Pós-Graduação Lato Sensu
Curso de Especialização em Redes De Computadores

Serviços de Segurança em Redes

Prof. Dr. **Lucas Dias Hiera Sampaio**

PARTE 03

As partes não podem ser compiladas integralmente.
Para uso exclusivo do curso de Pós Graduação da Universidade.

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

Sistemas Criptográficos

Introdução

Na última semana discutimos as três categorias diferentes de sistemas criptográficos: criptografia simétrica, criptografia assimétrica e funções de hash. Esta semana vamos conhecer os diferentes algoritmos existentes, modos de operação e a importância dos parâmetros utilizados.

1. Sistemas de criptografia simétrica

Como visto anteriormente, os sistemas de criptografia simétrica são classificados em cifras de bloco e cifras de fluxo: nas cifras em bloco a informação é dividida em partes que são criptografadas de forma conjunta; já na criptografia em fluxo as cifras são criadas bit a bit operando a informação diretamente com um fluxo de chave (*key stream*).

Adicionalmente, existem diferentes modos de operação dos algoritmos de cifra em bloco que serão abordados nesta seção.

1.1. Modos de Operação de Cifra em Bloco

Existem diferentes formas de se executar a criptografia em uma sequência de informações utilizando blocos. A forma mais simples de realizar a criptografia é simplesmente utilizando a chave simétrica e realizando a criptografia de forma independente entre os blocos conforme mostram as Figuras 1 e 2. Esse modo de operação é denominado *Electronic Codebook* ou ECB.

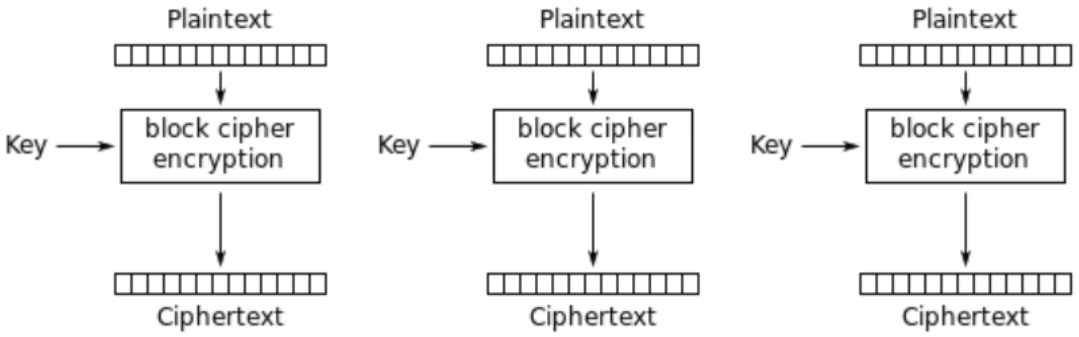

Figura 1: Criptografia Usando ECB

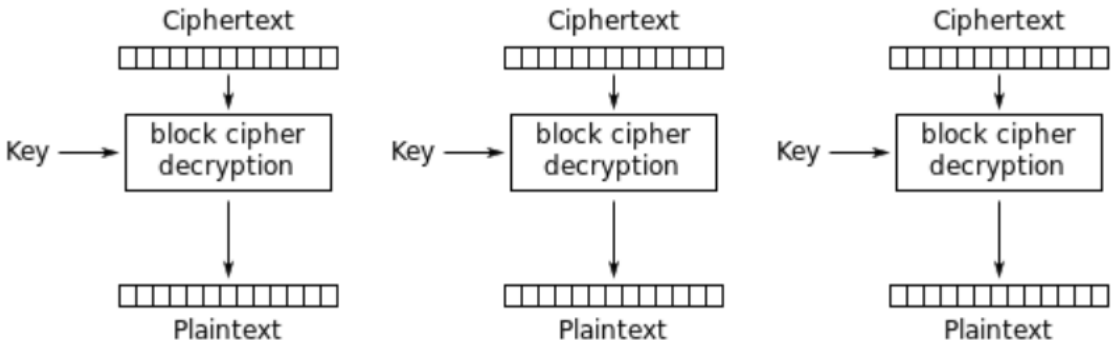

Figura 2: Descriptografia utilizando modo ECB.

Este método é raramente utilizado e deve ser evitado sempre que possível pois ele não provê difusão das informações, isto é, é fácil perceber padrões na informação criptografada pois há uma correlação direta entre texto plano e texto cifrado. A Figura 3 mostra um exemplo de imagem criptografada usando ECB.

