Universidade da Beira Interior
Faculdade de Engenharia
Departamento de Informática

Segurança Informática
Aula 3
Licenciatura em Engenharia Informática
Licenciatura em Informática Web
Licenciatura em Tecnologias e Sistemas da Informação

Computer Security
Lecture 3
Degree in Computer Science and Engineering
Degree in Web Informatics
Degree in Information Technologies and Systems

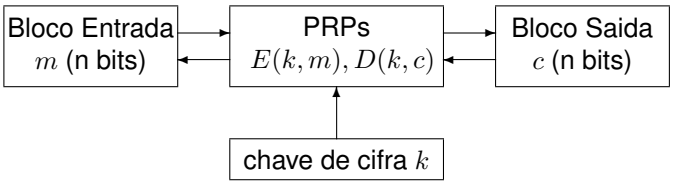
Sumário
Estudo de cifras de chave simétrica por blocos, nomeadamente dos modos de operação conhecidos por *Electronic Code Book*, *Counter Mode* e *Cipher Block Chaining*. Discussão acerca da implementação de cifras aleatorizadas, bem como da motivação para as construir. Análise de dois esquemas para distribuição de chaves de cifra.

Summary
Study of block ciphers, namely of the operation modes referred to as Electronic Code Book, Counter Mode *and* Cipher Block Chaining. *Discussion concerning the implementation of randomized encryption schemes, as well as of the underlying motivation to build them. Analysis of schemes for the distribution of encryption keys.*

1 Cifras de Chave Simétrica por Blocos
Block Ciphers

1.1 Introdução
Introduction

As **cifras de chave simétrica por blocos** são consideradas por muitos como **uma ferramenta fundamental**[1] **da criptografia**. De uma forma informal, pode dizer-se que estas cifras **elaboram em funções** designadas por *permutações pseudo aleatórias* (**Pseudo Random Permutations (PRP)**), que **aceitam** como valor de entrada **uma chave e um bloco de bits**, com tamanho fixo, e **devolvem um bloco com o mesmo tamanho**.



As PRPs são um subconjunto das funções pseudo aleatórias (Pseudo Random Functions (PRFs)) em que **o espaço de entrada e de saída são iguais**. As PRPs definem-se, assim, apenas sobre **dois** espaços $(\mathcal{K}, \mathcal{X})$:

$$E : \mathcal{K} \times \mathcal{X} \rightarrow \mathcal{X},$$

tal que:

1. **existe um algoritmo deterministico eficiente** para avaliar $E(k, x)$;
2. para cada k, **a função** $E(k, .)$ **é bijetiva** (i.e., é uma função um-para-um, em que cada elemento do espaço $\mathcal{X}$ é mapeado para um e um só elemento do espaço $\mathcal{X}$ e vice-versa);
3. **existe um algoritmo** determinístico eficiente **para inversão** $D(k, y)$.

Os **exemplos canónicos** que se costumam mencionar quando se lecionam cifras por blocos são:

- o **Data Encryption Standard (DES)**, cujas chaves de cifra são de 56 bits (7 bytes) e o tamanho do bloco é de 64 bits (8 bytes);
- o **Triple DES (3DES)**, cujas chaves de cifra são de 168 bits (21 bytes)) e o tamanho do bloco é de 64 bits (8 bytes);
- e o **Advanced Encryption Standard (AES)** —o ***standard* atual para cifras de chave simétrica por blocos**— que aceita chaves de cifra com 128 (8 bytes), 192 (12 bytes) ou 256 bits (16 bytes) e o tamanho do bloco é de 128 bits (16 bytes).

Se particularizarmos a definição de PRP para a cifra AES (que usa como base a cifra Rijndael) ou 3DES, obtemos as funções E e D definidas sobre os seguintes conjuntos:

AES-128: $\mathcal{K} \times \mathcal{X} \rightarrow \mathcal{X}$ onde $\mathcal{K} = \mathcal{X} = \{0,1\}^{128}$ e

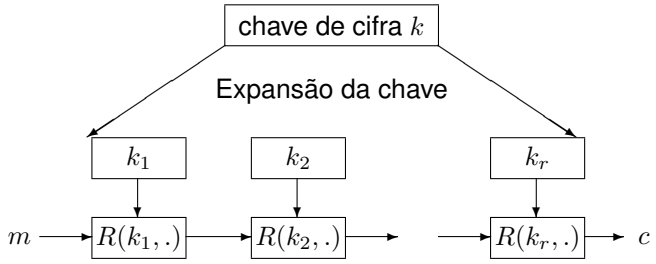
3DES: $\mathcal{K} \times \mathcal{X} \rightarrow \mathcal{X}$ onde $\mathcal{X}\{0,1\}^{64}$ e $\mathcal{K}\{0,1\}^{168}$.

