Criptografia Aplicada
LESI/LMCC - 2004/2005

Manuel Bernardo Barbosa

November 15, 2004

Manuel Bernardo Barbosa - mbb@di.uminho.pt

Avaliação

- A avaliação será efectuada de acordo com as duas componentes:
  - Teórica – 60% (com nota mínima de 8 valores no exame)
  - Prática – 40% (sem a qual a nota final será majorada a 12 valores)

- Avaliação prática:
  - O trabalho é realizado maioritariamente nas aulas TP, tendo por base um conjunto de exercícios propostos.
  - Ao longo das aulas serão implementados pequenos módulos de software que possam ser utilizados para construir uma pequena aplicação com funcionalidades de segurança.
  - No final do semestre será necessário entregar um relatório e, possivelmente, uma apresentação do trabalho efectuado.
  - Os trabalhos serão realizados por grupos de exactamente dois alunos.

Programa

- Terminologia
  - Cifras Simétricas
  - Cifras Assimétricas (Criptografia de Chave Pública)
  - Funções de Hash Criptográficas, Assinaturas Digitais e Message Authentication Codes
  - Identificação

- Certificação e Public Key Infrastructure (PKI)
  - Certificados de Chave Pública X.509
  - Infraestrutura de Chave Pública
  - Certification Revokation Lists (CRL)
  - Problemas com o X.509 e PKI
  - PKCS: Public Key Cryptography Standards
  - Certificados de Atributos

Programa$_{cont}$

- Aplicações Correntes da Criptografia
  - Pretty Good Privacy (PGP)
  - IPsec
  - Secure Sockets Layer (SSL)
  - Kerberos
  - Secure Shell (SSH)
- Utilização de dispositivos portáteis em aplicações criptográficas
  - iButtons
  - Smartcards
  - PDA

O que é a Criptografia?

- A **Criptografia** é a ciência de tornar possível a comunicação segura entre dois agentes, sobre um canal aberto. A **Criptoanálise** é a $arte$ de gorar os objectivos da Criptografia, isto é, $quebrar$ a segurança da comunicação.

- Conjuntamente, a Criptografia e a Criptoanálise formam uma área a que se chama **Criptologia**; uma área com profundas raízes na Matemática.

- A Criptografia existe desde a antiguidade. Foi sempre uma ciência associada às actividades militares, e em que a segurança dependia, em grande parte, do secretismo que rodeava as técnicas utilizadas.

- Esta tendência acentuou-se no Século XX, nomeadamente durante as $1^{\underline{a}}$ e $2^{\underline{a}}$ Guerras Mundiais e prolongou-se durante as primeiras décadas da Guerra Fria.

O que é a Criptografia?$_{cont}$

- Só no princípio dos anos 70 surgiu como área de investigação académica de conhecimento generalizado. Foi reconhecida a importância de eliminar o obscurantismo como factor na segurança dos sistemas criptográficos.

- Hoje, a Criptografia e a sua utilização nos Sistemas de Informação modernos ganham cada vez mais importância. De facto, o seu estudo é já recomendado nos Curricula IEEE/ACM para cursos de Informática e Ciências da Computação.

- Esta disciplina enquadra-se precisamente neste contexto. Nela estuda-se a utilização da Criptografia nos Sistemas de Informação modernos. Este estudo centra-se mais nos conceitos, standards, frameworks e aplicações ligados à Criptografia, do que na teoria que lhe está associada.

História da Criptografia Moderna

- **1970 a 1977** – Desenvolvimento e estandardização do Data Encryption Standard.
- **1976** – Primeiro paper de Diffie e Hellmann definindo os princípios da criptografia de chave pública.
- **1978** – Rivest, Shamir e Adleman descobrem a primeira cifra assimétrica: o RSA.
- **1985** – Descoberta da cifra assimétrica El Gamal.
- **1995** – Estandardização do Digital Signature Algorithm.
- **2001** – Escolha do substituto do DES: Advanced Encryption Standard.

Módulo I

Terminologia

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.

Segurança da Informação$_{cont}$

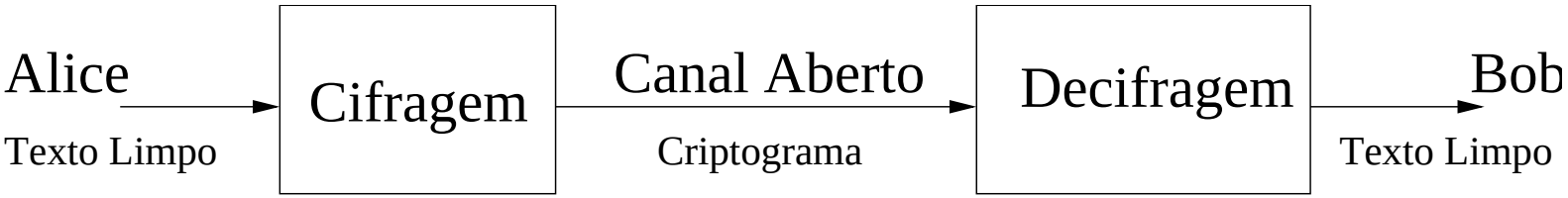
- Desde há muito que este tipo de objectivos são satisfeitos de outras formas e, na maior parte dos casos, sem recorrer a técnicas criptográficas.

- 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 algoritmos matemáticos: são muitas vezes necessários procedimentos 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

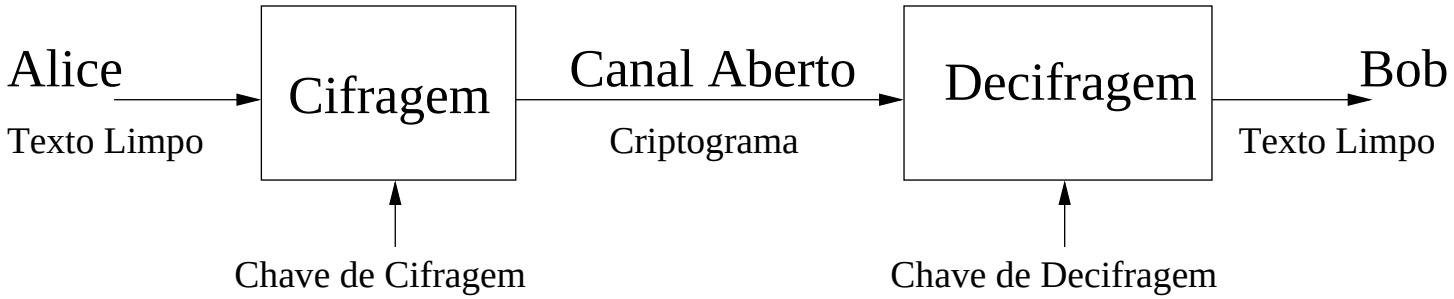
- Uma **cifra** é um algoritmo criptográfico, i.e. uma função matemática injectiva, que efectua as transformações entre o **texto limpo** e o **criptograma**.

- 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. Na realidade, a qualidade de uma cifra mede-se pelo tempo que permanece inquebrável, mesmo sendo do conhecimento geral.

- As cifras modernas são famílias de funções cujo comportamento depende de um parâmetro numérico denominado por **chave**.

Cifras e Chaves$_{cont}$



- 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 precisa de a 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 . . .
  - **Texto limpo escolhido adaptativo**. Implica acesso à $máquina$ de cifragem.
  - **Criptograma escolhido**. Implica acesso à $máquina$ de decifragem.
  - **Outros. . .** Violência, extorsão, etc.

