Resumos para a parte 1 do exame

Cifras simétricas

São cifras que usam a **mesma chave** para cifrar e decifrar. Têm como vantagem **performance** e como desvantagem a **troca de chaves** (para ter segurança N pessoas precisam de `N * (N - 1) / 2` chaves).

Existem dois tipos de cifras simétricas:

- **Simétricas continuas**: a cifragem é feita byte a byte, ou seja, o texto cifrado é gerado à medida que o texto plano é lido. - Princípio fundamental é a maneabilidade (através da keystream)
- **Simétricas por blocos**: a cifragem é feita por blocos de tamanho fixo, ou seja, o texto cifrado é gerado quando o bloco é lido.

Cifra de DES

A cifra de DES é uma cifra simétrica por blocos. Usa na sua arquitetura **redes de feistel**. Tem outras variantes como o **3DES** que envolve 3x mais rondas na rede de feistel (DES tem 16 rondas, 3DES tem 48 rondas).

Cifra de AES

A cifra de AES é uma cifra simétrica por blocos. Usa na sua arquitetura **substituição-permutação**.

Funciona da seguinte forma:

1. Divide o texto-limpo em blocos de 128 bits (16 bytes);
2. É fornecida uma chave de cifragem;
3. É calculado o **número de rodadas** dependendo do tamanho da chave (128 bits = 10 rondas, 192 bits = 12 rondas, 256 bits = 14 rondas);
4. A chave é expandida em várias sub-chaves, uma para cada rodada; Através da "key expansion"
5. Após todas as rodadas é obtido o texto cifrado.

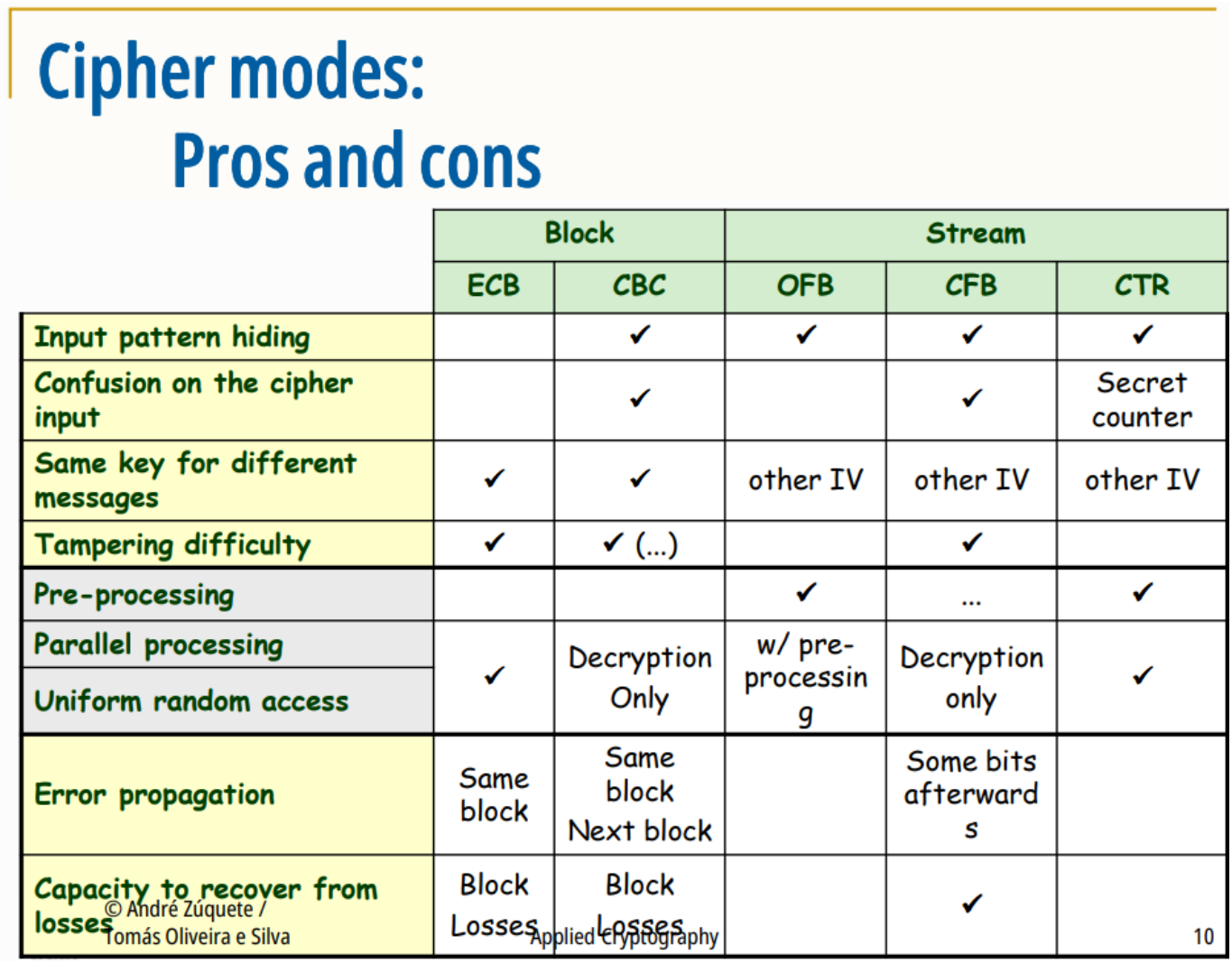
Em cada **rodada**:

