia 128 ou mais bits).
- Associadas à função de cifragem, são necessárias unidades de partição e *padding* (enchimento), para produzir blocos de tamanho apropriado.
- Na decifragem, a função inversa da função de cifragem é aplicada aos blocos do criptograma. Os processos de *padding* e partição são revertidos no final.

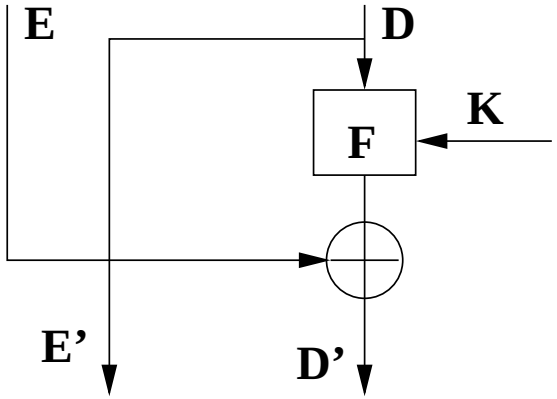
- As cifras por blocos são fundamentais aos sistemas criptográficos. Não só para garantir confidencialidade, mas também na construção de funções de hash, geradores de números pseudo-aleatórios, MACs, etc.
- As cifras por blocos devem ter as seguintes características:
  - Cada bit do criptograma deve depender de todos os bits da chave e de todos os bits do texto limpo.
  - Não deve haver nenhuma relação estatística evidente entre os bits do texto limpo e os bits do criptograma.
  - A alteração de um bit do texto limpo ou da chave deve alterar cada bit do criptograma com probabilidade $1/2$.
  - A alteração de um bit do criptograma deve originar uma alteração imprevisível no texto decifrado.

Exemplo: Data Encryption Standard (DES)

- O DES é um standard mundial há 25 anos, sendo muito utilizado em aplicações bancárias até hoje.
- Apesar de hoje em dia apresentar alguns sinais de velhice, essencialmente devido ao pequeno tamanho da chave, resistiu admiravelmente a anos de criptoanálise intensiva.
- Foi desenvolvido pela IBM e estudado pela NSA antes de ser adoptado como standard. Este processo foi alvo de alguma polémica.
- O tamanho dos blocos é de 64 bits. O tamanho da chave é de 56 bits, sendo armazenada em 8 bytes para incluir bits de paridade.

- O algoritmo consiste na aplicação sucessiva (16 vezes) ao texto limpo de uma operação de cifragem: um **round**.
- Para cada round é derivada uma chave de round através de uma permutação da chave da cifra. Este processo chama-se **key schedule**.
- Um round do DES é construido com base num **Circuito de Feistel**, o que permite implementações muito eficientes: o mecanismo de cifragem serve também para decifragem.
- De facto, no DES, estas operações são idênticas, tirando pequenos pormenores ao nível do key schedule (como a operação é inversa, a ordem de aplicação das chaves de round é também inversa).

- Um circuito de Feistel tem a seguinte estrutura:



- Cada bloco de texto limpo é partido em duas metades de 32 bits cada (E e D). Uma delas aparece sem alterações no final do round, pelo que são necessários dois rounds para cifrar todos os bits do bloco.
- A metade que é cifrada passa por uma função de Feistel (F), onde é permutada, combinada com a chave de round, e passada por uma função não linear (implementada através de S-boxes pré definidas).

Exemplo: IDEA

- Apareceu nos anos 90, e assenta o seu funcionamento em bases teóricas sólidas. Durante alguns anos foi a melhor cifra por blocos no mercado.
- Não substituiu o DES por dois motivos: é patenteado e é necessária uma licença; não foi criado com esse objectivo e não passou pelo escrutínio desejável.
- O tamanho do bloco é de 64 bits, mas a chave é já de 128 bits. O mesmo algoritmo é utilizado para cifragem e decifragem, permitindo implementações eficientes.
- Baseia-se na partição do bloco em quatro partes, e na aplicação sucessiva de operações algébricas (XOR, adição módulo 2^{16} e multiplicação módulo $2^{16}+1$) a essas parcelas e aos bits da chave.

Ex: Advanced Encryption Standard (AES)

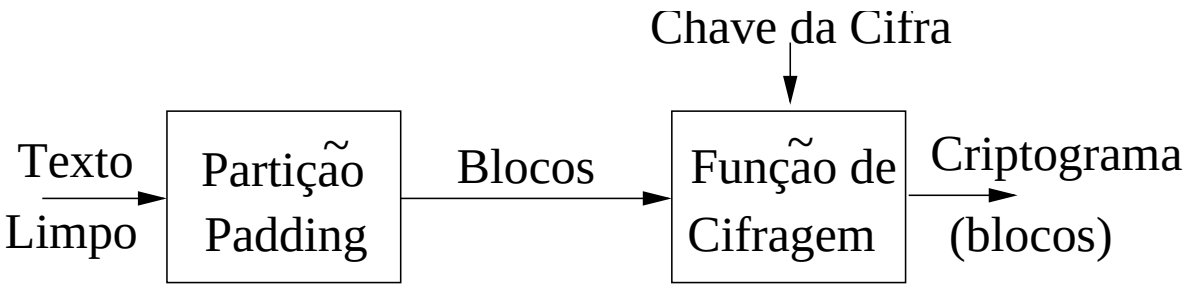
- Foi escolhido recentemente, como resultado de um concurso, para subtituir o DES. O algoritmo chama-se **Rijndael**, desenvolvido numa universidade Belga.
- O Rijndael é uma cifra com tamanho de bloco variável ($N_b \times 32$ bits, N_b =4,6 e 8) e com tamanho de chave variável ($N_k \times 32$ bits, N_k =4,6 ou 8). O AES fixa $N_b = 4$.
- Cada round é uma **rede de substituição-permutação**. O número de rounds depende de N_b e N_k.
- Utiliza operações algébricas de forma inteligente: a segurança é conseguida através de operações complexas, mas de implementação muito eficiente tanto em software como em hardware.

- O funcionamento do AES baseia-se no processamento de um estado de 4×4 bytes.
- Em cada round o estado passa pela mesma sequência de operações:
  - `AddRoundKey` Cada byte é combinado com a chave do round, derivada do algoritmo de *key schedule*.
  - `SubBytes` Operação não linear em que cada byte é substituído por um valor constante de uma *S*-box.
  - `ShiftRows` As linhas da matriz são rodadas de forma diferente (no AES os shifts são 0, -1, 2 e 3 para a direita).
  - `MixColumns` Cada coluna é sujeita a uma transformação linear invertível. Esta operação é equivalente a uma multiplicação de matrizes.
- As duas primeiras operações visam principalmente a **mistura**. As duas últimas a **difusão**.

Cifras Por Blocos: Modos de Funcionamento

- As cifras por blocos podem ser utilizadas de diversas formas, chamadas **modos**. Os critérios que levam ao aparecimento destas variantes são:
  - Segurança.
  - Eficiência.
  - Propagação de erros/Tolerância aos erros.
  - Adequação do “throughput” do modo de funcionamento (dependente do tamanho dos blocos) às necessidades da aplicação.
- Os modos de utilização mais comuns são:
  - Electronic Code Book Mode
  - Cypher Block Chaining Mode
  - Cypher Feedback Mode
  - Output Feedback Mode

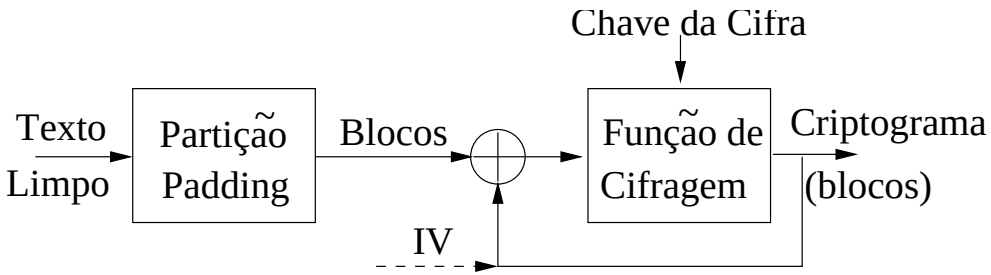
Electronic Code Book Mode (ECB)



- Segurança
  - Os padrões existentes no texto limpo não são disfarçados.
  - Um bloco cifrado duas vezes com a mesma chave resulta em criptogramas iguais.
  - Susceptível a ataques por *code book* (compilação de pares texto limpo/criptograma).
  - Susceptível a ataques por remoção, troca e repetição de blocos.