Criptoanálise de um Protocolo

- Quando se trata de atacar um protocolo criptográfico, alguns dos ataques possíveis são os seguintes:
  - **Chave conhecida**. O intruso utiliza chaves utilizadas em instâncias anteriores do protocolo para tentar obter as chaves actuais.
  - **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.

Módulo I.A

Cifras Simétricas

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. se uma cifra tem uma complexidade de processamento de 2^{128}, é este o número de operações necessárias para quebra-la.

- 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.

Segurança de uma cifra$_{cont}$

- Uma cifra diz-se **incondicionalmente segura** se o número de criptogramas conhecidos nunca for suficiente para quebrar a cifra.

- Todas as cifras são quebráveis por pelo menos um ataque de criptograma conhecido: 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.

Mistura e Difusão

- 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 e o criptograma. Na sua forma mais simples, implementa-se através de uma substituição. Geralmente implica a aplicação de operações algébricas e funções não lineares aos bits do texto limpo e aos bits da chave.

- **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. Geralmente recorre-se a permutações, rotações, operações algébricas, funções não lineares, etc.

Classificação de Cifras

- Os *inputs* (e os *outputs*) das cifras são invariavelmente sequências de símbolos de um determinado alfabeto: usualmente serão sequências de bits que resultam da codificação de mensagens.

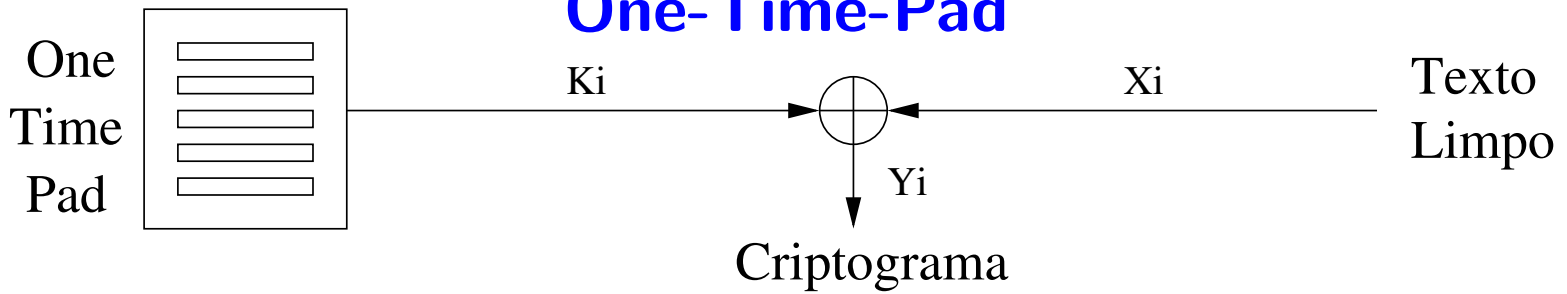
- As cifras classificam-se também com base na relação que estabelecem entre os símbolos do texto limpo e os símbolos do criptograma:
  - **Monoalfabética**. Cada símbolo do texto limpo influencia apenas um símbolo do criptograma. Este tipo de cifra é susceptível a ataques por análise de frequências, o que limita a sua segurança.
  - **Polialfabética**. Cada símbolo do criptograma depende de vários símbolos do texto limpo: os padrões de ocorrência dos símbolos no texto limpo são *mascarados*. Uma cifra polialfabética é *pura* se cada símbolo do criptograma depende de todos os símbolos do texto limpo.

Classificação de Cifras$_{cont}$

- A implementação de uma cifra polialfabética pura tem severas limitações a nível prático, e que se prendem com:
  - A inviabilidade, em muitas aplicações, de dispor de todo o texto limpo na altura em que se pretende iniciar a cifragem.
  - A complexidade computacional de cifrar mensagens muito longas com uma cifra polialfabética pura.

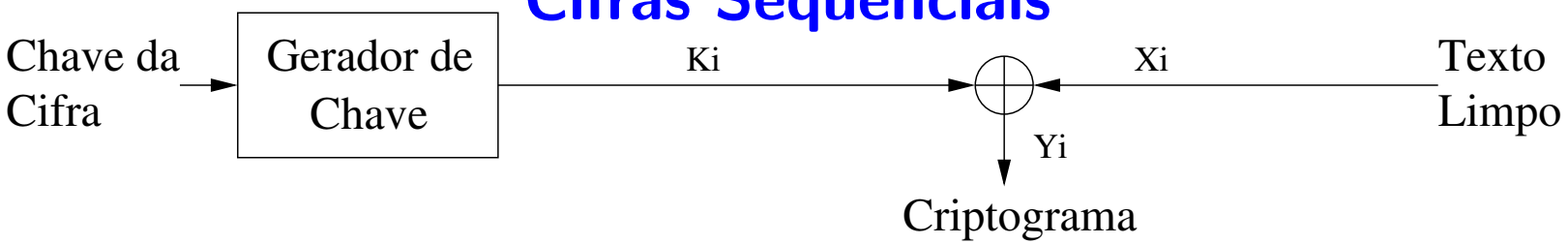
- As cifras modernas podem ser vistas como aproximações às cifras polialfabéticas puras. Desta perspectiva, importa distinguir três tipos de cifras:
  - One-Time-Pad.
  - Cifras Sequenciais.
  - Cifras Por Blocos.

One-Time-Pad



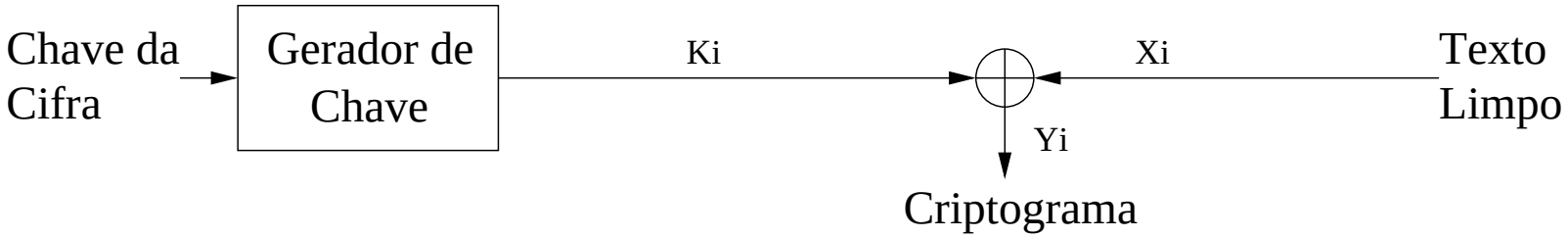
- A chave no one-time-pad é uma sequência de símbolos, do mesmo tamanho do texto limpo, e gerada aleatoriamente.
- Cada símbolo do texto limpo é combinado com um único símbolo da chave para originar um símbolo do criptograma.
- É **incondicionalmente segura**. A chave só é utilizada uma vez, até porque é recuperável por um ataque de texto limpo conhecido.
- Problemas de ordem prática: geração e distribuição da chave.