1. **AddRoundKey**: o bloco é combinado com uma sub-chave exclusiva daquela rodada usando o XOR, o resultado é uma matriz 4x4 (16 bytes); - Confusão
2. **SubBytes**: cada byte no bloco é substituído por outro byte de acordo com uma tabela de substituição (S-box); - Confusão
3. **Shiftrows**: as linhas no bloco são permutadas (movidas) para a esquerda, essas permutação podem variar (0,1,2 e 3); - Difusão
4. **MixColumns**: as colunas no bloco são permutadas; - Difusão

Existem **5 modos de cifra**, onde:

- Três são continuas: CTR, CFB e OFB;
- Duas são por blocos: ECB e CBC;

Estas são as vantagens e desvantagens de cada modo:



Cipher modes: Pros and cons

	Block		Stream		
	ECB	CBC	OFB	CFB	CTR
Input pattern hiding		✓	✓	✓	✓
Confusion on the cipher input		✓		✓	Secret counter
Same key for different messages	✓	✓	other IV	other IV	other IV
Tampering difficulty	✓	✓ (...)		✓	
Pre-processing			✓	...	✓
Parallel processing	✓	Decryption Only	w/ pre-processing	Decryption only	✓
Uniform random access					
Error propagation	Same block	Same block Next block		Some bits afterwards	
Capacity to recover from losses	Block Losses	Block Losses		✓	

© André Zúquete / Tomás Oliveira e Silva Applied Cryptography 10

Eletronic Code Book (ECB)

O **ECB** é um modo de operação de cifra que consiste em cifrar cada bloco de dados de forma independente. Ou seja, o bloco de dados é cifrado com a mesma chave, mas o resultado é diferente (caso os blocos não sejam iguais) para cada bloco de dados.

Nota: **Não é recomendado** o uso para mensagens acima de 12 bytes não pseudo-aleatórios.

Exemplo para **cifrar**:

Texto	Bloco 1	Bloco 2	Bloco 3
Limpo	aaaaaaa	bbbbbbb	aaaaaaa
Algoritmo	AES-e(k,m[0])	AES-e(k,m[1])	AES-e(k,m[2])
Cifrado	934fafa	9f9f9f9	934fafa

Exemplo para **decifrar**:

Texto	Bloco 1	Bloco 2	Bloco 3
Cifrado	934fafa	9f9f9f9	934fafa

Texto	Bloco 1	Bloco 2	Bloco 3
Algoritmo	AES-d(k,c[0])	AES-d(k,c[1])	AES-d(k,c[2])
Limpo	aaaaaaa	bbbbbbb	aaaaaaa

Vantagens deste modo:

- Permite o acesso aleatório a dados cifrados.
- Permite processamento paralelo da informação.
- Não tem problemas de propagação de erros entre blocos.

Desvantagens deste modo:

- Não é seguro a ataques COA (*Cipher-Only attack*) e ataques por *code book* (compilação de pares texto limpo/criptograma).
- Não permite pré-processamento.
- O último bloco necessita sempre de preenchimento.
- Erros de sincronização são irrecuperáveis.