O **modelo de construção** das cifras de chave simétrica por blocos é normalmente representado conforme se mostra na figura seguinte. Tipicamente, o algoritmo

[1] Dan Boneh, nas suas aulas de criptografia, entitula-as como as *crypto work horse*.

de cifra ou de decifra **começa por aplicar funções de expansão da chave de cifra**, dando origem a várias chaves de ronda k_i que, por sua vez, são **usadas como segundo parametro de entrada** nas funções núcleo destas cifras, também elas chamadas de **funções ronda (**$R(k_i, .)$**)**. **Cada uma destas funções é invertível**, e aplica uma transformação ao resultado da função anterior, até que o bloco de criptograma (ou bloco pseudo aleatório) seja produzido.

A **decifra** faz-se **aplicando as chaves e as funções de ronda no sentido inverso**.



O **DES define 16 rondas** para a cifra de um bloco, enquanto que o **3DES define 48**. O **AES define 10, 12 ou 14 rondas** para chaves de **128, 192 ou 256 bits**, respetivamente. Alguns números relativos à eficiência computacional do 3DES e do AES são apresentadas no final deste capítulo.

1.2 Intuição acerca da Segurança
Security Intuition

A construção destas cifras elaboram nos métodos da **confusão** e **difusão** referidos por Shannon para construção de boas cifras práticas. As funções ronda e de expansão de chave devem ser construidas com o propósito de obscurecer a relação com a chave de cifra e **dissipar as propriedades estatísticas do texto-limpo pelo criptograma**.
Note-se que, mesmo que a função PRP seja segura e assegure as duas propriedades anteriores para cada bloco, o mesmo pode não acontecer para a mensagem toda, conforme se prova na secção seguinte.

De um modo geral, **para que uma PRP** $F(k, .)$ **seja segura, esta deve ser indistinguível de uma qualquer outra função** $F(.)$ **escolhida ao acaso da totalidade de funções de** $\mathcal{X}$ **para** $\mathcal{X}$. Por outras palavras, deve ser computacionalmente inviável a um adversário distinguir entre a escolha de uma destas funções, determinadas pela chave, contra a escolha de uma qualquer função definida para aqueles espaços.

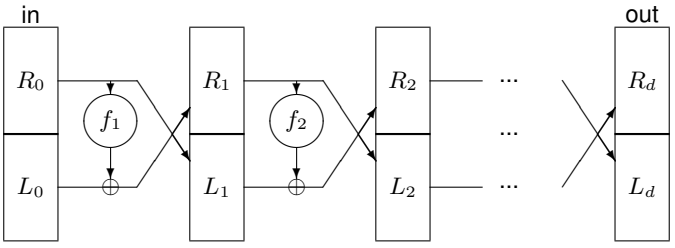
1.3 Rede de Feistel
Feistel Network

Uma das mais **bem sucedidas cifras de chave simétrica por blocos é o DES**[2]. O núcleo desta cifra é constituído por **Redes de Feistel**, que são **funções sempre invertíveis**, mas **construidas a partir de funções que podem não ter inversa**.

Horst Feistel desenhou a cifra **Lucifer** enquanto trabalhava na *International Business Machines* (**IBM**) *Corporation* no **início dos anos 70**. O **DES é uma simplificação dessa cifra**, que operava originalmente com **blocos e chaves de 128 bits**. Em **1973**, o então *National Bureou of Standards* pediu que fossem feitas propostas para cifras por blocos para ser usada em computadores vendidos ao governo, tendo **a IBM submetido uma variante (simplificada) da Lucifer**. Em **1976**, a cifra DES foi adotada como uma **norma federal**, mas o tamanho dos blocos era de **64 bits e as chaves de 56 bits**.

O **pequeno tamanho das chaves** de cifra do DES foi, de resto, o facto que ditou a **quebra da cifra em 1997** por busca exaustiva (ataque de força bruta). O DES foi primeiro substituído pelo **3DES** e depois pelo **AES em 2000**, após um concurso aberto pelo *National Institute of Standards and Technology* (NIST).

As redes de Feistel são, na verdade, **uma forma bastante engenhosa de construir PRPs a partir de PRFs**. Formalmente, pode-se dizer que, se $f_1, ..., f_d$ forem quaisquer funções definidas de $\{0,1\}^n$ para $\{0,1\}^n$, então o circuito (ou rede) representado a seguir define uma função $F : \{0,1\}^{2n} \to \{0,1\}^{2n}$ invertível:



Repare-se que a inversão é dada pela aplicação iterativa de:

$$F_i^{-1} = \begin{cases} R_{i-1} = L_i \\ L_{i-1} = f_i(L_i) \oplus R_i \end{cases} \quad \text{para } i = d, d-1, ..., 1$$

Mais que isso, **um teorema de Luby–Rackoff**, de 1985, assegura que **qualquer rede de Feistel com mais do que 3 rondas** construida a partir de **PRFs seguras define um PRP seguro**, i.e., se

$$f : K \times \{0,1\}^n \to \{0,1\}^n$$

é um PRF seguro, então a função F resultante de uma rede de Feistel de 3 rondas

$$F : K^3 \times \{0,1\}^{2n} \to \{0,1\}^{2n}$$

é uma PRP segura. **O DES especifica 16 rondas**, enquanto que o 3DES usa 48. O **AES não usa redes de Feistel** no seu núcleo.