Cifras Sequenciais



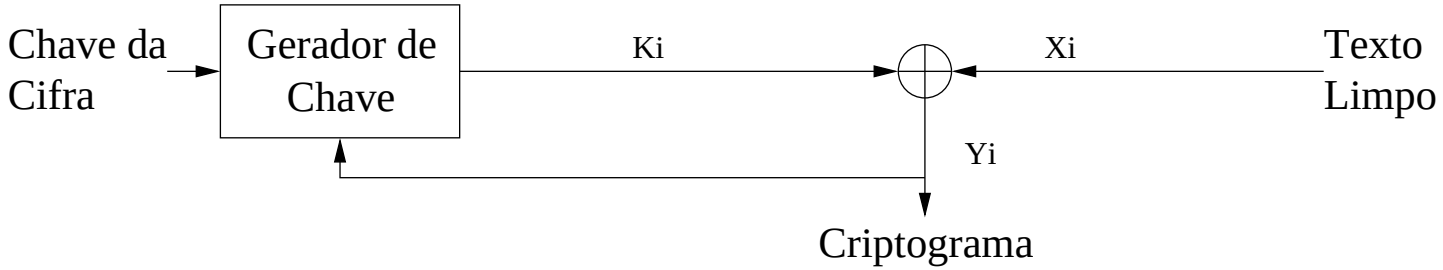
- Estas cifras operam sobre *streams* de texto limpo, um bit ou um byte de cada vez, combinando-o com uma stream de chaves gerada internamente.
- O esquema de decifragem é idêntico ao de cifragem. A sequência de chaves tem de ser reproduzida exactamente para recuperar o texto limpo.
- A chave da cifra funciona como semente/inicialização do gerador.
- A segurança da cifra reside na dificuldade de prever a sequência de valores gerados sem saber a chave da cifra i.e. reside totalmente no gerador de chaves, que deverá gerar sequências com o **maior período 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 o que permite a recuperação da perda do sincronismo.

- Tipicamente, o texto-limpo é iniciado por uma sequência aleatória que garante que o sincronismo já foi alcançado no início do texto limpo propriamente dito.

- 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 uma matriz ($S\text{-}box$) de 8x8, com uma permutação dos valores de 0 a 255, permutação essa gerada a partir da chave da cifra.

Exemplo: RC4$_{cont}$

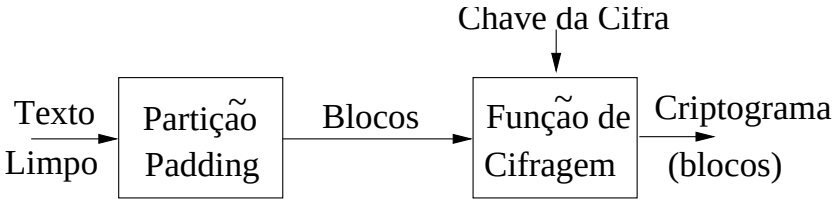
- O preenchimento da S-box começa pela geração de dois arrays de bytes S e K, de 256 bytes cada um:
  - S é preenchido sequencialmente com valores de 0 a 256.
  - K é preenchido com os bytes da chave, repetida as vezes necessárias.

- Depois executa-se o seguinte algoritmo de "randomize":

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

- Os proprietários dizem que é imune à criptoanálise diferencial. Há ainda alguns problemas de direitos de utilização do nome da cifra.

Cifras Por Blocos



- O principal componente destas cifras é uma função de cifragem polialfabética pura, que 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 a partir do texto limpo.

- Na decifragem, a função inversa da função de cifragem é aplicada aos blocos do criptograma. Os processos de $padding$ e partição são posteriormente revertidos para recuperar o texto limpo.

Cifras Por Blocos$_{cont}$

- As cifras por blocos são fundamentais aos sistemas criptográficos. Não apenas para garantir confidencialidade, mas também como componentes na construção de funções de hash, geradores de números pseudo-aleatórios, ou Message Authentication Codes.

- 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 uma probabilidade de $1/2$.
  - 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. O seu processo de desenvolvimento e aprovação é ainda hoje 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.

Exemplo: Data Encryption Standard (DES)$_{cont}$

- O algoritmo consiste na aplicação sucessiva (16 vezes) ao texto limpo de uma operação de cifragem simples: um **round**.

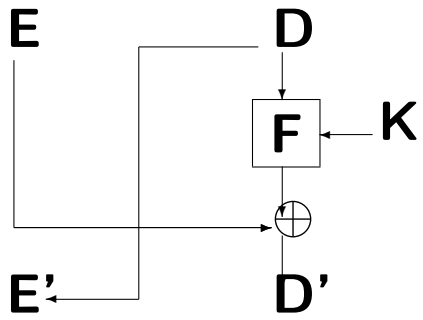
- Para cada round é derivada uma chave de round através de uma permutação da chave da cifra. Este processo chama-se **key schedule**.

- Um round do DES é construido com base num **Circuito de Feistel**, o que permite implementações muito eficientes, dado que 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).

Exemplo: Data Encryption Standard (DES)$_{cont}$

- 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: porque é patenteado e é necessária uma licença para a sua utilização; porque não foi criado com esse objectivo, donde não passou pelo escrutínio desejável.

- O tamanho do bloco é, como no DES, de 64 bits, mas a chave é já de 128 bits. O mesmo algoritmo é utilizado para cifragem e decifragem, o que é um factor importante para a obtenção de implementações eficientes.

- O algoritmo 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 vencedor chama-se **Rijndael** e foi desenvolvido por investigadores de uma universidade Belga.

- O AES é uma cifra com tamanho de bloco variável ($N_b \times 32$ bits, com N_b =4,6 e 8) e com tamanho de chave variável ($N_k \times 32$ bits, com N_k =4,6 ou 8). Como o DES, esta é uma cifra iterada. Neste caso, o número de rounds da cifra depende de N_b e N_k.

- Cada round é constituido por três camadas de operações de características distintas que, conjuntamente, introduzem a difusão e mistura necessárias.

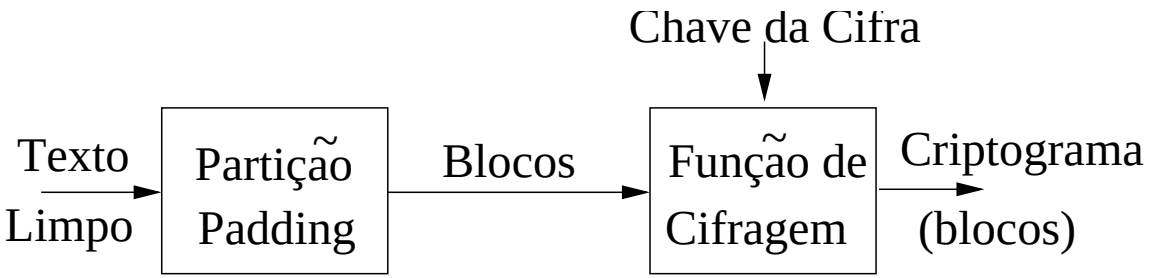
- O AES utiliza operações algébricas de forma inteligente: um nível de segurança adequado é conseguido através de operações complexas, mas de implementação muito simplificada.