Cipher Block Chaining (CBC)

O **CBC** é um modo de operação de cifra que consiste em cifrar cada bloco de dados de forma dependente do bloco anterior. Ou seja, o bloco de dados é cifrado com a mesma chave, mas o resultado é diferente (mesmo os blocos sendo iguais) para cada bloco de dados. Este método evita **alguns** ataques por maninupulação de blocos.

Exemplo para **cifrar**:

- Recebe um bloco de texto-limpo;
- Recebe um vetor de inicialização (IV) - gerado aleatóriamente, único para cada mensagem e partilhado entre as duas partes;
- É feito o XOR entre o bloco de texto-limpo e o IV;
- O resultado do XOR é cifrado com a chave;
- O resultado cifrado é enviado para o bloco seguinte e irá ser feito o XOR com o bloco seguinte;
- Processo repete-se até ao fim da mensagem;

Exemplo para **decifrar**:

- Recebe um bloco de texto-cifrado;
- Este bloco é decifrado com a chave;
- É feito o XOR entre o bloco de decifrado e o IV;
- Esse bloco cifrado é enviado para o bloco seguinte e irá ser feito o XOR com o bloco seguinte após ser decifrado;
- Processo repete-se até ao fim da mensagem;

Vantagens deste modo:

- É seguro contra ataques CPA (*Chosen-Plaintext Attack*).
- Os padrões são mascarados pelo XOR e efeito cascata.
- Textos limpos iguais resultam em criptogramas distintos, inviabilizando ataques por *code book*.

- Permite o acesso aleatório a dados cifrados.
- Permite processamento paralelo da informação cifrada.

Desvantagens deste modo:

- Ataques por manipulação do IV podem não ser detetáveis.
- Não permite processamento paralelo da informação na cifragem.
- Erros de perda de bits são irrecuperáveis e um erro num bit afeta o bloco de texto limpo correspondente e o seguinte (cascata).

***Padding* (Preenchimento)**

O **padding** é uma técnica de preenchimento de dados que consiste em adicionar um número de bytes ao final de um bloco de dados, de forma a que o tamanho do bloco de dados seja múltiplo do tamanho do bloco de dados da cifra.

Nota: O último bloco tem sempre *padding*, mesmo que o bloco da mensagem seja múltiplo do tamanho do bloco da cifra é acrescentado um bloco de *padding*.

Output Feeback Mode (OFM)

O **OFM** é um modo de operação de cifra que consiste em cifrar o IV com a chave e fazer o XOR entre o resultado e o bloco de texto-limpo.

Exemplo a **cifrar**:

- Recebe um bloco de texto-limpo;
- Recebe um vetor de inicialização (IV) - gerado aleatóriamente, único para cada mensagem e partilhado entre as duas partes;
- É cifrado o IV **anterior** com a chave;
- É feito o XOR entre o bloco de texto-limpo e o resultado do IV cifrado;
- Processo repete-se, usando o IV do bloco anterior, até ao fim da mensagem;

Vantagens deste modo:

- Permite pré-processamento.
- Permite processamento paralelo da informação (caso, seja feito o pré-processamento).
- Permite acesso aleatório a dados cifrados (caso, seja feito o pré-processamento).

Desvantagens deste modo:

- Sendo ela uma cifra de chave simétrica contínua, ela fica **maneável**.
- Erros de perda bits são irrecuperáveis.

Ciphertext Feedback Mode (CFM)

O **CFM** é um modo de operação de cifra que consiste em cifrar o IV com a chave e fazer o XOR entre o resultado e o bloco de texto-limpo. A diferença entre o CFM e o OFM é que o CFM usa o bloco de texto-cifrado anterior e não o IV cifrado.

Exemplo a **cifrar**:

- Recebe um bloco de texto-limpo;
- Recebe um vetor de inicialização (IV);
- É cifrado o IV **anterior** com a chave (neste caso o IV é obtido através da operação xOr entre o IV cifrado e o bloco de texto-limpo);
- Processo repete-se, usando o IV do bloco anterior, até ao fim da mensagem;

Exemplo a **decifrar**:

- Recebe um bloco de texto-cifrado;
- Recebe um vetor de inicialização (IV);
- É decifrado usando o IV cifrado na primeira iteração;
- As restantes iterações usa como IV o bloco de texto-cifrado anterior e é feito o xOr com o bloco de texto-cifrado atual;
- Processo repete-se, até ao fim da mensagem;

Vantagens deste modo:

- Permite processamento em paralelo para a decifragem.
- Existe a capacidade de recuperação de erros.