- Eficiência
  - Permite o acesso aleatório a dados cifrados.
  - Qualquer bloco pode ser decifrado independentemente.
  - Pela mesma razão, permite o processamento paralelo da informação.
  - Não há possibilidade de efectuar pré-processamento.
- Tolerância aos erros
  - Este modo não apresenta problemas de propagação de erros entre blocos.
  - Um erro afecta apenas um bloco de texto limpo.
  - Erros de sincronização (perda de bits) são irrecuperáveis.

Cipher Block Chaining Mode (CBC)



- O primeiro bloco é combinado com um **Vector de Inicialização** (IV), típicamente aleatório. O IV não tem de ser secreto e tem de estar disponível na decifragem.
- Segurança
  - Os padrões do texto limpo são mascarados pelo XOR.
  - De textos limpos iguais passam a resultar criptogramas distintos: impede ataques por code book e por repetição.
  - Ataques por manipulação de blocos são detectáveis.

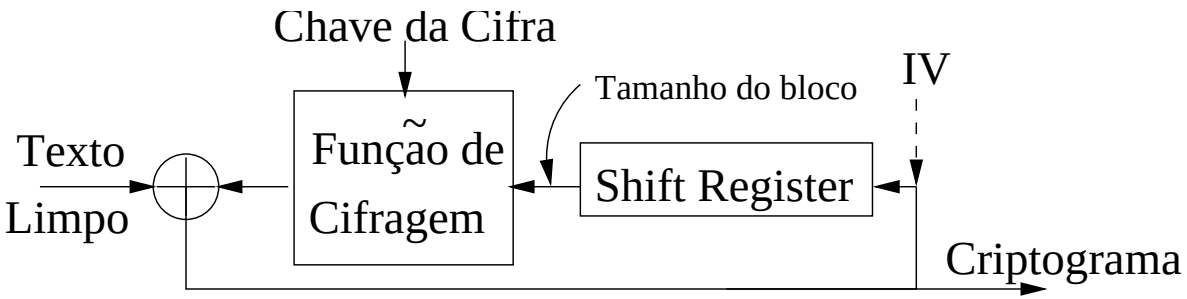
- Eficiência
  - Qualquer bloco pode ser decifrado independentemente, desde que se conheça o bloco anterior.
  - Pela mesma razão, permite o processamento paralelo da informação cifrada (não aplicável na cifragem). No entanto, uma alteração ao texto limpo, e.g. num ficheiro, implica uma nova cifragem completa.
  - Permite o acesso aleatório a dados cifrados.
  - Não há possibilidade de efectuar pré-processamento.
- Tolerância aos erros
  - Um erro num bit do criptograma afecta o bloco de texto limpo correspondente, e um bit no bloco seguinte.
  - Erros de sincronização (perda de bits) são irrecuperáveis.

Cifras por Blocos vs Cifras Sequenciais

- As cifras por blocos são cifras mais genéricas, vocacionadas para implementação em software, e utilizadas na maioria das aplicações criptográficas.
- As cifras sequenciais são vocacionadas para hardware, e utilização em canais de comunicação de alta velocidade, nos quais os dados são transferidos como *streams*.
- Uma cifra sequêncial permite cifrar informação à medida que ela vai sendo gerada e.g. toques num teclado.
- Nas cifras por blocos é necessário um determinado número de bits de texto limpo para fazer uma cifragem.
- Uma alternativa é efectuar o padding, mas isto agrava a utilização de um canal de comunicação.

- Além disso, uma utilização mais segura das cifras por blocos (CBC) implica cifrar todos os blocos de texto limpo numa só cifragem.
- O núcleo de uma cifra por blocos pode também ser utilizado para construir uma cifra sequencial, em aplicações onde uma cifra deste tipo seja mais apropriada.
- O Cipher Feedback Mode e o Output Feedback Mode servem este fim.
- A grande diferença é que **a função de cifragem passa a ser utilizada como gerador de chaves**.
- Nestes modos de funcionamento, os dados podem ser cifrados em parcelas inferiores ao tamanho do bloco.

Cipher Feedback Mode (CFB)

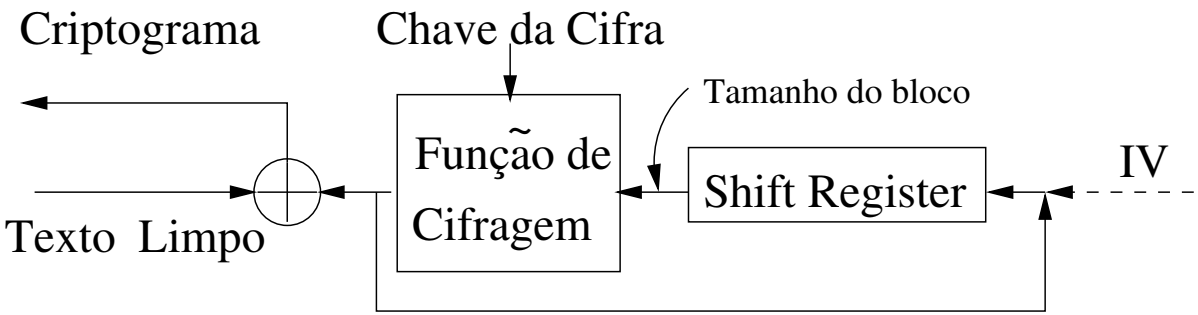


- A cifragem é efectuada exactamente como nas cifras sequenciais. A sequência de bits da chave é retirada da saída da função de cifragem.
- Necessáriamente, a função de cifragem opera sobre blocos de um tamanho pré-determinado. Esses blocos são obtidos de um shift-register.
- O shift-register está inicialmente preenchido com um IV. Os bits do criptograma vão substituindo os bits do IV.

- O IV deve ser aleatório, para originar sequências de chaves e criptogramas diferentes em cifragens sucessivas. Não precisa de ser secreto.
- A função de cifragem é usada da mesma forma na cifragem e decifragem. Na realidade, a função inversa não é necessária neste modo de funcionamento.
- Segurança
  - Os padrões do texto limpo são mascarados pela pseudo-aleatoriedade da sequência de chaves.
  - A alteração do IV é determinante. Como em todas as cifras sequenciais, a repetição de uma sequência de chaves torna a cifra vulnerável a ataques.

- Eficiência
  - Qualquer bit pode ser decifrado independentemente, desde que se conheça um número suficiente de bits anteriores do criptograma. Permite o acesso aleatório a dados cifrados.
  - Pela mesma razão, permite o processamento paralelo da informação cifrada (não aplicável na cifragem).
  - Há possibilidade de efectuar algum pré-processamento dos bits da chave.
- Tolerância aos erros
  - Erros de sincronização (perda de bits) são recuperáveis em determinadas condições (quais?): é uma cifra auto-sincronizável.
  - Um erro no criptograma tem como efeito imediato uma decifragem errada do bit de texto limpo correspondente. Enquanto o bit errado estiver no shift-register, o sistema vai debitar lixo.

Output Feedback Mode (OFB)



- O OFB corresponde a uma cifra sequencial síncrona.
- O shift-register e o IV cumprem a mesma função que no CFB. No entanto, o shift-register é alimentado pelos bits da chave já utilizados.
- Segurança
  - Os padrões do texto limpo são mascarados pela pseudo-aleatoriedade da sequência de chaves.
  - Como no caso anterior, a alteração do IV é determinante.

- Eficiência
  - Não faz sentido falar de processamento paralelo, uma vez que a sequência de chaves não depende do criptograma.
  - É possível efectuar a geração de chaves antecipadamente, pelo que a cifragem pode tornar-se muito eficiente.
- Tolerância aos erros
  - Neste modo não há propagação de erros. Um erro no criptograma afecta apenas um bit no texto limpo.
  - Erros de sincronização não são recuperáveis.