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)



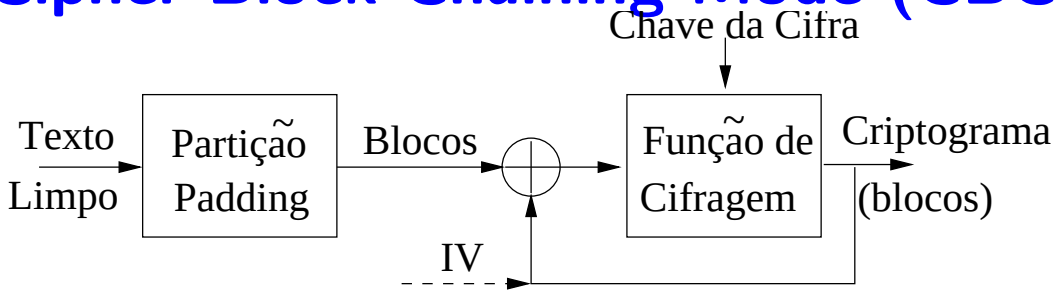
- Segurança
  - Os padrões existentes no texto limpo não são disfarçados.
  - Um bloco cifrado duas vezes com a mesma chave resulta em criptogramas iguais.
  - Susceptível a ataques por $code\ book$ (compilação de pares texto limpo/criptograma).
  - Susceptível a ataques por remoção, troca e repetição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

Electronic Code Book Mode (ECB)$_{cont}$

- 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 pela operação XOR. De blocos de texto limpo iguais passam a resultar criptogramas distintos.
  - Isto impede ataques por code book e ataques por repetição.
  - Ataques por manipulação de blocos são, em grande parte, detectáveis.

Cipher Block Chaining Mode (CBC)$_{cont}$

- 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 implementação em 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 do texto limpo, mas isto agrava significativamente a utilização de um canal de comunicação.

Cifras por Blocos vs Cifras Sequenciais$_{cont}$

- Além disso, uma utilização mais segura das cifras por blocos (CBC) implica cifrar todos os blocos de texto limpo numa só cifragem.

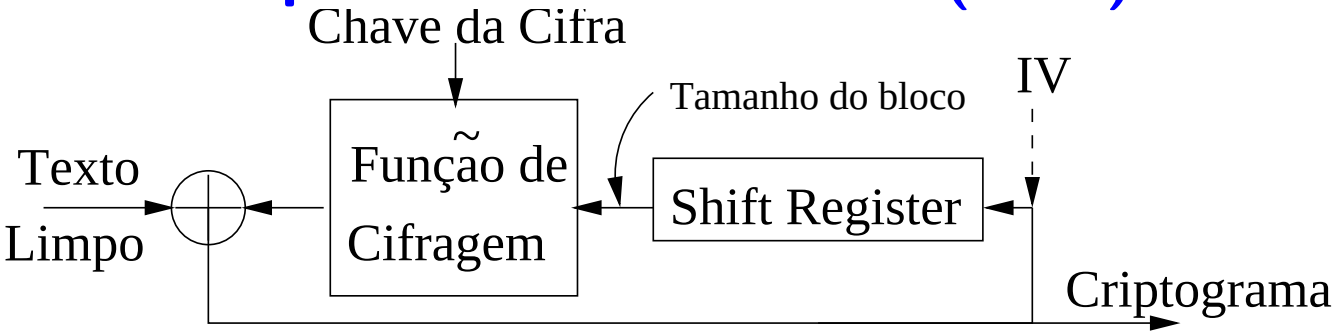
- O núcleo de uma cifra por blocos pode também ser utilizado para construir uma cifra sequencial, em aplicações onde uma cifra deste tipo seja mais apropriada.

- O Cipher Feedback Mode e o Output Feedback Mode servem este fim.

- A grande diferença destes modos 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.

Cipher Feedback Mode (CFB)$_{cont}$

- 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.

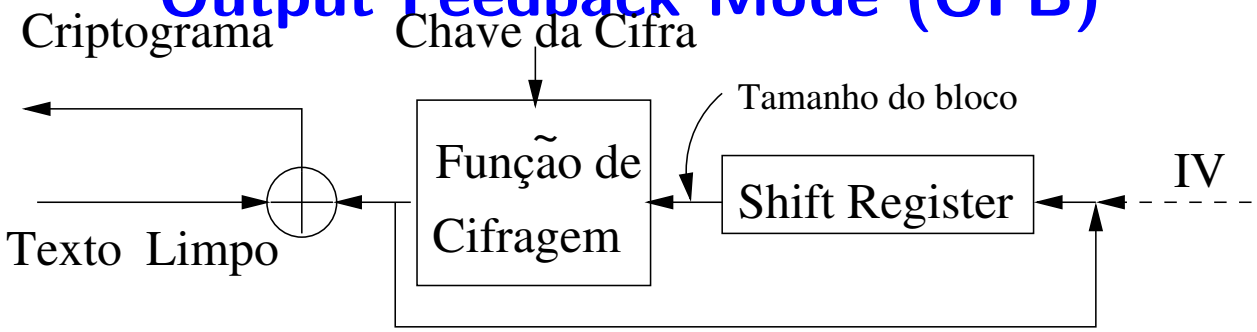
- Segurança
  - Os padrões do texto limpo são mascarados pela pseudo-aleatoriedade da sequência de chaves.
  - A alteração do IV é determinante. Como em todas as cifras sequenciais, a repetição de uma sequência de chaves torna a cifra vulnerável a ataques.

Cipher Feedback Mode (CFB)$_{cont}$

- 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.

Output Feedback Mode (OFB)$_{cont}$

- 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.

Padding

- As cifras por blocos 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.

Padding$_{cont}$

- 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.

Padding$_{cont}$

- Um aspecto importante dos últimos três métodos é a possibilidade de verificar a correcção da 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 geralmente muito 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 da comunicação.

- Num universo de N agentes é necessário acordar $N(N-1)/2$ chaves: uma chave para cada par de utilizadores. Ou não?

Pré-distribuição de Chaves Secretas

- A aproximação mais directa à comunicação segura entre N agentes, utilizando cifras simétricas é a geração e distribuição de todas as chaves secretas necessárias antes de se iniciar a troca de informação.

- A principal desvantagem desta aproximação é o enorme número de chaves que é necessário gerar, armazenar e proteger: $N(N-1)/2$.

- A única forma 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, e de as disponibilizar aos outros agentes quando são necessárias. Como?

- 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.

Chaves de Sessão

- No entanto, esta aproximação tem limitações que é impossível transpor:
  - uma vez que as chaves permanecem válidas durante a actividade do sistema elas estão expostas a ataques durante um tempo prolongado.
  - se uma chave for corrompida, e isso não for detectado, a comunicação entre o par de agentes correspondente deixa de ser segura.

- O conceito de **chave de sessão** resolve estes problemas:
  - É gerada uma chave secreta para cada troca de informação.
  - Esta chave é estabelecida entre emissor e receptor, utilizada para a troca de informação e depois é destruída.

- O estabelecimento das chaves de sessão pode ser feito de duas formas: a **distribuição de chaves**, e o **acordo de chaves**.

Distribuição de Chaves

- Num esquema de 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, e não com um intruso?

- 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. Voltaremos a este assunto mais tarde.

Distribuição de Chaves$_{cont}$

- As **chaves para cifragem de chaves** são geradas e pré-distribuidas por um canal seguro exterior ao sistema (e.g. em mão numa disquete, etc.) N chaves deste tipo que permitem ao TA comunicar com todos os agentes 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 secretas utilizadas para comunicação segura entre o TA e os agentes.