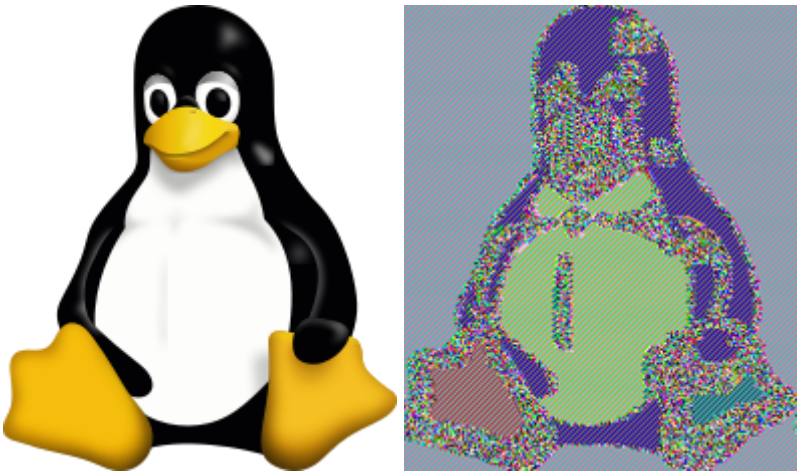

Figura 3: Na esquerda a imagem original. Na direita a imagem criptografada utilizando ECB.

Em 1976 Ehrsam, Meyer, Smith e Tuchman desenvolveram o modo de operação para algoritmos de criptografia simétrica em blocos denominado *cipher block chaining* ou CBC. Neste modo de operação o texto cifrado de cada bloco passa por uma operação de ou-exclusivo (XOR) bit a bit com o texto plano do bloco seguinte. Como não há blocos antes do primeiro bloco, um vetor de inicialização (VI) é utilizado para realizar a operação de ou-exclusivo com o texto plano do primeiro bloco da mensagem. As Figuras 4 e 5 apresentam o funcionamento do modo CBC.

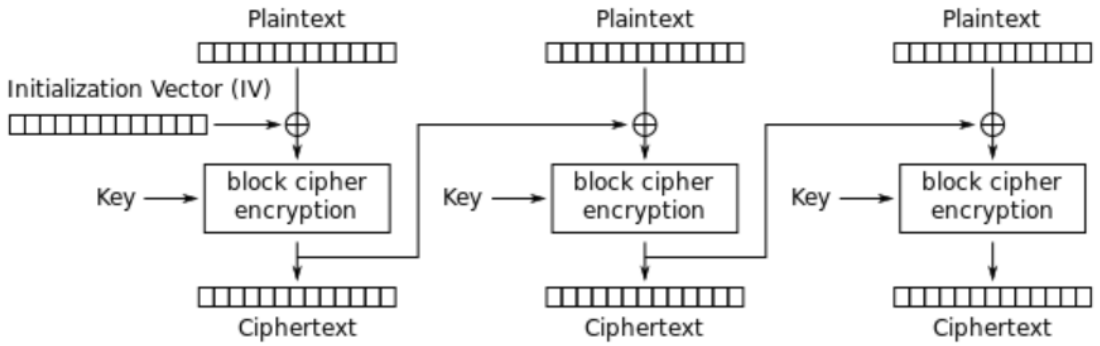

Figura 4: Criptografia utilizando o modo CBC.

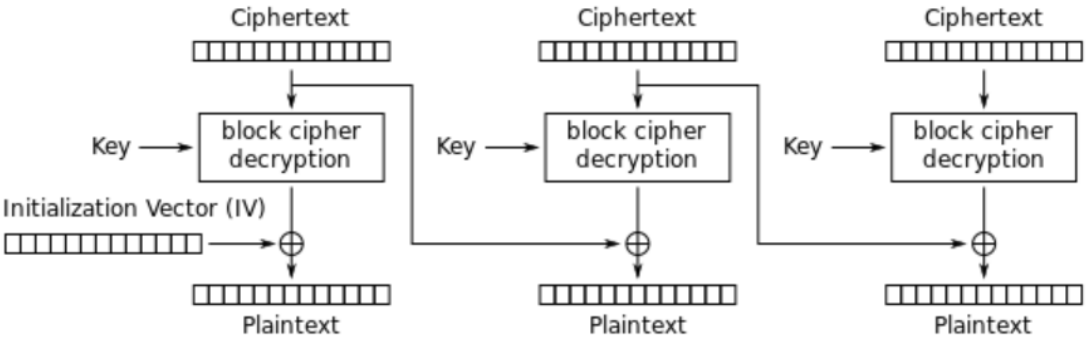

Figura 5: Descriptografia utilizando o modo CBC.

É importante observar que no CBC consegue descriptografar toda a mensagem com exceção do primeiro bloco caso o vetor de inicialização seja comprometido. Neste sentido, para impedir que isso fosse possível, foi desenvolvido o modo *propagating cipher block chaining* ou PCBC. Neste modo, o texto pleno de cada bloco passa por uma operação de ou-exclusivo junto ao texto pleno e o texto cifrado do bloco anterior, logo, caso o vetor de inicialização seja comprometido, toda a cadeia de blocos será descriptografada incorretamente. As Figuras 6 e 7 demonstram o funcionamento do PCBC.