[2]Largamente aceite e usado por instituições bancárias.

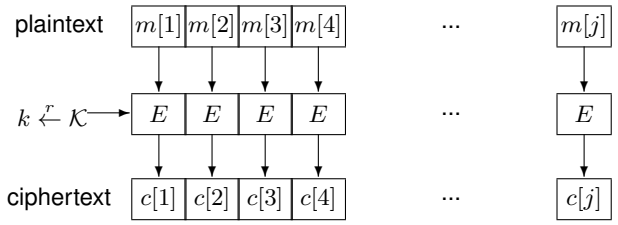
1.4 Modos de Cifra
Cipher Modes

Foi dito antes que as cifras de chave simétrica por blocos operam sobre uma primitiva que aceita uma chave k e um bloco m_i de tamanho fixo da mensagem a cifrar. É óbvio que as mensagens não serão sempre do mesmo tamanho que o bloco e que tenhamos que, então, elaborar em esquemas que usem aquela primitiva e permitam cifrar ou decifrar mensagens de tamanho arbitrário. Na literatura da especialidade, estes esquemas são conhecidos por **modos de cifra**, e definem os circuitos ou algortimos de cifra e de decifra. **Nesta aula falaremos de 4 modos diferentes:**

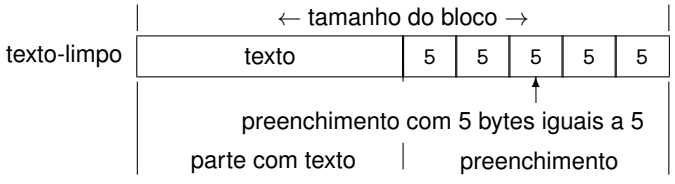
1. *Electronic Code Book*;
2. *Deterministic Counter Mode*;
3. *Cipher Block Chaining* e;
4. *Randomized Counter Mode*.

1.5 Electronic Code Book
Electronic Code Book

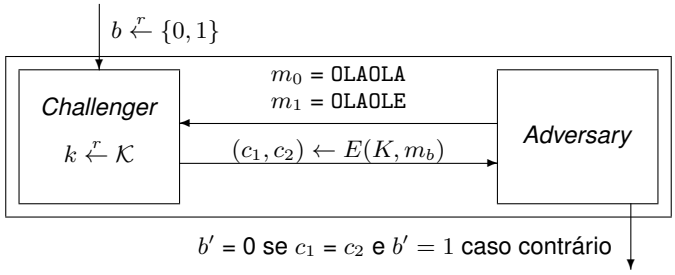
O **modo mais simples e mais direto (e também normalmente errado)** de usar uma cifra de chave simétrica por blocos é **conhecido por *Electronic Code Book* (ECB)**. Neste modo, e visto que este tipo de cifra só opera sobre blocos de tamanho fixo n, a **mensagem** a cifrar é **simplesmente dividida em blocos de tamanho** n, e **cada bloco é cifrado independentemente dos outros, e com a mesma chave**, conforme se mostra na figura seguinte.



Nota: já que o tamanho da mensagem pode não ser um **múltiplo do tamanho do bloco** n, o **último bloco (**m_j**) pode ter de ser preenchido** até que a condição se verifique (i.e., *padding*). Normalmente, tal consegue-se repetindo o número de bytes que foram necessários para preencher o bloco na parte final do bloco incompleto:



Ao contrário da *one time pad* ou de outras cifras de chave simétrica contínuas construidas sobre um gerador seguro, **a cifra que usa o modo ECB não é**, de uma maneira geral, **semanticamente segura no modelo COA**. Neste caso, **o adversário pode sempre ganhar o jogo** que se representa em baixo, elaborando duas mensagens: **a primeira com dois blocos de texto iguais**; e **a segunda com dois blocos diferentes**. Por exemplo, se o tamanho dos blocos for 3 bytes, a primeira mensagem pode ser `OLAOLA` e a segunda `OLAOLE`.



Este ataque vale-se do facto de **blocos iguais do texto-limpo darem origem a blocos iguais no criptograma (se a mesma chave for usada)**. Repare-se que o adversário nem precisa decifrar o criptograma para ganhar o jogo, o que significa que, em termos práticos, o uso descuidado deste modo pode ditar que um atacante descubra, por exemplo, que partes de um ficheiro são iguais, o que pode ser um problema para espaços de mensagens relativamente pequenos.