- 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. O seu princípio de funcionamento é o seguinte:
  - Dois agentes A e B pretendem comunicar utilizando uma cifra simétrica e, logo, estabelecer uma chave de sessão.
  - Existe um canal de comunicação aberto 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, por forma a ambos os agentes terem a garantia da **frescura da chave**.
  - Os dois agentes executam um conjunto de passos que lhes permitem confirmar que foi estabelecida uma chave de sessão adequada.

Acordo de Chaves$_{cont}$

- Em nenhum dos passos anteriores 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** que vamos estudar no próximo módulo.

- 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. Por exemplo:
  - 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: $\log_3 4 \pmod{11}$.

Exemplo: Diffie-Hellman$_{cont}$

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 $K = X^y \pmod{n} = Y^x \pmod{n}$.

Módulo I.B

Cifras Assimétricas

(Criptografia de Chave Pública)

Funções One-Way

- A ideia central à Criptografia de Chave Pública é a de uma função One-Way. Uma função deste tipo, é fácil de calcular, mas muito difícil (de preferência impraticável) de inverter i.e. dado x, $f(x)$ é fácil de calcular mas, dado $f(x)$, é muito dificil obter o x.

- 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? Primeiro, não há muitas funções deste tipo por aí. Segundo, suponhamos que dispomos de uma função verdadeiramente one-way: impossível de inverter. Se a utilizamos para cifrar informação, isso significa que nunca será possível recupera-la!

Funções One-Way com Trapdoor

- As funções utilizadas em criptografia pertencem a um subconjunto das funções one-way a que se chama funções **one-way com trapdoor**.

- Estas funções têm a particularidade de oferecerem uma porta de entrada secreta que permite a sua inversão de forma facilitada. O princípio de funcionamento destas funções é o seguinte:
  - 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

- Tal como acontece na criptografia simétrica, a função de cifragem é parametrizada com uma chave. Por outras palavras, a função $f(x)$ é, na realidade, uma família de funções indexada por uma chave K: $f_K(x)$.

- 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. Assim, a chave K denomina-se a **Chave Pública** da cifra.

- Associado a cada Chave Pública está um segredo k que dá o acesso à porta secreta da função $f(x)$: a **Chave Privada** da cifra. Esta chave 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

- Foi o primeiro algoritmo assimétrico que sobreviveu (e ainda sobrevive) ao escrutínio da comunidade científica. Surgiu no final dos anos 70.

- O funcionamento do RSA é muito simples de compreender: a sua segurança baseia-se na 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 $K \times k = 1 \pmod{\phi(m)}$.

Exemplo: RSA$_{cont}$

- 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 consegue 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

- A comunicação utilizando cifras assimétricas exige os seguintes passos:

  1. A Alice e o Bob combinam uma cifra assimétrica.
  2. O Bob envia à Alice, por 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, uma vez que a Chave Privada permanece na posse de um único agente.

- Num universo de N agentes, cada qual terá o seu par de chaves, podendo as chaves públicas ser armazenadas num servidor central no qual **não tem de haver o mesmo tipo de confiança absoluta**.

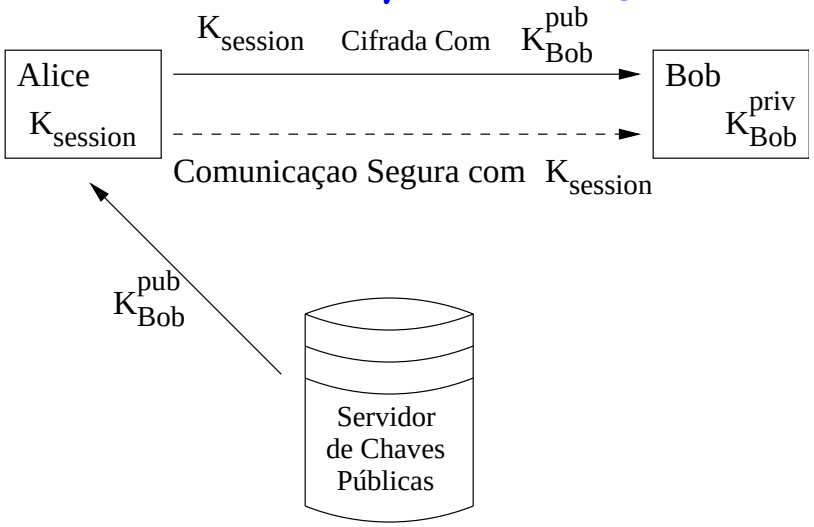
- Resta o problema de garantir que uma chave pública pertence a um determinado agente. Porquê?

Cifras Simétricas vs Cifras Assimétricas

- No mundo real as cifras assimétricas não são um substituto para as cifras simétricas. Isto por duas razões:
  - 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.
  - É sempre possível efectuar um ataque por texto limpo conhecido:
    * Se conhecemos um criptograma e a chave pública podemos, teóricamente, testar todas as possibilidades para o texto limpo até obtermos uma correspondência.
    * Quando os textos limpos possíveis são poucos, isto é um problema.

- As aplicações mais comuns das cifras assimétricas vêm complementar a funcionalidade das cifras simétricas e.g. **os sistemas de híbridos de distribuição de chaves de sessão**.

Sistemas Híbridos/Envelopes Digitais



- Neste esquema de distribuição de chaves desaparece a necessidade de um TA e do estabelecimento 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 sua Chave Privada.

Módulo I.C

Autenticação de Origem de Mensagens

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 simultâneamente 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: são funções que transformam um input de tamanho variável num output de tamanho (geralmente menor) fixo.

- Quando se pretende garantir integridade, o que interessa é 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: haverão sempre várias mensagens que serão mapeadas no mesmo valor de hash.

Funções de Hash Criptográficas$_{cont}$

- O que se consegue na realidade é uma identificação probabilística: é provável que o valor de hash tenha sido originado por aquela mensagem.

- Isto consegue-se utilizando as funções de hash ditas "**livres de colisões**" em que, além de não haver uma relação visível entre input e output:
  - É muito difícil encontrar duas mensagens que originem o mesmo valor de hash (**fortemente livre de colisões**).
  - Dado um valor de hash e a mensagem que o originou, é muito difícil arranjar outra mensagem que origine o mesmo valor de hash (**fracamente livre de colisões**).

- 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.

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**.

- Apesar de este algoritmo ser ainda seguro, uma série de avanços na criptoanálise de alguns dos seus rounds levou o autor a melhora-lo, desenvolvendo o MD5.

- Tanto o MD4 como o 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 $\{1, 0, 0, 0, \ldots\}$ por forma a obter um bit stream de comprimento $b' = 448 \pmod{512}$.

- À sequência anterior é adicionada uma representação de 64 bits do valor b, obtendo-se um bit stream de comprimento múltiplo de 512.

- Quatro registos de 32 bits (A, B, C e D), que no final conterão o output do algoritmo, são inicializados com valores constantes. Cada bloco de 512 bits da mensagem tratado como 16 sub-blocos de 32 bits (X_k).

- O processamento consiste em 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 e.g. $A = B + ((A + f(B, C, D) + X_k + t_i) <<< s)$ com $f(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$, e s,t_k variáveis.

Exemplo: Secure Hash Algorithm (SHA)

- O SHA foi desenvolvido pelas agencias governamentais americanas NIST e NSA para incluir no standard de assinaturas digitais DSS.
- O valor de hash produzido pelo algoritmo é de 160 bits.
- O SHA foi desenhado para que seja "*. . . computationally unfeasible to recover a message corresponding to a given message digest, 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 seguidos no seu desenvolvimento não foram divulgados.
- Hoje em dia o MD5 e o SHA são os algoritmos de hash tidos como mais seguros para aplicações criptográficas.