Escolha de um modo de funcionamento

- O ECB é muito utilizado para cifrar pequenas parcelas de informação aleatórias, e.g. chaves. Para este tipo de informação as falhas de segurança deste modo não são relevantes.
- O CBC é o modo de funcionamento recomendado para aplicações genéricas. É muito utilizado para cifrar ficheiros, onde os erros são pouco frequentes. É a melhor escolha para aplicações baseadas em software.
- O CFB e o OFB servem para aplicações onde é necessária uma cifra sequencial.
- O OFB é preferido quando o meio de transmissão introduz muitos erros.

Aproximação Teórica

- Um algoritmo de cifra por blocos é visto como uma família de funções ou permutações, de tamanho finito.
- A chave da cifra é um índice para esta família de funções.
- A segurança de um algoritmo de cifra simétrica é expressa como a máxima vantagem de um adversário no jogo:
  - O adversário tem acesso a um oráculo que recebe textos limpos e os transforma.
  - O oráculo poderá ser de dois tipos: (1) retorna um conjunto de bits aleatório ou (2) retorna um criptograma gerado com uma chave pré-gerada aleatóriamente.
  - O adversário tem de ser capaz de decidir de que tipo de oráculo se trata.
- O valor concreto da segurança do algoritmo é estimado por técnicas de criptoanálise.

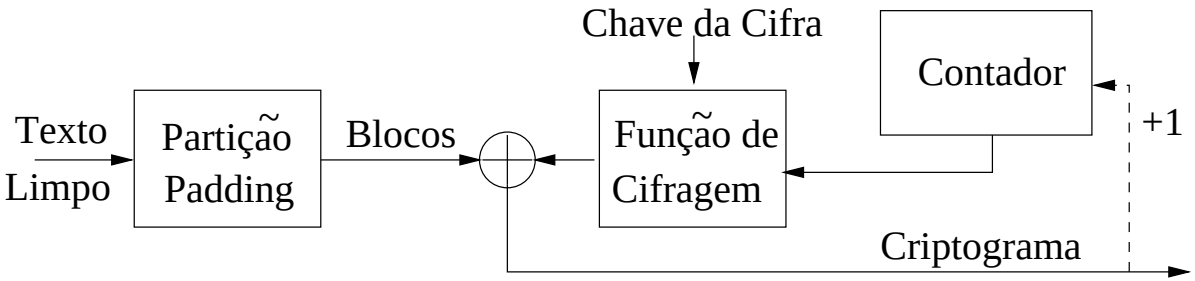
Objectivos de segurança

- Os modos de funcionamento são analisados com base num modelo para mensagens de tamanho arbitrário.
- O modelo mais utilizado para cifras simétricas chama-se Left Or Right Indistinguishability (LOR).
- A segurança neste modelo é expressa como a máxima vantagem de um adversário no seguinte jogo:
  - O adversário tem acesso a um oráculo que recebe um par de textos limpos (x_0, x_1), e retorna um criptograma.
  - O criptograma é gerado com base no texto limpo x_b e numa chave k. b e k são pré-gerados aleatóriamente.
  - O adversário tem de adivinhar o valor de b.
- A segurança de um modo de funcionamento é expressa em função da segurança do algoritmo de cifra simétrica.

Modelos de ataque

- O jogo anterior exprime o objectivo de segurança LOR para um atacante que apenas tem acesso a criptogramas: ataques por criptograma conhecido.
- Para reflectir cenários de ataques mais realistas, o modelo tem de ser extendido:
  - **Texto limpo escolhido (Chosen Plaintext Attack – CPA)** O adversário tem também acesso a um oráculo que efectua cifragens para textos limpos arbitrários com a chave k. (Restrições?)
  - **Criptograma escolhido (Chosen Ciphertext Attack – CCA)** Além do anterior, o adversário obtém um oráculo que decifra criptogramas arbitrários. (Restrições?)
- O modo de funcionamento CBC fornece boas garantias de segurança no modelo LOR-CPA.

Counting Mode



- Modo de funcionamento alternativo que, em termos teóricos dá melhores garantias no cenário CPA.
- Os blocos de texto limpo não passam pela função de cifragem, mas sim o valor de um contador.
- O contador é inicializado, e.g. a zero, e é incrementado imediatamente antes da cifragem de cada bloco.
- Para um mesmo valor da chave, o valor do contador nunca é reinicializado.

Padding

- Alguns modos de funcionamento requerem que o texto limpo tenha tamanho múltiplo do tamanho do bloco.
- Quando isto não acontece, é necessário introduzir *padding*, estendendo o texto limpo utilizando uma das seguintes variantes:
  1. Caracteres de espaço em branco.
  2. Caracteres nulos.
  3. Caracteres nulos, excepto o último que contém o número de bytes de padding.
  4. Bytes contendo todos o número de bytes de padding (PKCS#5).
  5. Começar com o byte 0x80, seguido de um número adequado de bytes nulos.

- Os dois primeiros métodos podem ser utilizados quando o texto limpo é um string ASCII, e não inclui espaços/caracteres nulos no fim. Caso contrário, surge uma ambiguidade.
- O terceiro método, desde que se convencione que o *padding* é sempre introduzido, qualquer que seja o tamanho do último bloco, pode ser utilizado sem qualquer ambiguidade.
- O quarto método é o recomendado na norma PKCS#5, e é o mais utilizado. Também aqui se introduz *padding* qualquer que seja o tamanho do último bloco.
- O último método é mais comum na comunidade dos smartcards. Também aqui o byte 0x80 é sempre incluído para evitar ambiguidades.

- Um aspecto importante dos últimos três métodos é a possibilidade de verificar a decifragem do último bloco.
- Uma alternativa, quando se usa o modo ECB, é utilizar uma variante do método #3, em que em vez de caracteres nulos, se utilizam bytes aleatórios. Desta forma consegue evitar-se que duas cifragens sucessivas do mesmo texto limpo resultem no mesmo criptograma.
- Uma outra situação importante é o caso em que o tamanho do texto limpo pode revelar informação crítica. Neste caso é possível adicionar um número aleatório de bytes aleatórios de *padding*.
- Normalmente, para evitar ambiguidades, utiliza-se sempre padding e, na maior parte dos casos, o método #4, da norma PKCS#5.

Comunicação Utilizando Cifras Simétricas

- A comunicação utilizando cifras simétricas exige os seguintes passos:
  1. A Alice e o Bob combinam uma cifra.
  2. A Alice e o Bob acordam uma chave.
  3. A Alice cifra a mensagem e envia-a ao Bob.
  4. O Bob decifra a mensagem.
- Como a criptoanálise de uma cifra é difícil, os ataques centram-se muitas vezes nos dois primeiros passos.
- É por este motivo que a gestão de chaves é um problema muito importante em Criptografia. A chave tem de permanecer secreta antes, durante e depois.
- A aproximação mais directa é a geração e distribuição prévia das chaves secretas necessárias.

Pré-distribuição de Chaves Secretas

- A principal desvantagem é o enorme número de chaves que é necessário gerar, armazenar e proteger: para N agentes, serão $N(N-1)/2$ chaves.
- Uma forma de racionalizar a utilização de recursos é transformar um dos agentes num servidor central de chaves secreta e responsável por armazenar todas as chaves secretas existentes.
- Isto traz diversas vantagens imediatas: a redução do espaço necessário para o armazenamento, a centralização da gestão das chaves, a redução da exposição das chaves, etc.
- Problema: como disponibilizar as chaves aos outros agentes quando são necessárias?

Chaves de Sessão

- Esta aproximação tem outras limitações que é impossível transpor:
  - uma vez que as chaves permanecem válidas durante a actividade do sistema elas estão expostas durante um tempo prolongado.
  - se uma chave for corrompida, e isso não for detectado, o canal entre esse par de agentes deixa de ser seguro.
- O conceito de **chave de sessão** resolve estes problemas:
  - É gerada uma chave secreta para cada comunicação.
  - Esta chave é estabelecida entre emissor e receptor, utilizada naquele instante, e destruída.
- O estabelecimento das chaves de sessão pode ser feito por: **distribuição de chaves**, e **acordo de chaves**.

Distribuição de Chaves

- Um dos agentes é responsável por gerar a chave de sessão. O problema a resolver é a transmissão da chave de sessão para o interlocutor através de um canal seguro.
- Isto coloca diversas questões: como é que se implementa um canal seguro? Como é que se obtém a garantia de que se está efectivamente a falar com o interlocutor?
- Em geral a solução adoptada consiste na introdução no sistema de um **Agente de Confiança** ou Trusted Agent (TA), e na adopção de uma classificação das chaves secretas em dois tipos: **chaves para cifragem de chaves** e **chaves para cifragem de dados**.
- **Nota:** Na comunicação agente/TA podem também ser usadas cifras assimétricas.