As vantagens e desvantagens mais interessantes deste modo de cifra podem ser resumidos da seguinte forma:

	Segurança
-	**Não é semanticamente seguro no modelo COA e pode apenas ser usado para cifrar pedaços de informação aleatórios e tipicamente mais pequenos que o tamanho do bloco** (e.g., outras chaves de cifra).
-	Os **padrões existentes no texto limpo não são disfarçados**, já que um bloco cifrado duas vezes com a mesma chave resulta em criptogramas iguais.
-	Suscetível a ataques por *code book* (compilação de pares texto limpo/criptograma).
-	Suscetível a ataques por remoção, troca e repetição de blocos.
	Eficiência
+	**Permite o acesso aleatório a dados cifrados**, i.e., qualquer bloco pode ser decifrado independentemente.
+	Pela mesma razão, permite o **processamento paralelo** da informação.
-	Não há possibilidade de efectuar pré-processamento.
-	O último bloco necessita sempre de tratamento em termos de preenchimento.

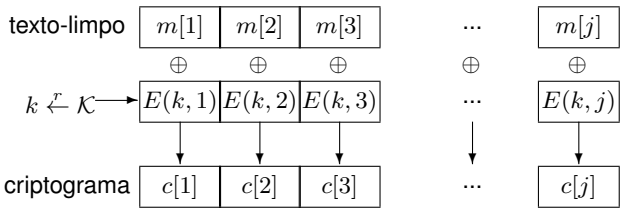
	Tolerância a Erros
+	Este modo **não apresenta problemas de propagação de erros entre blocos**, i.e., um erro afecta apenas um bloco de texto limpo.
-	Erros de sincronização (**perda de bits**) **são irrecuperáveis**.

1.6 Modo Contador Determinístico
Deterministic Counter Mode

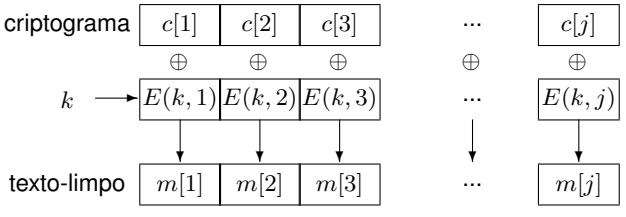
Na secção anterior foi demonstrado que o modo ECB não é semanticamente seguro perante o modelo de ataque mais simples de todos –COA. O problema é que dois blocos iguais são cifrados sempre da mesma forma quando se usa este modo de cifra. É possível apresentar **uma construção baseada em PRPs simples que resolve esse problema**, mas que **não é semanticamente segura no modelo de ataque seguinte**: o CPA, em que o atacante tem acesso a vários pares conhecidos texto-limpo/criptogramas. Repare-se que isto pode acontecer facilmente na realidade, bastando para isso que um atacante esteja a escutar uma comunicação onde passem várias mensagens cifradas com a mesma chave.

O **modo contador determinístico define uma cifra de chave simétrica contínua a partir de uma cifra por blocos**, fazendo uso apenas da função de cifra $E(k,.)$, que é inicializada com uma chave e reincidentemente usada para gerar blocos pseudo aleatórios, através da cifra dos valores de um contador que começa, por exemplo, com 1. Estes blocos são depois somados módulo 2 com o texto a cifrar, conforme se mostra na figura seguinte.

O **algoritmo de cifra** do modo CTR pode ser esquematizado da seguinte forma:

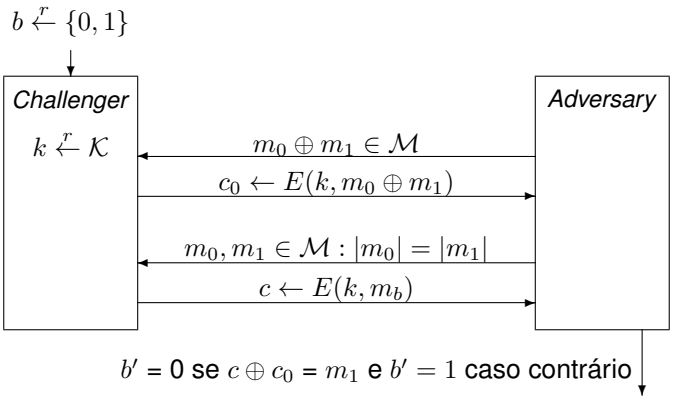


A decifra é exatamente igual à cifra, trocando apenas o criptograma de lugar com o texto-limpo no esquema final. **Note que**, na figura seguinte, **a chave já não é aleatória, mas sim determinada**.
De forma análoga ao que foi feito em cima, o **algoritmo de decifra** para o modo CTR pode ser representado como se segue:



Note, no modo CTR determinístico, que **o algoritmo de decifra é igual ao algoritmo de cifra** (porque, no fundo, estamos a usar a cifra de chave simétrica por blocos para construir uma cifra de chave simétrica contínua).

Apesar de ter **resolvido o problema associado ao ataque COA**, esta construção ainda não é ideal, porque agora **se cifrarmos a mesma mensagem com a mesma chave, teremos também dois criptogramas iguais**. O jogo que o adversário conseguiria facilmente ganhar está representado em baixo. Note que tanto o número da experiência ($b = 0$ ou $b = 1$) é decidido antes de começar o desafio.