Message Authentication Codes

- Um Message Authentication Code (MAC) é uma função de hash criptográfica cujo resultado depende, não só do conteúdo da mensagem que é processada, mas também de uma chave secreta.

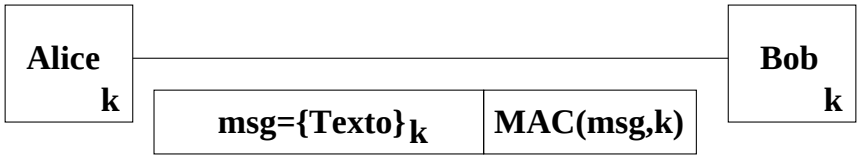
- Para gerar o MAC é necessário conhecer o algoritmo e a chave secreta. O mesmo acontece para verificar se o MAC é válido.

- Pode ser utilizado como uma impressão digital que garante que o checksum da mensagem foi calculado na sua origem: torna impossíveis alterações do checksum para mascarar uma modificação da mensagem.

- Outra aplicação possível é a 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?
  - Poder-se-ia pensar que se apenas Alice e Bob conhecem k, então nenhum intruso consegue alterar msg sem ser detectado na decifragem.
  - **Isto não é sempre verdade**: implica a existência de algum tipo de semântica ou redundância no texto limpo. E se for aleatório?
  - Incluindo um MAC na mensagem, é possível comprovar a origem da mensagem e a sua integridade, antes de a decifrar.
  - É também possível (e até mais frequente) utilizar MACs em aplicações que não requerem confidencialidade.

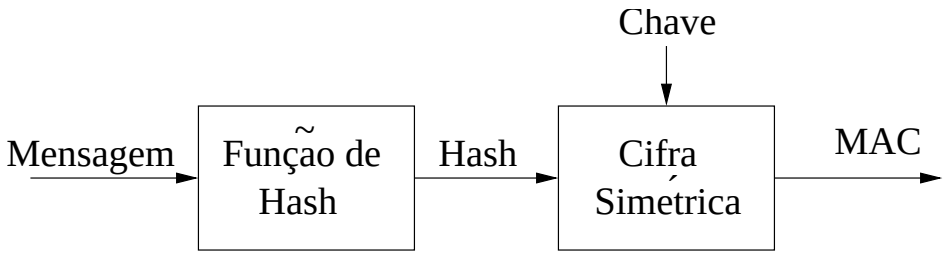


- Que garantias tem Bob quando recebe a mensagem?

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. Quais são as propriedades que se esperam de uma assinatura escrita?
  - A assinatura é **autêntica**: convence o receptor do documento de que o signatário explicitamente assinou o documento i.e. que conhecia o seu conteúdo, por ser o seu autor ou, simplesmente, por o aceitar.
  - A assinatura **não é falsificável**: prova que o signatário, e não outra pessoa, assinou o documento.
  - A assinatura **não é reutilizável**: faz parte do documento e não se pode transpor para outro documento.
  - A assinatura garante a **integridade do documento**: o documento permaneceu inalterado desde que a assinatura lá foi colocada.
  - A assinatura **não é repudiável**: o signatário não pode, à posteriori, negar que assinou o documento.

Assinaturas Digitais$_{cont}$

- 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.

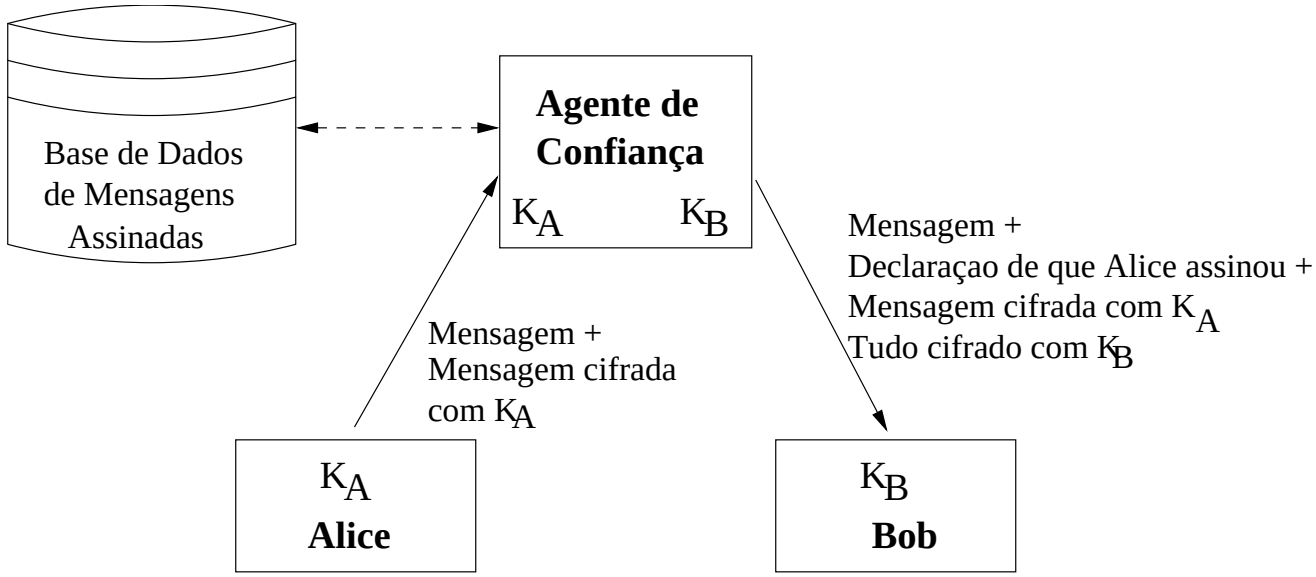
Assinaturas Digitais$_{cont}$

- **Uma assinatura digital é uma sequência de bytes cujo valor não tem significado fora do contexto de um algoritmo específico**. É gerada pelo signatário e acompanha o documento assinado, como anexo.

- Associados a um esquema de assinaturas digitais há sempre dois procedimentos complementares a considerar:
  - **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

- Assinaturas digitais com cifras simétricas e um agente de confiança (TA):



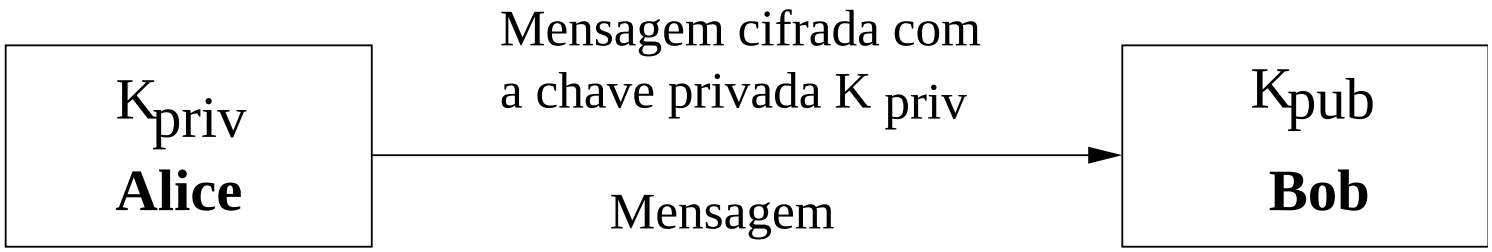
  - Este sistema é demasiado exigente para o agente de confiança, nomeadamente se se espera eficiência na comunicação entre muitos agentes, e no armazenamento das mensagens assinadas.