- As **chaves para cifragem de chaves** são geradas e pré-distribuidas por um canal seguro exterior ao sistema (e.g. em mão num CD, etc.).
- N chaves deste tipo permitem ao TA comunicar com cada agente de forma segura.
- As **chaves para cifragem de dados** correspondem às chaves de sessão. São geradas por um dos interlocutores ou pelo TA e transmitidas, através do TA, utilizando as chaves anteriores.
- Como grande desvantagem, este esquema tem a necessidade de pré-distribuir, armazenar e proteger as N chaves para cifragem de chaves.
- O Kerberos é um exemplo de um protocolo comercial baseado nesta filosofia.

Acordo de Chaves

- Os protocolos de acordo de chaves são mecanismos que derivam da criptografia assimétrica:
  - Dois agentes A e B pretendem comunicar utilizando uma cifra simétrica e, logo, estabelecer uma chave de sessão.
  - Existe um canal inseguro entre os dois agentes.
  - Os dois agentes seguem um conjunto de passos que lhes permitem identificar-se mutuamente.
  - Os dois agentes cooperam para gerar uma chave de sessão trocando informação através do canal aberto. Na composição da chave entra informação aleatória gerada por ambas as partes: garante a **frescura da chave**.
  - Os dois agentes executam um conjunto de passos que lhes permitem confirmar que foi estabelecida uma chave de sessão adequada.

- Não pode circular no canal aberto informação que permita a um intruso adivinhar a chave secreta acordada.
- Como é que isto se consegue? O segredo está na utilização de funções matemáticas com características especiais: as chamadas **funções one-way**.
- O que se consegue na prática é eliminar a necessidade de se armazenar informação secreta, bem como a necessidade da presença de um Agente de Confiança durante o funcionamento do sistema. (Em geral é necessário um TA para garantir as questões de identificação. Voltaremos a este assunto mais tarde.).
- O SSL é um exemplo de um protocolos comercial que recorre a este tipo de tecnologia.

Exemplo: Diffie-Hellman

- Foi inventado em 1976 e abriu o caminho para a criptografia de chave pública. É considerado o primeiro algoritmo desta categoria.
- É exclusivamente um protocolo de acordo de chaves. Não pode ser utilizado para cifragem e decifragem.
- A sua segurança baseia-se na dificuldade em resolver o problema do logaritmo discreto em corpos finitos:
  - Considere-se o conjunto $\{x \in Z : 0 \leq x < 11\}$, sobre o qual se definem a multiplicação e a adição módulo 11.
  - É fácil calcular a exponencial $3^4 \pmod{11} = 81 \pmod{11} = 4$.
  - No entanto, não há nenhuma forma eficiente de calcular o inverso: $\log_3 4 \pmod{11}$.

Protocolo Diffie-Hellman

1. A Alice e o Bob acordam os parâmetros públicos do protocolo:
   - um número primo grande n, e
   - um número primo g, menor que n (e com algumas outras restrições que garantem a segurança do protocolo).
2. A Alice gera um número aleatório grande x.
3. A Alice calcula $X = g^x \pmod{n}$ e envia-o ao Bob.
4. O Bob gera um número aleatório grande y.
5. O Bob calcula $Y = g^y \pmod{n}$ e envia-o à Alice.
6. Ambos conseguem calcular $K = X^y \pmod{n} = Y^x \pmod{n}$.

Funções One-Way

- A ideia central à Criptografia de Chave Pública é a de uma função One-Way: fácil de calcular, mas muito difícil (de preferência impraticável) de inverter.
- Um exemplo muito simples de uma função one-way é a exponenciação: é fácil de calcular y^x, mas obter $log_y(y^x)$ é muito mais complicado.
- Este conceito só foi aproveitado para a criptografia (pelo menos no mundo académico) nos anos 70. Foi descoberto por Diffie e Hellman. Qual a dificuldade?
  - Não há muitas funções deste tipo por aí.
  - Suponhamos que temos uma função impossível de inverter. Se a utilizamos para cifrar informação, isso significa que nunca será possível recupera-la!

Funções One-Way com Trapdoor

- Em criptografia usa-se um subconjunto das funções one-way a que se chama funções **one-way com trapdoor**.
- Estas funções oferecem uma porta de entrada secreta que permite a sua inversão de forma facilitada:
  - Como qualquer função one-way, $f(x)$ é fácil de calcular.
  - Como qualquer função one-way, $f^{-1}(f(x))$ é difícil de calcular.
  - Se for conhecido o segredo k, $f^{-1}(f(x), k)$ é fácil de calcular.
- A aplicação deste tipo de função à criptografia é imediata: qualquer agente que conheça a função f pode cifrar informação. No entanto, apenas um agente que conheça o segredo k pode efectuar a decifragem.

Chaves Públicas e Chaves Privadas

- A função de cifragem é parametrizada com uma chave: $f_K(x)$ é uma família de funções indexada por uma chave K.
- A chave de cifragem K, bem como a função $f(x)$, podem ser do conhecimento geral: o seu conhecimento não afecta a segurança de um criptograma. K é a **Chave Pública** da cifra.
- Associado a cada Chave Pública está um segredo k que dá o acesso à porta secreta da função $f_K(x)$: a **Chave Privada** da cifra. Esta não pode ser calculável a partir dos parâmetros públicos da cifra.
- Uma cifra assimétrica consiste portanto na função $f(x)$, e num par de chaves (K, k), em que K é a Chave Pública e k é a Chave Privada.

Exemplo: RSA

- Surgiu no final dos anos 70 e ainda sobrevive ao escrutínio da comunidade científica.
- O funcionamento do RSA é muito simples: a sua segurança relaciona-se com a dificuldade de factorizar um número inteiro grande nos seus factores primos.
- Para se gerar o par de chaves escolhem-se, inicialmente, dois números primos grandes p e q, e calcula-se o **módulo** $m = p \times q$.
- A chave privada k é escolhida aleatóriamente tal que $0 < k < m$ e não tenha divisores comuns com $\phi(m) = (p-1)(q-1)$. Esta restrição torna possível o cálculo da Chave Pública K tal que $K \times k = 1 \pmod{\phi(m)}$.

- O módulo m é tornado público juntamente com a Chave Pública K. A Chave Privada k é mantida secreta, o mesmo acontecendo com os factores p e q.
- A cifragem y de uma codificação x do texto limpo obtém-se por:

$$y = x^K \pmod{m}$$

- A decifragem obtém-se por:

$$x = y^k \pmod{m}$$

Paradigma da Criptografia de Chave Pública

As cifras simétricas podem ser comparadas a um **cofre**. Dois agentes possuem uma chave para o cofre. O agente emissor coloca a mensagem dentro do cofre. O agente receptor retira a mensagem do cofre.

As cifras assimétricas funcionam como uma **caixa de correio**. Qualquer pessoa pode meter uma mensagem na caixa do correio, desde que conheça o endereço (chave pública). Apenas o dono da caixa do correio tem a chave que permite recuperar as mensagens (chave privada).

Comunicação Utilizando Cifras Assimétricas

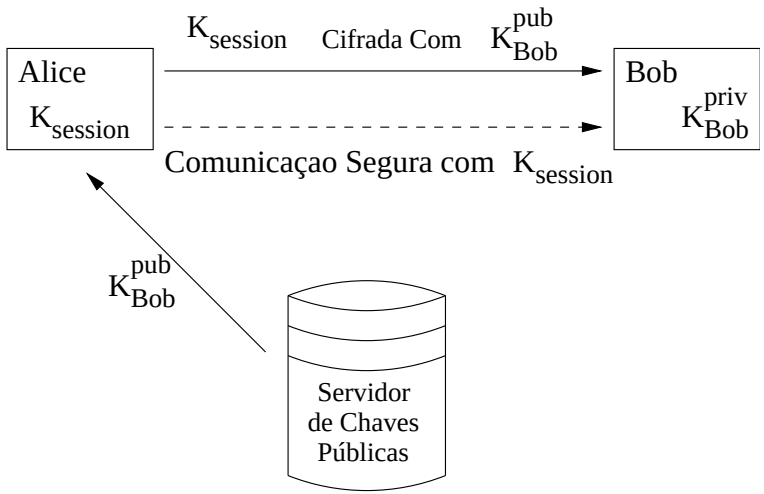
1. A Alice e o Bob combinam uma cifra assimétrica.
2. O Bob envia, em canal aberto, a sua Chave Pública.
3. A Alice cifra a mensagem com a Chave Pública e envia-a ao Bob.
4. O Bob decifra a mensagem com a sua Chave Privada.