No ataque antes representado, o adversário pode pedir que lhe seja cifrado um texto antes de responder ao desafio. Neste caso, é pedida a cifra de $m_0 \oplus m_1$ inicialmente, em antecipação ao que este adversário tem preparado a seguir. **Na altura do desafio, o adversário envia as duas mensagens em separado, sendo que já conhece a cifra da sua soma módulo 2**, pelo que lhe é simples diferenciar ambos os casos. Note-se que a cifra de $m_0 \oplus m_1$ é $m_0 \oplus m_1 \oplus r$, em que r é uma *pad* pseudo aleatória. Note-se também que a chave de cifra k não muda durante o desafio (portanto o *pad* r também não muda). Durante o desafio, o adversário pede a cifra de duas mensagens, diferentes da primeira (de acordo com as regras do jogo), mas relacionadas com aquela. O *Challenger* devolve $c = m_0 \oplus r$ ou $c = m_1 \oplus r$. A soma módulo 2 de c com c_0 dá m_1 ou m_0, assim tenha sido cifrada m_0 ou m_1, respetivamente. O adversário ganha o jogo dessa forma, sem nunca quebrar quaisquer regras, já que **todas as mensagens são diferentes**.

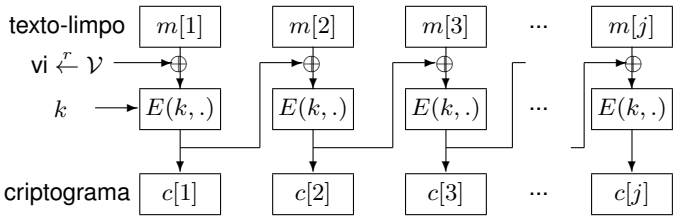
1.7 Cifras Aleatorizadas - Modo *Cipher Block Chaining*
Randomized Encryption - Cipher Block Chaining

O problema sublinhado no final da secção anterior aponta diretamente na direção em que a solução deve ser procurada. **A solução passa por garantir que**, mesmo que se cifre a mesma mensagem duas vezes com a mesma chave, **o criptograma deve ser diferente de ambas as vezes**. **É**, por isso, **necessário adicionar um fator de aleatoriedade no processo de cifragem** que cause o efeito desejado. As duas últimas construções (modos de cifra) apresentadas nesta parte são ci-

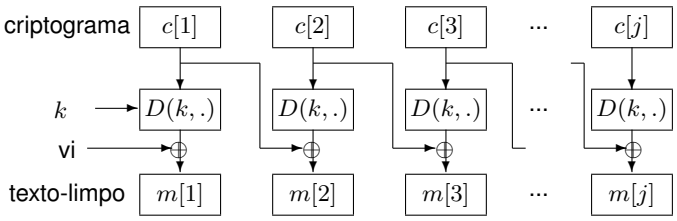
fras aleatorizadas, tendo que ser instanciadas com um parâmetro adicional aqui **designado de vetor de inicialização** (VI).

O VI é um valor que deve ser potencialmente único para cada combinação chave de cifra/mensagem a cifrar mas **não precisa ser secreto, podendo ser enviado juntamente com o criptograma** no contexto de uma comunicação (e.g., no início do criptograma). O requisito de ser *potencialmente único* é normalmente conseguido através da escolha aleatória de valores de um espaço $\mathcal{V}$ suficientemente grande, e.g., $\mathcal{V} = \{0,1\}^{128}$. O modo *Cypher Block Chaining* (CBC) caracteriza-se pelos circuitos de cifra e de decifra incluidos a seguir.

O **circuito de cifra** do modo CBC pode-se representar como mostra a figura seguinte:



O **circuito de decifra**[3] pode-se representar da seguinte forma:



As vantagens e desvantagens mais interessantes deste modo de cifra podem ser resumidas da seguinte forma:

	Segurança
+	É **semanticamente seguro no modelo CPA**, **desde que o VI seja imprevisível** (apesar de poder ser público).
+	**Os padrões do texto limpo são mascarados pelo xOR** e pelo **efeito cascata**.
+	De textos limpos iguais resultam criptogramas distintos, **inviabilizando ataques por *code book* e por repetição**.
+-	Alguns ataques por **manipulação de blocos são detetáveis**.
-	Ataques por manipulação do IV podem não ser detetáveis.

[3]Note-se que, na decifra, tanto a chave como o VI já são valores determinados e conhecidos.

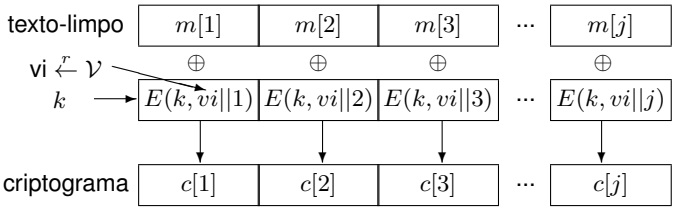
	Eficiência
+	**Permite o acesso aleatório a dados cifrados**, i.e., se o bloco anterior for conhecido, qualquer bloco pode ser decifrado sem decifrar os anteriores.
+-	Permite o **processamento paralelo da informação cifrada**, mas não na cifragem.
–	Não há possibilidade de efectuar pré-processamento e **cada alteração ao texto limpo**, e.g., num ficheiro, **implica uma nova cifragem completa**.
-	O último bloco necessita sempre de tratamento em termos de preenchimento.
	Tolerância a Erros
-	**Um erro num bit** do criptograma **afeta o bloco de texto limpo correspondente, e um bit no bloco seguinte.**
-	Erros de sincronização (**perda de bits**) **são irrecuperáveis**.