Assinaturas Digitais: Versão 1_{cont}

- 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: Versão 2_{cont}

- 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

- As assinaturas digitais baseadas na criptografia de chave pública dominam a criptografia actual, principalmente por não requererem um TA on-line.
- **Nem todos os algoritmos de assinatura digital são adaptações directas de algoritmos de cifra assimétrica**.
- Por esta razão 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

- A assinatura RSA é uma adaptação directa da cifra assimétrica RSA.
- Os cálculos matemáticos envolvidos são exactamente os mesmos, tendo em conta que a cifragem passa a ser efectuada utilizando a chave privada e que a decifragem passa a ser feita com a chave pública.
- Na geração da assinatura, o signatário simplesmente 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 um novo standard de assinaturas digitais de utilização livre: o Digital Signature Standard.

- 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 computacionalmente para facilitar a sua implementação em smartcards.

- O DSA baseia-se nas técnicas de chave pública El Gamal, cuja segurança se baseia no problema do logaritmo discreto (PLD).

Exemplo: Digital Signature Algorithm (DSA)$_{cont}$

- As principais criticas ao DSA foram as seguintes:
  - O DSA não pode ser utilizado para cifragem ou distribuição de chaves.
  - O DSA foi desenvolvido pela NSA, e os adeptos das teorias da conspiração acreditam que existe uma porta das traseiras.
  - O DSA é mais lento que o RSA, e este é um standard $de\ facto$
  - O desenvolvimento do DSA 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 geralmente muito pouco eficientes, o que torna pouco prática a assinatura de mensagens de tamanho realista.

- Por esta razão é mais comum a geração de assinaturas digitais a partir de impressões digitais das mensagens.

- O processo de geração da assinatura começa assim pelo cálculo de um valor de hash, que é depois assinado com a chave privada do signatário.

- 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 tem a vantagem adicional de tornar mais dificil o ataque à assinatura: o hash funciona como checksum da mensagem.

Assinaturas Digitais e Cifragem

- Combinando a assinatura digital com a cifragem obtém-se um protocolo que combina confidencialidade e autenticação.

- Este conceito é intuitivo: as cartas são também assinadas (prova de autoria) e depois inseridas num envelope (privacidade).

- **Faz mais sentido assinar antes ou depois de fazer a 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.

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.

Exemplo: Protocolo de Acordo de Chaves Station-to-Station (STS)$_{cont}$

- 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.

Módulo I.D

Autenticação de Entidades: Identificação

Introdução

- O problema da identificação consiste em um agente (o **identificado** ou A) demonstrar 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 os seguintes:
  - No caso de entidades honestas, B aceita a identidade de A.
  - B não pode apropriar-se da 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.

Introdução$_{cont}$

- A identificação pode fazer-se de acordo com três princípios:
  - **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.

- Os esquemas de identificação biométricos são aqueles que apelam mais ao imaginário. São exemplos de mecanismos de identificação deste tipo:
  - a identificação através da impressão digital,
  - do reconhecimento das feições ou
  - do reconhecimento da voz.

- Há, no entanto, problemas importantes associados à utilização deste tipo de sistemas, tanto pelos níveis de segurança que é possível obter, como no que diz respeito à sua implementação prática, e mesmo a nível legal.

Introdução$_{cont}$

- Em geral, os sistemas de identificação utilizados na prática não se baseiam na utilização de informação biométrica.

- Em alternativa, 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

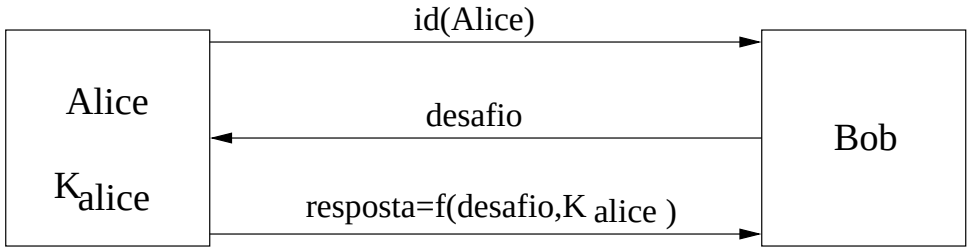
- Como demonstrar o conhecimento de informação privada ou secreta sem a divulgar explicitamente? Uma **prova de conhecimento zero** é um mecanismo que permite resolver este problema.

- É este tipo de protocolo que permite implementar um sistema de identificação ideal: demonstra-se o conhecimento do segredo associado à identidade sem revelar informação privada.

- Os protocolos que cumprem este princípio 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

- A Alice pretende identificar-se perante o Bob:

1. Parâmetros públicos: um número primo grande p, um número primo q divisor de $p-1$ e um número g tal que $g^q \pmod p = 1$.
2. A Alice gera um par de chaves: a chave privada é um número aleatório $0 < s < q$; a chave pública é $v = g^{-s} \pmod p$ e é enviada ao Bob.
3. A Alice escolhe um número aleatório $0 < r < q$, calcula $x = g^r \pmod p$ e envia x ao Bob.
4. O Bob envia à Alice um número aleatório $0 < e < 2^t - 1$: o desafio. t estabelece um nível de certeza, expresso em número de bits.
5. A Alice calcula a resposta $y = (r + s * e) \pmod q$ e envia-a ao Bob.
6. O Bob verifica que $x = g^y v^e \pmod p$.

- Mais uma vez convém notar que, para este sistema ser seguro, é necessário estabelecer a validade da chave pública da Alice: é necessário um sistema de certificação de chaves públicas.

Parêntesis

Trabalho Prático

Anonymous Digital Cash

Introdução

- O dinheiro físico acarreta muitos problemas, nomeadamente logísticos e de segurança, e por essa razão tem vindo a ser substituído em muitas utilizações por sistemas de pagamento automático.

- No entanto, o dinheiro físico tem uma propriedade que estes sistemas não podem fornecer, nomeadamente o anonimato/privacidade dos agentes.

- O sistema de dinheiro electrónico que se pretende implementar neste trabalho prático foi desenvolvido por Chaum em 1989 e tenta fornecer anonimato nas transacções monetárias electrónicas.

- Este sistema tem ainda outras propriedades interessantes, nomeadamente o facto de ser possível detectar e identificar agentes desonestos.

Pré-requisitos: Bit Commitment

- Os protocolos de Bit Commitment permitem a dois agentes resolver o seguinte problema:
  - O agente A pretende provar ao agente B que consegue prever um acontecimento com uma determinada antecedência.
  - O agente A pretende vender ao agente B esse segredo, mas não lho pode dizer antes de receber o dinheiro.
  - O agente B não quer pagar antes de ter a certeza de que o agente A fala verdade.

- Um Bit Commitment é um protocolo que permite a A comprometer-se com uma resposta, sem a divulgar a B.

- Depois de o acontecimento ter lugar, A consegue demonstrar a B que a resposta era correcta.