- Fica resolvido o problema da partilha e gestão de chaves secretas: a Chave Privada pertence a um único agente.
- Com N agentes, cada qual terá o seu par de chaves. As chaves públicas podem ser armazenadas num servidor central no qual **não tem de haver confiança absoluta**.
- Resta o problema de garantir que uma chave pública pertence a um determinado agente. Porquê?

Cifras Simétricas vs Cifras Assimétricas

- As cifras assimétricas não substituem as cifras simétricas:
  - Os algoritmos assimétricos são pelo menos 1000 vezes mais lentos que os algoritmos simétricos que fornecem a mesma segurança. Devido à existencia de *atalhos* nos ataques a estas cifras, as chaves têm de ter no mínimo 1024 bits, o que torna os cálculos muito pesados.
  - Se a cifra é determinística é possível efectuar um ataque por texto limpo conhecido:
    - Conhecendo o criptograma e a chave pública podemos testar todas as possibilidades para o texto limpo.
    - Quando os textos limpos possíveis são poucos, isto pode ser um problema.
- As aplicações mais comuns das cifras assimétricas vêm complementar a funcionalidade das cifras simétricas.

Sistemas Híbridos/Envelopes Digitais



- Desaparece a necessidade de um TA e da gestão de chaves para cifragem de chaves:
  1. A Alice obtém a Chave Pública do Bob a partir do servidor.
  2. A Alice cifra a Chave de Sessão com essa Chave Pública.
  3. O Bob decifra a Chave de Sessão com a Chave Privada.

Aproximação Teórica

- Os algoritmos de cifra assimétrica são geralmente construídos com base num problema matemático *hard*, e.g. logaritmo discreto.
- Até que ponto é difícil resolver estes problemas? Estas são questões tratadas pelos peritos em teoria dos números computacional.
- A segurança de um algoritmo de cifra assimétrica é expressa em termos da sua relação com o problema *hard* em que se baseia.
- As provas de segurança são reduções que demonstram que quebrar o esquema de cifra implicaria ser capaz de resolver o problema difícil.

- Em termos concretos, obtêm-se resultados da forma

$$\mathtt{Adv}^{\mathtt{Problema}}(B) \geq K \cdot \mathtt{Adv}^{\mathtt{Esquema}}_{\mathtt{Modelo}}(A)$$

- A é um algoritmo hipotético que é capaz de obter uma vantagem não negligenciável num ataque ao esquema de cifra para um determinado modelo de segurança.
- B é um algoritmo probabilístico que utiliza A para resolver o problema *hard*.
- K é uma constante que depende dos detalhes da prova. Quanto menor for este valor, mais distante está a segurança do esquema do problema *hard*.
- Um resultado deste tipo estabelece uma referência para o tamanho da instância do problema *hard* que deve ser utilizado i.e. para o tamanho das chaves.

Objectivos de segurança

- O modelo mais utilizado para cifras assimétricas chama-se Ciphertext Indistinguishability (IND).
- A segurança neste modelo é expressa como a máxima vantagem de um adversário no seguinte jogo:
  - O adversário recebe uma chave pública que vai tentar atacar.
  - Quando estiver preparado, o adversário escolhe dois textos limpos (x_0, x_1) para ser desafiado.
  - O adversário recebe um criptograma correspondente ao texto limpo x_b.
  - O adversário tem de adivinhar o valor de b.

Modelos de ataque

- O jogo anterior exprime o objectivo de segurança IND para um atacante com acesso a criptogramas e à chave pública
- Contráriamente às cifras simétricas, o cenário é sempre de texto limpo escolhido (Chosen Plaintext Attack – CPA).
- As cifras assimétricas mais seguras, são desenhadas para resistir a ataques mais exigentes. O modelo mais utilizado é o de Chosen Ciphertext Attack – CCA.
- Para emular este tipo de cenário, o modelo anterior é alterado para que o adversário tenha acesso a um oráculo que decifra criptogramas arbitrários. (Restrições?).
- Caso o acesso ao oráculo seja limitado até à recepção do criptograma de desafio, o ataque designa-se por *não-adaptativo*. Caso contrário, é *adaptativo*.

RSA – Optimal Asymmetric Encryption Padding

- O algoritmo RSA como apresentado anteriormente não é seguro nestes modelos: é determinístico.
- O OAEP utiliza o mesmo problema RSA, visto como uma *one-way trapdoor permutation*, para construir uma cifra comprovadamente segura.
- A segurança do OAEP está demonstrada no modelo IND-CCA2, isto é, contra ataques de criptograma escolhido adaptativo.
- A prova de segurança assume a utilização de funções de hash "ideais".

- No RSA – OAEP, em vez de se cifrar o texto limpo directamente, cifra-se uma versão já processada:

$$[H_1([P||M] \oplus H_2(S)) \oplus S]||[[P||M] \oplus H_2(S)]$$

- Aqui, M representa a mensagem, e P o padding adequado.
- H_1 e H_2 são duas funções de hash criptográficas.
- S é uma sequência de bits aleatória.
- O standard PKCS#1 adopta este modo de utilização do RSA.

Introdução

- A autenticação de origem de mensagens permite fornecer a uma entidade que recebe uma mensagem garantias sobre a identidade do seu emissor.
- Este tipo de autenticação pode ser necessária quando:
  - a troca de informação entre dois agentes se processa sem que ambos estejam on-line simultaneamente e.g. e-mail.
  - a verificação da autenticidade dos dados é feita posteriormente e.g. um recibo.
  - a comunicação entre dois agentes se processa através de intermediários e.g. internet.
- A autenticação da origem de mensagens implica garantias de integridade: quem altera uma mensagem passa a ser o seu emissor.

Funções de Hash Criptográficas

- As funções de hash criptográficas (também chamadas funções de hash one-way, message digests, impressões digitais, etc.) são fundamentais a muitos protocolos criptográficos para garantir de integridade.
- As funções de hash são utilizadas na computação há muito tempo: transformam um input de tamanho variável num output de tamanho (menor) fixo.
- Para garantir integridade interessa extrair uma impressão digital não invertível de uma mensagem: obter **um valor que identifique o conteúdo essa mensagem**.
- Como as funções de hash não são injectivas, a identificação não pode ser unívoca: várias mensagens serão mapeadas no mesmo valor de hash.

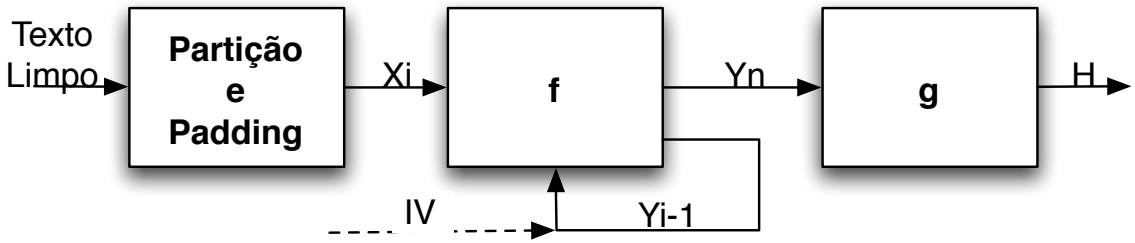
- O que se consegue na realidade é uma identificação probabilística: é provável que o valor de hash tenha sido originado por aquela mensagem.
- Espera-se que uma função de hash criptográfica seja:
  - **Pre-image resistant**. Dado um valor de hash é difícil encontrar uma pré-imagem desse valor.
  - **Second pre-image resistant** ou *fracamente livre de colisões*. Dado um hash e a mensagem que o originou, é difícil arranjar outra mensagem que origine o mesmo valor.
  - **Collision resistant** ou *fortemente livre de colisões*. É difícil encontrar duas mensagens que originem o mesmo hash.
- As funções de hash deste tipo podem ser tornadas públicas. A sua segurança está na baixa probabilidade de encontrar duas mensagens com o mesmo valor de hash.