O modo CBC é um dos **modos mais populares** de cifras por blocos, sendo **sobretudo usado para proteger ficheiros em disco**.

1.8 Cifras Aleatorizadas - Modo Contador Aleatório
Randomized Encryption - Randomized Counter Mode

Apesar de ter bastantes vantagens, o **modo CBC** é um pouco **avesso à paralelização**. Um dos modos que endereça este problema sem adicionar complexidade ao algoritmo de cifra e de decifra é designado por *Randomized Counter* (CTR) *Mode*.

Este modo é **bastante semelhante ao que foi apresentado antes** como Modo Contador Deterministico, à exceção da forma como a *pad* de cifra é gerada. No (*randomized*) CTR, **o algoritmo de cifra define que se gere um vetor de inicialização aleatório e potencialmente único para cada combinação chave/mensagem** (k, m). Este vetor é concatenado a um contador, que é posteriormente iterado e avaliado pela função de permutação sucessivamente para gerar uma sequência de bits pseudo aleatórios que podem ser somados módulo 2 com o texto limpo para obter o criptograma, conforme se representa a seguir:



Repare-se que, tal como para o modo CTR determinístico, **o algoritmo de decifra é igual ao de cifra**, trocando apenas as posições do texto limpo e do critptograma (e no algoritmo de decifra, o vetor de inicialização é já conhecido, em vez de aleatório). O facto d**os dois algorit-**

mos serem iguais constitui também uma vantagem, já que tal facto torna **a cifra mais simples e eficiente de implementar**.

Repare que nos diagramas que definem o modo de operação CBC e CTR, nomeadamente para o algoritmo de cifra, é **formalizado o facto da chave de cifra não estar a ser gerada aleatoriamente para cada mensagem**. **Apenas o VI é que é gerado dessa forma.**

Acredita-se que **este modo de cifra é uma das melhores**, senão a melhor construção que recorre a cifras por blocos. As vantagens e desvantagens mais interessantes deste modo de cifra podem ser resumidas da seguinte forma:

	Segurança
+	É **semanticamente segura no modelo CPA**.
+	Os **Os padrões do texto limpo são mascarados por se imitar uma cifra de chave simétrica contínua**.
+	De textos limpos iguais resultam criptogramas distintos (IV é diferente), **inviabilizando ataques por *code book* e por repetição**.
-	Ataques por **manipulação de bits não são detetáveis** (maneabilidade).
	Eficiência
+	**Permite o acesso aleatório a dados cifrados**.
+	Permite o **processamento paralelo da informação**.
+	É possível efectuar a geração do *pad* de cifra antecipadamente e em paralelo, pelo que a cifragem e decifragem podem tornar-se muito eficientes.
+	Por **atuar como uma cifra de chave simétrica contínua**, o **último bloco não necessita de** tratamento em termos de **preenchimento**.
	Tolerância a Erros
+	Neste modo **não há propagação de erros**. Um erro no criptograma afeta apenas um bit no texto limpo.
-	Erros de sincronização (**perda de bits**) **não são recuperáveis**.

1.9 Eficiência Computacional
Computational Performance

De modo a obter uma visão global da *performance* dos algoritmos de chave simétrica, a tabela que mostrava a velocidade de geração de números para cifras de chave simétrica contínuas foi complementada, com os respetivos valores para cifras de chave simétrica por blocos, e transcrita para aqui. Estes valores dizem respeito à implementação em C++ de Wei Dai na biblioteca Crypto++ 5.6.0, e foram obtidos numa máquina AMD Opteron com um processador a 2.2 GHz com sistema operativo Linux:

Cifras Simétricas

Continuas	**Chave**		**Velocidade**
RC4	128 bits		126 MB/sec
eStream Salsa20/12	128/256 bits		643 MB/sec
eStream Sosemanuk	128/256 bits		727 MB/sec
Por Blocos		**Bloco**	
3DES	168 bits	64 bits	13 MB/sec
AES-128	128 bits	128 bits	109 MB/sec

1.10 Nota Final
Final Remark

Apesar das cifras e modos de cifra discutidos anteriormente serem semanticamente seguros (alguns deles para o modelo CPA), **ainda pouco foi dito acerca de ataques de manipulação de criptogramas**. Em tais cenários, **o adversário tem a capacidade de interceptar e alterar criptogramas antes de os reenviar para o destino final**. Os modos **CBC e CTR não resolvem estes problemas**, sendo necessário a inclusão de **mecanismos de integridade para além da cifra**. Isto será assunto das próximas aulas.