Pré-requisitos: Bit Commitment$_{cont}$

- Um protocolo de Bit Commitment pode ser construído com base numa função de hash criptográfica:
  - O agente A gera dois valores aleatórios e utiliza-os, conjuntamente com a resposta, para gerar um valor de hash: o bit commitment.
  - O agente B recebe o bit commitment e um dos valores aleatórios.
  - Na verificação, o agente A demonstra conhecer a resposta e o valor aleatório que permitem obter o valor de hash correcto.
  - Os valor aleatório de A permite evitar ataques por texto limpo conhecido, e também evitar que o agente B consiga verificar o bit commitment de forma unilateral.
  - O valor aleatório de B só é de facto necessário se a função de hash não for totalmente one-way.

Pré-requisitos: Assinatura cega

- Os algoritmos de assinatura cega permitem a dois agentes resolver o seguinte problema:
  - O agente A pretende obter uma assinatura de B num documento.
  - O agente B não pode conhecer o conteúdo do documento.

- Um algoritmo de assinatura cega consiste em quatro passos:
  - O agente A aplica uma máscara o documento a assinar.
  - O agente B assina essa versão ilegível do documento.
  - O agente A obtém uma assinatura do documento original com base na assinatura produzida por B.
  - Qualquer agente consegue verificar que a assinatura obtida por A é uma assinatura válida.

Pré-requisitos: Assinatura cega$_{cont}$

- Um protocolo de assinatura cega desenvolvido por Chaum em 1985 baseia-se na assinatura RSA:
  - O valor de hash a ser assinado é visto como um número x.
  - Esse número é mascarado com um valor aleatório, cifrado com a chave pública do signatário:

$$x' = x * r^{K_{pub}} \pmod{m} \quad (1)$$

  - A assinatura de B resulta em:

$$s' = r * x^{K_{priv}} \pmod{m} \quad (2)$$

  - A assinatura original obtém-se por:

$$s = s' * r^{-1} \pmod{m} = x^{K_{priv}} \pmod{m} \quad (3)$$

Anonymous Digital Cash: conceitos

- Num protocolo de Anonymous Digital Cash estão envolvidos três agentes: a Alice, que vai fazer um pagamento, o Bob, que vai receber o pagamento, e um banco onde ambos têm conta.

- As propriedades desejáveis para este tipo de protocolo são:
  - A segurança não depende da localização física ou das condições de armazenamento e transferência.
  - O ADC não pode ser falsificado ou reutilizado (gasto duas vezes).
  - A privacidade do utilizador está sempre protegida.
  - Não deverá ser necessária uma ligação on-line a um agente de confiança (e.g. banco) na altura do pagamento.
  - O ADC deverá poder ser transferido a outros utilizadores.
  - *Um item de ADC deverá poder ser dividido em itens de valor inferior (obviamente mantendo o valor total).*

Anonymous Digital Cash: Primeira Versão

- Uma primeira versão poderia ser a seguinte:

1. A Alice prepara 100 pedidos (orders) de $1000 cada uma.
2. Utilizando um algoritmo de assinatura cega, o banco recebe 100 pedidos para assinar.
3. O banco escolhe um pedido aleatoriamente.
4. A Alice abre todos os outros pedidos demonstrando que são pedidos correctos para $1000.
5. O banco assina de forma cega o pedido seleccionado e a Alice obtém o pedido assinado, que funciona como dinheiro. O dinheiro é retirado da conta da Alice.
6. A Alice entrega o dinheiro ao Bob, que verifica a assinatura do banco.
7. O Bob entrega o dinheiro ao banco que, verificando a sua assinatura, credita a conta do Bob.

Anonymous Digital Cash: Primeira Versão$_{cont}$

- A assinatura cega assegura o anonimato da Alice: o banco não consegue identificar uma nota que é depositada.
- Por outro lado, a assinatura elimina a possibilidade de falsificação.
- A escolha inicial torna estatisticamente improvável que a Alice consiga fazer o banco assinar um pedido de uma quantia superior.
- Este protocolo tem um grande problema: **não se consegue detectar, ou impedir o Bob e/ou a Alice de gastarem duas vezes o dinheiro**.

Anonymous Digital Cash: Segunda Versão

- Para detectar a batota poder-se-ia utilizar a seguinte adaptação:

1. A Alice prepara 100 pedidos (orders) de \$1000, **incluindo em cada uma, um número de série aleatório suficientemente grande**.
2. Utilizando um algoritmo de assinatura cega, o banco recebe 100 pedidos para assinar.
3. O banco escolhe um pedido aleatoriamente.
4. A Alice abre todos os outros pedidos demonstrando que são pedidos correctos para \$1000.
5. O banco assina de forma cega o pedido seleccionado e a Alice obtém o pedido assinado, que funciona como dinheiro. O dinheiro é retirado da conta da Alice.
6. A Alice entrega o dinheiro ao Bob, que verifica a assinatura do banco.
7. O Bob entrega o dinheiro ao banco que, verificando a sua assinatura e a unicidade do número de série, credita a conta do Bob.

Anonymous Digital Cash: Última Versão

- Para identificar quem fez batota, é necessário ir mais longe:
  - Utiliza-se uma técnica de divisão de segredos para incluir a identidade da Alice no dinheiro, sendo essa informação recuperável no caso de a Alice ser a batoteira. Caso isto não seja possível, o batoteiro é o Bob.
  - Cada um dos N pedidos de dinheiro iniciais passa a incluir N pares de bit strings $I_k = (I_k^1, I_k^2)$:
    - A Alice cria um string de identificação de acordo com as exigências do banco ID_{Alice}.
    - A Alice gera um número aleatório com tantos bits como a sua string de identificação H_k^1.
    - A Alice faz um XOR entre os dois bit strings, obtendo um terceiro $H_k^2 = ID_{Alice} \oplus H_k^1$.
    - Cada par de bit strings I_k é um par de bit commitments obtidos de H_k^1 e H_k^2.

Anonymous Digital Cash: Última Versão$_{cont}$

- Quando o banco pede à Alice para abrir $N-1$ pedidos, ela é obrigada a demonstrar que todos os I_k estão construídos correctamente.

- Quando a Alice paga ao Bob, este também obtém metade da identificação da Alice para cada um dos I_k:
  - O Bob verifica a assinatura do banco e, em caso de sucesso, envia à Alice um bit string b de comprimento N.
  - Consoante b_k seja zero ou um, a Alice envia ao Bob a informação que permite abrir o bit commitment I_k^1 ou I_k^2 respectivamente.
  - Isto permite verificar que a Alice foi a construtora inicial do pedido.
  - Note-se que o Bob passa a conhecer, para cada k, uma das metades da identificação da Alice: H_k^1 ou H_k^2, consoante b_k.

Anonymous Digital Cash: Última Versão$_{cont}$

- Quando o dinheiro é depositado, e é detectada uma reutilização, o banco consegue identificar o batoteiro:
  - Se o Bob entregar um bit string b igual ao previamente depositado, é com grande probabilidade ele o batoteiro.
  - Caso contrário, é altamente provável que um dos b_k permita recuperar a identidade da Alice:
    - Sejam b e b' os bit strings que o banco obteve em dois depósitos sucessivos.
    - Se b e b' são diferentes, então a Alice teve de divulgar, pelo menos para um determinado k, as duas metades do segredo que é a sua identidade.
    - O banco procura esse k e combina a informação apresentada pelos dois depositantes para recuperar a identidade da Alice: $ID_{Alice} = H_k^1 \oplus H_k^2$.