Qual a dificuldade de obter colisões?

- Suponhamos que fixamos o valor de hash alvo numa gama de 2^H valores possíveis, e igualmente prováveis:
  - Um ataque por força bruta exige aproximadamente 2^{H-1} operações para encontrar uma colisão.
- Se o valor alvo não estiver pré-definido, então podemos executar um *birthday attack*:
  - Geramos n pares $(x, H(x))$, armazenando-os. As combinações possíveis são $n(n-1)/2$.
  - Para termos uma probabilidade de 50% de o ataque produzir uma colisão, temos $n \geq 1.2 \cdot 2^{H/2}$.
  - Este é o mesmo princípio que leva a que bastem aproximadamente 23 pessoas para se poder esperar que duas tenham o aniversário no mesmo dia.

Construção de Funções de Hash Criptográficas

- As funções de hash criptográficas mais utilizadas baseiam-se na construção genérica de Merkle-Damgård.



- A função *f* deverá ser livre de colisões, mas opera sobre inputs de tamanho fixo. A construção assegura que as suas propriedades se propagam para tamanhos arbirários.
- A função *g* fortalece as propriedades do resultado final e/ou ajusta o seu tamanho (função de finalização).
- O padding introduzido inclui uma codificação do comprimento total da mensagem.

Exemplo: MD4 e MD5

- O MD4 foi desenvolvido por Ron Rivest (um dos inventores do RSA) como uma função de hash para aplicações criptográficas que garantisse:
  - **Segurança**, no sentido em que o ataque mais eficiente à função de hash é o ataque por força bruta;
  - **Segurança directa**, no sentido em que a base da segurança da função de hash não reside em pressupostos de complexidade computacional como sejam a dificuldade em factorizar um inteiro grande.
  - **Eficiência e simplicidade**.
- Uma série de avanços na criptoanálise de alguns dos rounds do MD4 levou o autor a melhora-lo, desenvolvendo o MD5.
- MD4 e MD5 produzem valores de hash de 128 bits.

Exemplo: Funcionamento do MD5

- A mensagem é vista como uma sequência de b bits, aos quais é adicionada uma sequência $\{1, 0, 0, \ldots\}$ para a obter um stream de comprimento $b' = 448 \pmod{512}$.
- À sequência anterior é adicionada uma representação de 64 bits do valor b (comprimento final múltiplo de 512).
- Quatro registos de 32 bits (A, B, C e D) são inicializados com valores constantes. Cada bloco de 512 bits é tratado como 16 parcelas de 32 bits (X_k).
- Existem 4 rounds em que os X_k são combinados com os valores dos registos A, B, C e D através de operações lógicas bit a bit (não lineares) e rotações.
- No final do algoritmo, os registos contêm o resultado final.

Exemplo: Secure Hash Algorithm (SHA)

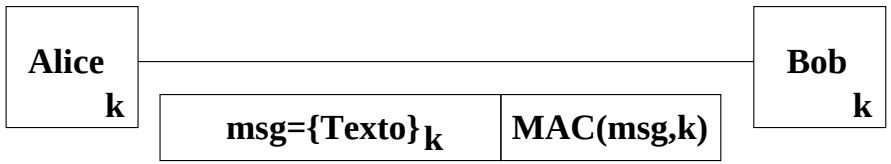
- O SHA foi desenvolvido pelas agencias governamentais americanas NIST e NSA para incluir no standard de assinaturas digitais DSS.
- Actualmente utiliza-se uma versão melhorada: o SHA-1.
- O valor de hash produzido pelo algoritmo é de 160 bits.
- O SHA foi desenhado para que seja "*. . . computationally unfeasible to recover a message . . . , or to find two different messages which produce the same message digest.*"
- O SHA foi também desenvolvido com base no MD4, mas de forma diferente e independentemente do MD5. Os critérios de desenvolvimento não foram divulgados.
- Recentemente foram publicados diversos ataques ao MD5 e ao SHA que são motivo de alguma preocupação.

Message Authentication Codes

- Um Message Authentication Code (MAC) pode ser visto como uma função de hash criptográfica cujo resultado depende, não só da mensagem, mas também de uma chave secreta.
- Para gerar o MAC é necessário conhecer o algoritmo e a chave secreta. O mesmo acontece para o verificar.
- Aplicações:
  - impressão digital que garante que o checksum da mensagem foi calculado na sua origem.
  - protecção de ficheiros contra ataques de virus, uma vez que o virus seria incapaz de produzir um MAC válido para esconder as alterações que introduzisse.

Message Authentication Codes: Utilização

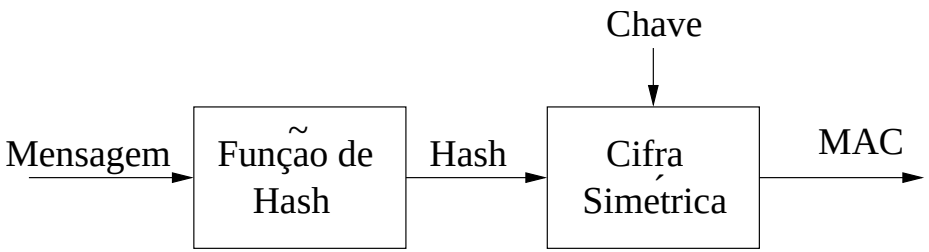
- A confidencialidade implica a autenticação da origem dos dados?
  - Se apenas Alice e Bob conhecem k, então nenhum intruso consegue alterar *msg* sem ser detectado na decifragem?
  - **Isto não é verdade**: implica, no mínimo, semântica ou redundância no texto limpo. E se for aleatório?
  - Com um MAC é possível comprovar a origem da mensagem e a sua integridade, antes de a decifrar.
  - É também possível (e até frequente) utilizar MACs em aplicações que não requerem confidencialidade.



- Que garantias tem Bob quando recebe a mensagem?

Message Authentication Codes: Implementação

- Opções de implementação:
  1. Gerar um valor de hash com uma função de hash criptográfica e utilizar uma cifra simétrica para cifrar esse valor.



  2. Utilizar uma cifra simétrica por blocos num modo com realimentação dos criptogramas (CBC ou CFB), cifrar a mensagem e aproveitar o último bloco do criptograma como MAC (talvez cifrando-o mais uma vez no mesmo modo).

Exemplo: HMAC

- O HMAC é um algoritmo de MAC desenvolvido inicialmente para ser utilizado no IPsec. É actualmente o algoritmo de MAC mais utilizado.
- Baseia-se numa função de hash criptográfica (HMAC-SHA1 ou HMAC-MD5) e utiliza a chave secreta em operações de concatenação dos streams de bytes processados:
  - $HMAC(K, text) = HASH(B|A)$
  - $B = (K \uplus padding_{output})$
  - $A = HASH((K \uplus padding_{input})|text)$
- Ignorando as operações de XOR, que combinam a chave com valores constantes, temos:
  - $HMAC(K, text) = HASH(K|HASH(K|text))$