2 Esquemas de Distribuição de Chaves de Cifra
Schemes for Distribution of Encryption Keys

2.1 O Problema Subjacente
The Underlying Problem

Anteriormente foram discutidas várias formas de cifrar dados e alguns conceitos e princípios que ajudam a definir a segurança dos algoritmos que lhes estão associados. Até agora foram discutidas, sobretudo, cifras de chave simétrica contínuas e por blocos, estando ainda em aberto o estudo de cifras de chave assimétrica. Foi dito que, se uma cifra está bem construida, então **a segurança do sistema criptográfico vai depender**, em última análise, **do secretismo da chave de cifra** (no caso das cifras de chave simétrica) **ou da chave de cifra privada** (no caso das cifras de chave pública).

Se quisermos transmitir informação confidencial sobre **um canal potencialmente inseguro**, **geramos uma chave de cifra[4] e ciframos essa informação antes de a transmitir**. O problema está em como se troca a chave que foi usada para cifrar, ou como se resolve a seguinte dicotomia:

- **Por um lado, o canal é inseguro.**
- **Por outro, ambas as entidades envolvidas na comunicação têm de ter a chave** que, por sí só, é **confidencial também**.

[4]Ou então um elemento de aleatorização, se a chave já tiver sido trocada.

Mais, como a criptanálise de uma cifra é normalmente difícil, **os ataques são frequentemente focados no comprometimento da chave de cifra ou da forma como é distribuída** (i.e., é mais fácil começar por aqui). É principalmente devido a este facto que **a gestão das chaves é um problema de importância crítica em criptografia** –as chaves têm de ser mantidas secretas antes, durante, e após a comunicação em que são usadas.

A **abordagem mais direta** para resolver o problema enunciado antes consiste em gerar e **distribuir as chaves de cifra antes da comunicação**, mas existem outras formas. No âmbito deste curso, podem-se apontar **2 esquemas principais** (simétricos) e **2 protocolos de acordo de chaves** (da designação inglesa *Key Agreement Protocols* (KAPs)) importantes:

- Esquemas
  1. **Todos os pares** partilham uma **chave de cifra de chaves de sessão**.
  2. Todas as entidades partilham uma **chave de cifra de chaves de sessão** com um **Agente de Confiança**.
- Protocolo de Acordo de Chaves
  1. **Diffie-Hellman** KAP, em que duas entidades conseguem **estabelecer uma chave de sessão através do diálogo** (i.e., troca de mensagens) sobre um canal público, **independentemente** de terem antes estabelecido ou não alguns parâmetros ou segredos. O facto do objetivo ser conseguido através de troca de mensagens faz deste método um protocolo, ao invés de um esquema;
  2. **Puzzles de Merkle**.

Para além dos acima referidos, é ainda possível indicar **algoritmos de cifra assimétrica como meio de troca de chaves de sessão**. Esta forma de trocar chaves é, de resto, **bastante popular atualmente**, sendo usado, por exemplo, nas ligações sobre o protocolo *Transport Layer Security* em que o servidor tem um certificado e se autentica. É comum, por exemplo, usar o algoritmo de cifra RSA para trocar chaves de sessão.

A seguir descrevem-se apenas os dois esquemas de trocas de chaves de sessão, ficando a discussão dos KAPs Diffie-Hellman e Puzzles de Merkle para a aula relativa à criptografia de chave pública.

2.2 Chave de Sessão, Chave de Cifra de Chaves de Sessão e Agente de Confiança

Session Key, Session Key Encryption Key and Trusted Party

Antes de prosseguir, há pelo menos três conceitos que importa esclarecer, e que foram realçados a negrito antes:

Chave de Sessão Uma chave que é usada para cifrar e decifrar, decifrar gerada somente para determinada comunicação (e depois destruída), e que é, ela própria, cifrada com uma chave de cifra da chave de sessão.

Chave de Cifra da Chave de Sessão Uma chave usada única e exclusivamente para cifrar e decifrar chaves de sessão.

Agente de Confiança **Uma entidade** do sistema de comunicações **em quem** os utilizadores **confiam plenamente** (i.e., não se acredita que pode ser comprometida e age sempre de forma idónea a correta) e com quem podem ter chaves pré estabelecidas.