Desvantagens deste modo:

- Sendo ela uma cifra de chave simétrica contínua, ela fica **maneável**.
- Um erro pode propagar-se para os bits seguintes.

Randomized Couter Mode (CTR)

O **CTR** é um modo de operação de cifra que consiste em cifrar através de uma chave de cifra de forma simétrica e continua a partir de uma cifra de blocos.

Exemplo para **cifrar**:

- Recebe um bloco de texto-limpo;
- Cria um contador (CTR) - do mesmo tamanho que o bloco, gerado aleatóriamente através da chave de cifra;
- Faz o XOR entre o bloco de texto-limpo e o contador;

Exemplo para **decifrar**:

- Recebe um bloco de texto-cifrado;
- Cria um contador (CTR) - do mesmo tamanho que o bloco, gerado aleatóriamente através da chave de cifra;
- Faz o XOR entre o bloco de texto-cifrado e o contador;

Vantagens deste modo:

- É seguro contra ataques CPA (*Chosen-Plaintext Attack*).
- Os padrões do texto-limpo são mascarados por imitar o processo de uma cifra contínua.
- Textos-limpos iguais resultam em criptogramas distintos, inviabilizando ataques por *code book*.
- Permite o acesso aleatório a dados cifrados.
- Permite o processamento paralelo da informação.

Desvantagens deste modo:

- Sendo ela uma cifra de chave simétrica contínua, ela fica **maneável**.
- Erros de perda bits são irrecuperáveis.

Funções de *hash*

Funções que dado um input de qualquer tamanho, produzem um output de um tamanho fixo.

Baseam-se em **três propriedades**:

- **Resistência à colisão**: é difícil encontrar dois inputs que produzam o mesmo output;
- **Resistência à pré-imagem**: é difícil encontrar um input que produza um output específico;
- **Resistência à segunda pré-imagem**: é difícil encontrar um input que produza o mesmo output que um input específico;

Funções de Hash Criptográficas (Abordagens)

- Construção Merkle-Damgard:
    1. Divide o input em blocos de tamanho fixo e aplica uma função de compressão a cada bloco, sendo que o output da função de compressão é usado como input para a próxima iteração.
    2. O output final é o output da última iteração.
    - Exemplo: MD5, SHA-1, SHA-2, SHA-3;
- Funções esponja (Sponge Functions):
    1. **Absorção**: Atualiza um estado interno (pool de entropia) com blocos de input de tamanho fixo (padding é adicionado se necessário).
    2. **Espremer (Squeezing)**: O output final é o output da última iteração.

Message Authentication Code (MAC)

Tem como propósito garantir que uma determinada mensagem não foi alterada e que foi enviada por uma determinada entidade (autenticidade).

Há várias formas de construir um MAC, algumas delas são:

- **HMAC** (Hash-based Message Authentication Code): este MAC tem dois algoritmos, um para construção e outro para verificação e recebe dois inputs, um é a chave e o outro é a mensagem;

As **três formas** possíveis de combinar os dois mecanismos são: Em termos de tempo de processamento o melhor é o terceiro e o pior é o segundo.

- *MAC and Encrypt*: Processo onde é calculado o MAC **através da mensagem** e depois é cifrada a mensagem concatenando o com o MAC (e.g. SSH);
- *MAC then Encrypt*: Processo onde é calculado o MAC **através da mensagem** e depois é cifrado MAC dessa mensagem (e.g. TLS);
- *Encrypt then MAC*: Processo onde é calculado o MAC **através da mensagem cifrada** (e.g. IPsec).

RSA

Um cifra de chave pública que precisa de:

- Gerador de chaves:

1. Primeiramente geramos dois números primos extremamente grandes, p e q.
2. Calcula-mos `N = p * q`.
3. Calcula-mos `phi(N) = (p - 1) * (q - 1)`.
4. Escolhe-se um expoente de encriptação `e` tal que `1 < e < phi(N)` e `mdc(e, phi(N)) = 1`.
5. Calcula-se o expoente de desencriptação `d` tal que seja o inverso multiplicativo de `e` módulo `phi(N)`, ou seja, `d * e = 1 mod phi(N)`.
6. A chave pública é `(N, e)` e a chave privada é `(N, d)`.