Assinaturas Digitais

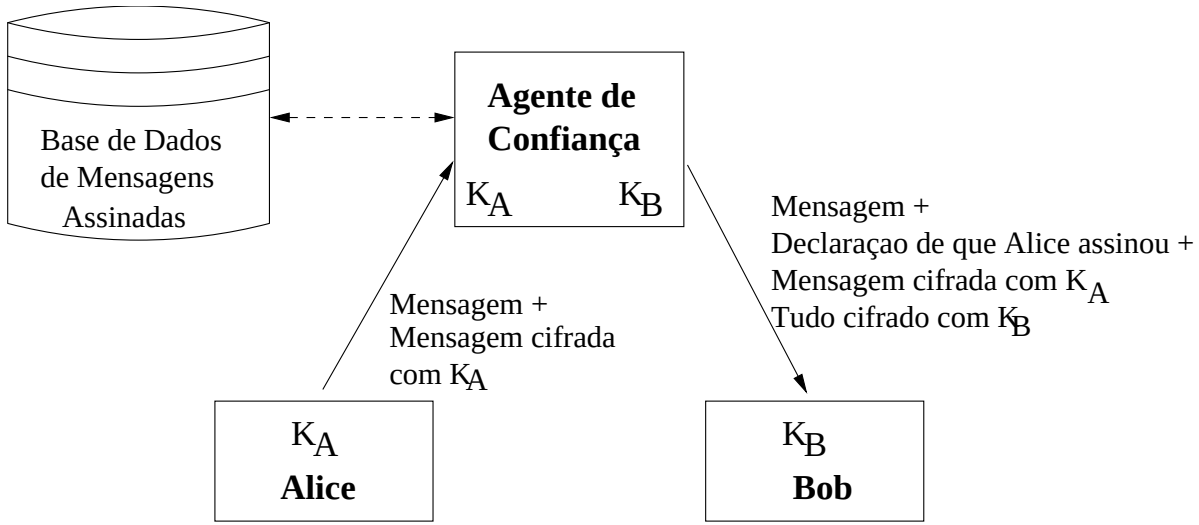
- A assinatura manuscrita é há muito utilizada como prova de autoria ou, pelo menos, de concordância com o conteúdo de um documento. A assinatura deve ser:
  - **autêntica**: convence o receptor do documento de que o signatário explicitamente assinou o documento i.e. que conhecia o seu conteúdo, e.g. por ser o seu autor.
  - **não falsificável**: prova que o signatário, e não outra pessoa, assinou o documento.
  - **não reutilizável**: faz parte do documento e não se pode transpor para outro documento.
  - garante da **integridade do documento**: o documento permaneceu inalterado desde que foi assinado.
  - **não repudiável**: o signatário não pode, à posteriori, negar que assinou o documento.

- Na realidade nenhuma destas propriedades é respeitada de forma absoluta por uma assinatura manuscrita.
- Uma **assinatura digital** é o equivalente a uma assinatura manuscrita, mas aplicada a documentos electrónicos i.e. qualquer tipo de ficheiro.
- A implementação de assinaturas digitais é mais complicada do que no caso de documentos em papel por diversas razões:
  - Os ficheiros de computador são muito fáceis de copiar.
  - Se se utilizasse uma versão digitalizada da assinatura manuscrita embebida no documento assinado, essa assinatura seria muito fácilmente manipulável e reutilizável.
  - Os ficheiros de computador são muito fáceis de alterar sem deixar vestígios.

- **Uma assinatura digital é uma sequência de bytes e não tem significado fora do contexto de um algoritmo específico**. Acompanha o documento assinado.
- Associados a um esquema de assinaturas digitais há sempre dois procedimentos complementares:
  - **Geração da assinatura**. Procedimento segundo o qual o signatário produz a sequência de bytes a partir do documento e de informação secreta ou privada (uma chave). Este procedimento deve assegurar as propriedades requeridas para a assinatura.
  - **Verificação da assinatura**. Procedimento segundo o qual o destinatário do documento assinado consegue assegurar-se de que a sequência de bytes que recebe como assinatura é um valor válido, gerado pelo signatário para esse fim.

Assinaturas Digitais: Versão 1

- Com cifras simétricas e um agente de confiança (TA):

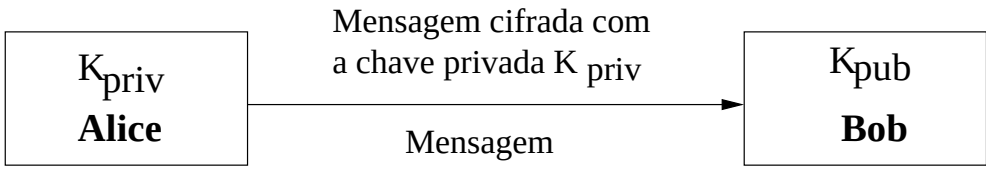


  - Demasiado exigente para o agente de confiança: eficiência na comunicação entre muitos agentes e armazenamento das mensagens assinadas.

- Assinaturas digitais com cifras simétricas:
  - O procedimento de verificação é efectuado por defeito: o Bob confia no TA e aceita a garantia que lhe é transmitida.
  - A assinatura é autêntica porque o TA garante ao Bob que a Alice assinou a mensagem.
  - A assinatura não é falsificável porque apenas a Alice e o TA conseguem cifrar a mensagem com K_A.
  - A assinatura não é reutilizável porque o Bob não conseguiria apresentar outra mensagem cifrada com K_A.
  - A mensagem não é alterável pela mesma razão.
  - A assinatura não é repudiável porque o Bob pode sempre demonstrar que a Alice assinou mostrando a mensagem cifrada com K_A.
  - A base de dados do TA permite evitar a transferência da mensagem cifrada com K_A para o Bob.

Assinaturas Digitais: Versão 2

- Assinaturas digitais baseadas em cifras assimétricas:



- É necessário um algoritmo de cifra assimétrica que permita a cifragem utilizando a chave privada e a decifragem utilizando a chave pública.
- Neste caso não há necessidade de um agente de confiança (a não ser para a emissão de certificados de chave pública como vamos ver mais tarde).

- Assinaturas digitais baseadas em cifras assimétricas:
  - A verificação da assinatura consiste na decifragem da mensagem cifrada utilizando a chave pública da Alice, comparando o resultado com a mensagem recebida como texto limpo.
  - A assinatura é autêntica porque apenas a Alice conhece a sua chave privada.
  - A assinatura não é falsificável nem reutilizável porque o Bob não conseguiria apresentar outra mensagem cifrada com a chave privada da Alice.
  - A mensagem não é alterável pela mesma razão.
  - A assinatura não é repudiável porque o Bob pode sempre demonstrar que a Alice assinou, mostrando a mensagem cifrada com a chave privada da Alice.

Assinaturas Digitais: Notas Importantes

- Actualmente, as soluções baseadas na criptografia de chave pública dominam, principalmente por não requererem um TA on-line.
- **Nem todos os algoritmos de assinatura são adaptações directas de cifras assimétrica**.
- É muito importante **não confundir** uma assinatura digital, que confere **autenticação** com uma cifra assimétrica.
- **Deve evitar-se** referir as operações de assinar e verificar uma assinatura como *cifrar com chave privada* ou *decifrar com chave pública*.
- O objectivo das assinaturas digitais **não é conferir confidencialidade**: acompanham o documento a que se referem, que pode ou não ser cifrado.

Exemplo: Assinatura RSA

- É uma adaptação directa da cifra assimétrica RSA.
- Os cálculos matemáticos envolvidos são exactamente os mesmos, mas a cifragem passa a ser efectuada com a chave privada e a decifragem com a chave pública.
- Na geração da assinatura, o signatário aplica este processo modificado de cifragem ao documento que pretende assinar.
- O destinatário recebe o documento acompanhado pela sua versão cifrada, que funciona como assinatura.
- O destinatário verifica a assinatura decifrando-a utilizando a chave pública do signatário e comparando o resultado com o documento original.

Exemplo: Digital Signature Algorithm (DSA)

- Em 1991 a agencia americana NIST propôs este standard de assinaturas digitais de utilização livre.
- Esta decisão causou polémica, uma vez que muitas aplicações tinham já sido desenvolvidas com base no RSA, nessa altura licenciado, significando isto uma considerável perda para as empresas responsáveis.
- Além de ser livre de royalties, o grande objectivo de desenvolvimento do DSS foi tornar a operação de **geração** da assinatura mais leve (e.g. para smartcards).
- O DSA baseia-se nas técnicas de chave pública El Gamal, cuja segurança se baseia no problema do logaritmo discreto (PLD).

- As principais criticas ao DSA foram as seguintes:
  - Não pode ser utilizado para cifragem ou distribuição de chaves.
  - Foi desenvolvido pela NSA, e os adeptos das teorias da conspiração falavam numa alegada porta das traseiras.
  - É mais lento que o RSA, e este é um standard *de facto*
  - O seu desenvolvimento não foi um processo público, e a sua utilização pode infringir patentes.
  - O tamanho da chave era muito pequeno: 512 bits inicialmente. Sendo esta a única crítica verdadeiramente objectiva, o standard foi alterado para permitir também chaves de 1024 bits.
- Uma curiosidade é o facto de ser possível utilizar componentes da implementação do DSA para cifrar informação utilizando o algoritmo de chave pública El Gamal, e mesmo o RSA.