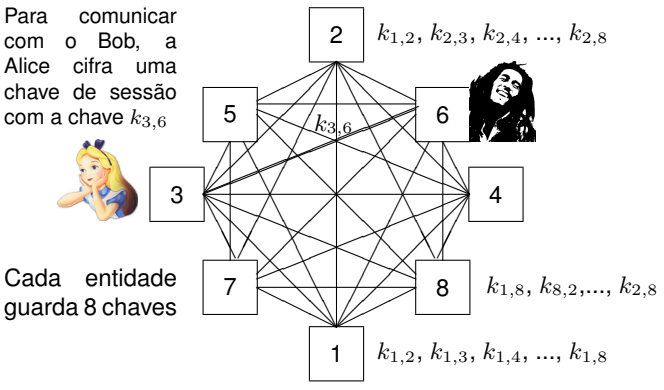
2.3 C_2^n Chaves de Cifra de Chaves de Sessão

C_2^n Keys for Encryption of Session Keys

Considere o cenário em que temos n ($n = 1, 2, ...$) entidades a tentar comunicar usando uma infrastrutura potencialmente insegura, e pergunte-se quantas chaves precisaria para estabelecer canais cifrados entre cada duas dessas entidades:

- Para $n = 2$ (duas entidades), precisaríamos de 1 chave.
- Para $n = 3$, precisaríamos de 3 chaves.
- Para $n = 4$, precisaríamos de 6 e, por indução matemática, precisaríamos de C_2^n chaves para n entidades (e.g., para 10 entidades, já precisávamos de $45 = C_2^{10} = 10 \times 9/2$ chaves).

Cada entidade teria **1 chave para cada uma das outras entidades no sistema**, o que parece, de um ponto de vista muito geral, ótimo e seguro. O problema é que **são muitas chaves para pré-distribuir e gerir localmente**. Para minimizar o impacto desta solução, **estas chaves seriam usadas para cifrar as chaves de sessão** que, por sua vez, seriam usadas para proteger ligações momentâneas entre entidades. Para obter uma ideia visual deste esquema, atente na figura seguinte:



No caso de existirem C_2^n chaves pré-distribuídas, a **troca de chaves de sessão é simples**: **uma das duas entidades** envolvidas na comunicação **gera a chave de ses-**

são e envia-a à outra entidade cifrada com a chave pré-estabelecida. Formalmente:

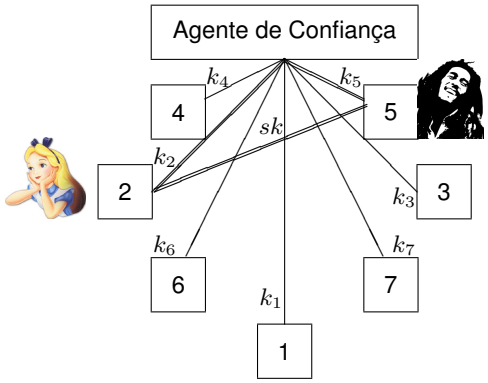
1. Alice (A): $sk \xleftarrow{r} \mathcal{K}$; $c \leftarrow E(k_{3,6}, sk)$;
   A Alice gera uma chave de sessão aleatoriamente e cifra-a com a chave que partilha com Bob.
2. A $\rightarrow$ Bob (B): c;
   A Alice envia ao Bob o criptograma resultante da cifra de sk com a chave pré-estabelecida $k_{3,6}$.
3. B: $sk = D(k_{3,6}, c)$.
   Bob recebe e decifra o criptograma, obtendo a chave de sessão.

Para que todo este **esquema funcione** é necessário que as C_2^n **chaves sejam trocadas por um canal seguro ou entregues manualmente** (usando, e.g., um **disco amovível**) **previamente**.

2.4 n Chaves de Cifra de Chaves de Sessão com Agente de Confiança

n Keys for Encryption of Session Keys with a Trusted Party

Uma forma de evitar o elevado número de chaves necessária num sistema consiste em **nomear um agente de confiança** (em quem todos confiam cegamente), e **trocar as chaves de sessão via esse agente**.



Desta forma, **só precisamos de** n chaves de cifra de chaves de sessão (**uma por cada entidade** ligada ao sistema). Repare-se que **a grande contrapartida deste esquema é que tem de existir um sistema que atue como agente de confiança**. Contudo, para um grande número de entidades em comunicação, a diferença de chaves em relação ao esquema anterior é significativa (e.g., 45 contra 10). De uma maneira geral, **a razão é de** $(n-1)/2$. O **estabelecimento de uma chave de sessão** pode ser conseguida da seguinte forma:

1. Alice (A): $sk \xleftarrow{r} \mathcal{K}$; $c_1 \leftarrow E(k_2, sk)$;
   A Alice gera uma chave de sessão aleatoriamente e cifra-a com a chave que partilha com o agente de confiança.
2. A $\rightarrow$ Agente de Confiança (AC): c_1;
   A Alice envia ao agente de confiança o criptograma resultante da cifra de sk com a chave pré-estabelecida k_2.
3. AC: $sk \leftarrow D(k_2, c_1)$; $c_2 \leftarrow E(k_5, sk)$;
   O agente de confiança recebe o criptograma, decifra-o para obter sk, que cifra novamente com k_5.
4. AC $\rightarrow$ Bob (B): c_2;
   O agente de confiança envia o criptgrama para o Bob.
5. B: $sk = D(k_5, c_2)$.
   Bob recebe e decifra o criptograma, obtendo a chave de sessão.

Nota: o conteúdo exposto na aula e aqui contido não é (nem deve ser considerado) suficiente para total entendimento do conteúdo programático desta unidade curricular e deve ser complementado com algum empenho e investigação pessoal.