- Para cifrar fazemos a seguinte operação: `C = M^e mod N`, sendo o M a mensagem a cifrar e C a mensagem cifrada;
- Para decifrar fazemos a seguinte operação: `M = C^d mod N`, sendo o C a mensagem cifrada e M a mensagem decifrada.

A segurança desta cifra basea-se no problema do logaritmo discreto e no problema da fatorização de números primos.

Nota: Normalmente não se utiliza o **Text Book RSA** (apenas cifrar o conteúdo pretendido). Invés disso é acrescentado um padding aleatório para aumentar a segurança (OAEP ou Optimal Assimetric Encription Padding).

Aritmética Modular

A aritmética modular é essencial para manter os cálculos dentro de um intervalo específico, preservando a segurança e a integridade do esquema de partilha de segredo.

Teorema Chinês do Resto

O Teorema do Resto Chinês é um resultado matemático que facilita a **resolução de sistemas de congruências lineares**. Este teorema é utilizado para acelerar o **processo de decifragem do RSA**. Ele diz que se tivermos um sistema de congruências lineares, podemos resolver o sistema de forma mais rápida se resolvermos cada congruência individualmente.

Por exemplo, se o expoente usado para cifrar uma mensagem é sempre 3 e em que não se usa padding aleatório, enviar a mesma mensagem m para três destinatários diferentes, com chaves públicas (n1, 3), (n2, 3) e (n3, 3), é altamente desaconselhado, já que se as três mensagens cifradas, `m3 mod n1`, `m3 mod n2` e `m3 mod n3`, forem intercetadas é possível recuperar a mensagem original, da seguinte forma:

- Seja m a mensagem original;
- Seja n1, n2 e n3 os módulos das chaves públicas;
- Seja c1 = `m^3 mod n1`, c2 = `m^3 mod n2` e c3 = `m^3 mod n3`, isto é os residuos da divisão de m^3 por n1, n2 e n3, respectivamente;
- Seja N = `n1*n2*n3`, N é o produto dos módulos das chaves públicas;
- Seja N1 = N/n1, N2 = N/n2 e N3 = N/n3;
- Seja y1, y2 e y3 os inversos multiplicativos de N1, N2 e N3, respetivamente;
- Seja x = `c1*y1*N1 + c2*y2*N2 + c3*y3*N3`;
- A mensagem original é `m = x^(1/3) mod N`;

Criptografia de chave pública

A criptografia de chave pública moderna é baseada na teoria dos números e em problemas difíceis de resolver. Alguns exemplos são os protocolos de acordos de chaves, assinaturas digitais e cifras de chave pública.

Exemplos de problemas intratáveis com mais relevância:

- O problema do logaritmo discreto;
- O problema da fatorização de números compostos em números primos;

Problema do logaritmo discreto

Consiste em encontrar o valor de um número inteiro (exponencial) que é utilizado para cifrar uma mensagem, conhecido o valor da base e o resultado da exponenciação, num grupo finito. Esse problema é considerado difícil de resolver computacionalmente quando se trabalha com números grandes, tornando-o fundamental para garantir a segurança da criptografia de chave pública.

Nota: A definição mais geral do problema recorre apenas a grupos cíclicos, aplicando-se, por isso a outros grupos diferentes de números primos. O melhor algoritmo conhecido para resolver o problema do logaritmo discreto é conhecido por *general number field sieve*, determinando o tamanho que o número tem de ter para que o problema seja considerado seguro.

Problema da fatorização de números compostos

Consiste no facto de ser é computacionalmente inviável fatorizar um número composto pelo produto de dois números primos grandes. Por exemplo, se tivermos o número 15, ele pode ser decomposto em 3 e 5, que são números primos. O problema da fatorização torna-se difícil à medida que os números tornam-se maiores, e isso é utilizado em esquemas de criptografia de chave pública baseados em algoritmos como o RSA.

Assinatura Digital

Os esquemas de assinatura digital atuais são normalmente construídos usando criptografia de chave pública. Estas são algumas das propriedades:

- Autenticidade da Informação
- Integridade dos Dados
- Garantia de Não Repúdio
- Autenticação da Origem da Informação
- Dificuldade de Falsificação

Repare-se que, na verdade, enquanto que a assinatura digital pode garantir todas as propriedades anteriores, o mesmo não costuma acontecer para as assinaturas manuscritas.