Assinaturas Digitais e Funções de Hash

- Os algoritmos assimétricos são muito pouco eficientes: assinar mensagens de tamanho realista é um problema.
- É mais comum gerar assinaturas digitais a partir de impressões digitais das mensagens.
- O processo de geração começa pelo cálculo de um valor de hash, que é depois assinado.
- A verificação da assinatura implica um novo cálculo do hash da mensagem. A validade da assinatura é avaliada através de uma algoritmo que processa este hash, a chave pública do signatário e a assinatura.
- Este método, em teoria, torna mais dificil o ataque à assinatura: o hash funciona como checksum da mensagem.

Assinaturas Digitais e Cifragem

- Combinando assinatura digital e cifragem obtém-se um protocolo para confidencialidade e autenticação.
- **Faz mais sentido assinar antes ou depois da cifragem**?
- A primeira hipótese é melhor por vários motivos:
  - Se a mensagem não é assinada antes de ser cifrada, pode haver dúvidas quanto ao conhecimento que o signatário tinha do seu conteúdo.
  - Um intruso não tem acesso à informação de autenticação, isto é, à assinatura, a não ser que quebre a cifra.
  - Um ataque de substituição ou reutilização da assinatura deixa de fazer sentido.
- Este conceito é intuitivo: as cartas são também assinadas e depois inseridas num envelope.

Assinaturas Digitais e Cifragem

- Combinando assinatura digital e cifragem obtém-se um protocolo para confidencialidade e autenticação.
- **Faz mais sentido assinar antes ou depois da cifragem**?
- A primeira hipótese é melhor por vários motivos:
  - Se a mensagem não é assinada antes de ser cifrada, pode haver dúvidas quanto ao conhecimento que o signatário tinha do seu conteúdo.
  - Um intruso não tem acesso à informação de autenticação, isto é, à assinatura, a não ser que quebre a cifra.
  - Um ataque de substituição ou reutilização da assinatura deixa de fazer sentido.
- Este conceito é intuitivo: as cartas são também assinadas e depois inseridas num envelope.

Exemplo: Protocolo de Acordo de Chaves Station-to-Station (STS)

- O protocolo de acordo de chaves Diffie-Hellman é vulnerável a um ataque chamado *man-in-the-middle*:
  - O intruso intercepta as mensagens trocadas entre os agentes, substituindo-as por mensagens suas.
  - Como não há autenticação, os agentes terminam o protocolo acreditando que acordaram uma chave entre si.
  - Na realidade, cada um dos agentes acordou uma chave secreta com o intruso, sem o saber.
  - Enquanto o intruso fizer o relay das mensagens entre os agentes, estes não se apercebem da sua presença.
- Outro problema com o protocolo DH é não incluir, explicitamente, um passo de confirmação da chave acordada.

- O protocolo STS corrige estes problemas acrescentando um novo passo ao protocolo DH, com base num esquema de assinaturas digitais:
  - Uma vez acordada a chave de sessão, os agentes assinam digitalmente o par ordenado (X, Y).
  - Estas assinaturas são trocadas entre os agentes, cifradas com a chave recém-acordada.
  - Só se considera que o protocolo terminou com sucesso se as assinaturas forem recuperadas e verificadas correctamente.
- Mais uma vez convém notar que, para este sistema ser seguro, é necessário estabelecer a validade das chaves públicas dos agentes: é necessário um sistema de certificação de chaves públicas.

Introdução

- Um agente (o **identificado** ou A) demonstra a sua identidade a outro agente (o **identificador ou B**). Estes agentes podem, ou não, ser utilizadores humanos.
- A autenticação de entidades implica uma noção de tempo: é feita em tempo-real. Isto pode não acontecer para a autenticação de mensagens.
- Os objectivos dos protocolos de identificação são:
  - No caso de entidades honestas, B aceita a identidade de A.
  - B não pode usar a identidade de A perante terceiros.
  - É muito pouco provável que uma entidade C consiga passar-se por A.
  - As propriedades anteriores mantêm-se verdadeiras mesmo depois de um número arbitrário de execuções do protocolo.

- A identificação pode fazer-se de acordo com:
  - **Aquilo que se sabe** e.g. passwords.
  - **Aquilo que se possui** e.g. um cartão funcionando como passaporte.
  - **Aquilo que se é** e.g. propriedades biométricas inerentes ao indivíduo.
- São exemplos de mecanismos de identificação biométricos:
  - a identificação através da impressão digital,
  - do reconhecimento das feições ou
  - do reconhecimento da voz.
- Estamos interessados em protocolos de identificação que não se baseiam na utilização de informação biométrica.

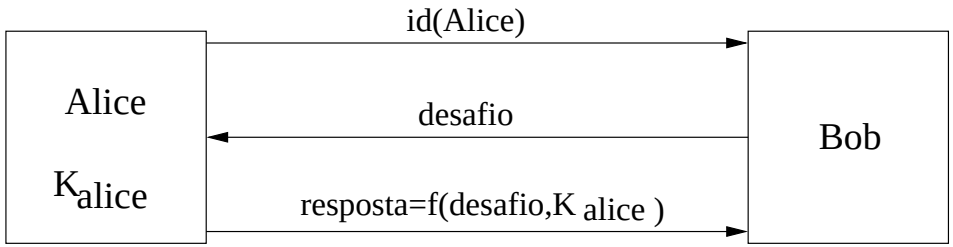
- Associa-se um *token* de informação privada à identidade do agente.
- Idealmente, apenas o próprio agente conhecerá ou terá acesso a essa informação.
- A identificação efectua-se através da demonstração do conhecimento ou posse dessa informação privada.
- Exemplo: **Mecanismo Login/Password**
  - O utilizador identifica-se fornecendo a sua identificação e a sua informação privada (a password).
  - Problema: o identificador (ou intruso) conhece a informação privada !

Princípio do Conhecimento Zero

- Como demonstrar o conhecimento de informação privada ou secreta sem a divulgar explicitamente? A **prova de conhecimento zero** resolve este problema.
- Permite implementar um sistema de identificação ideal: demonstra-se o conhecimento do segredo associado à identidade sem revelar informação privada.
- Os protocolos deste tipo são probabilísticos:
  - É feita uma pergunta aleatória cuja resposta depende do segredo.
  - Não conhecendo o segredo, é possível acertar na resposta com 50% de probabilidade.
  - Fazendo uma série de perguntas, consegue-se estabelecer a identidade com uma probabilidade de erro arbitrariamente pequena.

Mecanismo de Desafio/Resposta

- Este mecanismo é o componente básico dos protocolos de identificação que seguem, ou aproximam, o princípio do conhecimento zero:



- Para o protocolo ser verdadeiramente de conhecimento zero, a resposta não pode implicar a transferência de informação da Alice para o Bob que permita a reconstrução do segredo !!!
- Note-se que qualquer mecanismo de assinatura digital pode ser utilizado como $f(desafio, K_{alice})$.

Exemplo: Protocolo de Identificação de Schnorr

- Parâmetros públicos: dois primos grandes p e q, sendo q divisor de $p-1$; um número g tal que $g^q \pmod{p} = 1$.
- A Alice gera um par de chaves: a chave privada é um número aleatório $0 < s < q$; a chave pública, conhecida pelo Bob, é $v = g^{-s} \pmod{p}$.
- A Alice pretende identificar-se perante o Bob:
  1. A Alice escolhe um número aleatório $0 < r < q$, calcula $x = g^r \pmod{p}$ e envia x ao Bob.
  2. O Bob envia à Alice um desafio aleatório $0 < e < 2^t - 1$.
  3. A Alice responde com $y = (r + s * e) \pmod{q}$.
  4. O Bob verifica que $x = g^y v^e \pmod{p}$.
- Mais uma vez é necessário estabelecer a validade da chave pública da Alice.

Exemplo: Protocolo de Identificação de Schnorr

- O valor t estabelece um nível de certeza.
- Para um valor de t muito pequeno, a probabilidade de um agente C se poder fazer passar pela Alice é grande:
  - C adivinha o valor e; e gera um valor de y qualquer.
  - C envia $x = g^y v^e \pmod{p}$.
  - Se o desafio coincidir com e, a resposta y será correcta.
- Para valores de t elevados, a prova do conhecimento de s é mais convincente, uma vez que a probabilidade do ataque anterior ser conseguido é muito baixa.
- Paradoxalmente, para valores de t elevados o protocolo Schnorr não é de *conhecimento zero*, uma vez que o Bob pode controlar e para obter uma solução para uma equação em s que não seria capaz de contruir sozinho.