Protocolos de acordos de chaves

Neste capítulo é apresentado formas de trocar ou estabelecer um segredo entre duas entidades sem haver nada secreto acordado à partida. Essa troca é realizada por uma comunicação segura, aonde as partes enviam mensagens encriptadas uma para a outra e usam técnicas de criptografia e matemática para garantir que apenas elas possam obter a chave secreta compartilhada. O objetivo é garantir a segurança e a privacidade das comunicações entre as partes envolvidas.

Nota: Este tipo de protocolo deve ser utilizado em situações de ataque ao homem no meio passivo, onde o atacante não consegue modificar as mensagens que são enviadas entre as partes.

Protocolo de Diffie-Hellman

Este é um exemplo de um protocolo de acordo de chaves, onde duas partes podem concordar sobre uma chave secreta, mesmo que nunca tenham trocado qualquer informação secreta anteriormente. A sua segurança basea-se no problema do logaritmo discreto.

Funciona da seguinte forma:

G^k = 1 mod P, onde k > 0

1. Alice e Bob escolhem um número primo P e um número G que seja raiz primitiva de P;
2. Alice escolhe um número secreto A e Bob escolhe um número secreto B;
3. Alice calcula G^A mod P e envia para Bob;
4. Bob calcula G^B mod P e envia para Alice;
5. Alice calcula (G^B mod P)^A mod P;
6. Bob calcula (G^A mod P)^B mod P;
7. Ambos obtêm o mesmo resultado, que é a chave secreta partilhada.

Para estabelecer um segredo entre 3 partes (ao invés de 2), é feito do seguinte modo:

1. Alice, Bob e Carol escolhem um número primo P e um número G que seja raiz primitiva de P;
2. Alice escolhe um número secreto A, Bob escolhe um número secreto B e Carol escolhe um número secreto C;
3. Alice calcula G^A mod P (y_a) e envia para Bob;
4. Bob calcula G^B mod P (y_b) e envia para Carol;
5. Carol calcula G^C mod P (y_c) e envia para Alice;
6. Bob calcula (y_c)^B mod P (K_ab);
7. Carol calcula (y_a)^C mod P (K_bc);
8. Alice calcula (y_b)^A mod P (K_ca);
9. Cada participante combina as chaves partilhadas para obter a chave secreta partilhada (K = K_ab + K_bc + K_ca).

Curvas Elípticas

A segurança deste método basea-se no problema do logaritmo discreto.

2

Numa curva eliptica, a adicao de 2 pontos dá origem a um terceiro ponto nessa mesma curva eliptica, sendo P1+P2=P3.

O que acontece é que é tracada uma reta que passa por P1 e P2 que interceta P3 na curva noutro local. Se os dois pontos P1 e P2 coincidirem, isto e, forem o mesmo, a reta vai ser tangente à curva eliptica e irá ecntrar o segundo ponto na curva eliptica, fazendo P1+P2=P3 <=> P1+P1=P3 <=> 2*P1=P3. Desta forma obtem-se uma multiplicacao por 2.

Para explorar esta propriedade são feitas várias somas consecutivas da seguinte forma:

```
11P = 8P + 2P + P
8P = 4P + 4P
```

```
4P = 2P + 2P
2P = P + P
```

Para multiplicar um ponto de um curva eliptica, usando um número inteiro negativos adiciona-se um passo, que é a multiplicação do ponto por -1, e depois faz-se a soma dos pontos. Por exemplo:

```
11P = -8P - 2P - P
-8P = -4P - 4P
-4P = -2P - 2P
-2P = (-P) + -(P) = -P - P
```

O protocolo Diffie-Helman pode ser implementado usando curvas elípticas, pois a operação de multiplicação de um ponto por um escalar é computacionalmente difícil de ser invertida;

Este protocolo é implementado por curvas elípticas da seguinte forma:

- Alice e Bob concordam em usar uma curva elíptica sobre um corpo finito e um ponto gerador G;
- Alice escolhe um ponto `y_a` e Bob escolhe um ponto `y_b`;
- Alice calcula `k_a = y_a * G` e Bob calcula `k_b = y_b * G`;
- A chave será o ponto `k_a * y_b = k_b * y_a`;