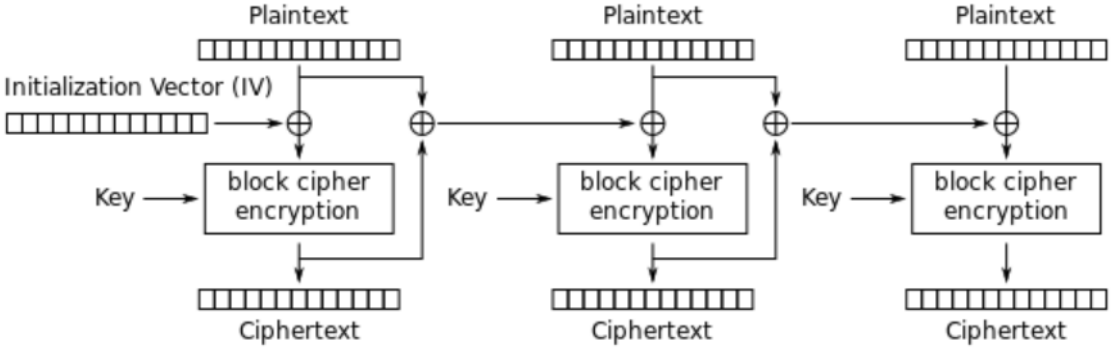

Figura 6: Criptografia utilizando o modo de operação PCBC.

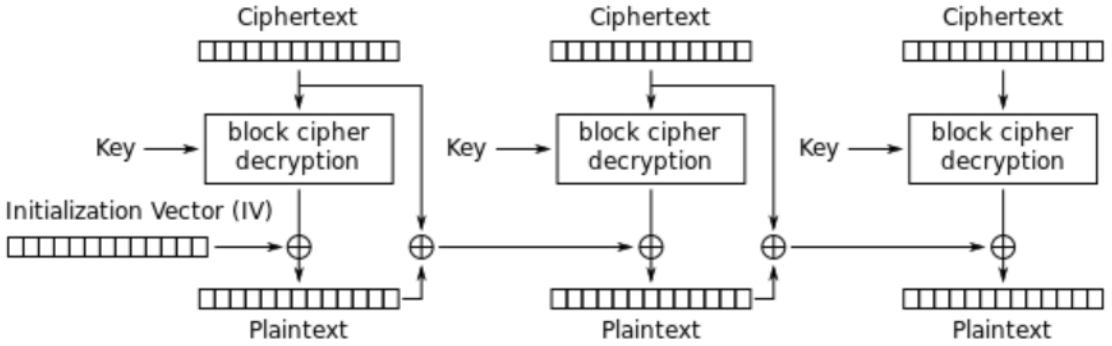

Figura 7: Descriptografia utilizando o modo de operação PCBC.

Estes três modos de operação discutidos até este momento são originalmente o conjunto de modos de operação que de fato fazem a criptografia da informação em blocos. Os métodos discutidos a partir desse ponto transformam algoritmos de criptografia em blocos em sistemas de criptografia em fluxo.

O primeiro modo de operação que transforma o algoritmo de criptografia em blocos em um algoritmo de criptografia em fluxo é o *Cipher Feedback* ou CFB. Ele funciona criptografando o texto cifrado do bloco anterior utilizando o algoritmo de criptografia em blocos e a chave de criptografia e posteriormente realizando a operação de ou-exclusivo com o texto pleno (como acontece nos algoritmos de criptografia em fluxo). O CFB também faz uso de um VI no primeiro bloco conforme mostram as Figuras 8 e 9.

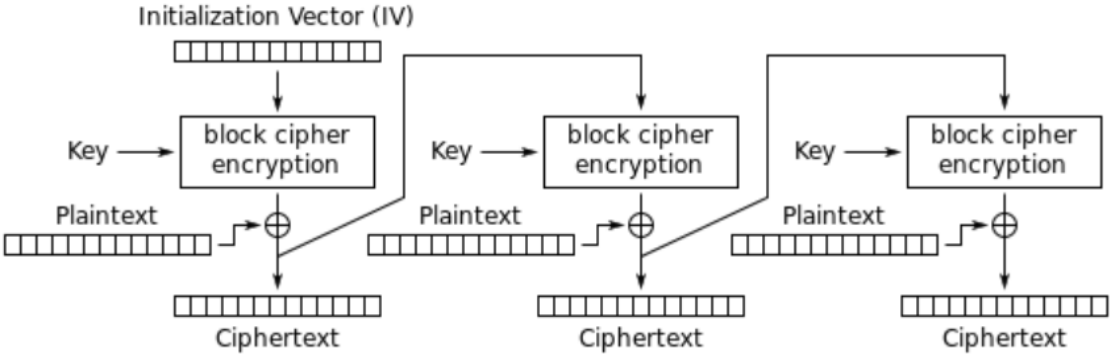

Figura 8: Criptografia utilizando o modo CFB.

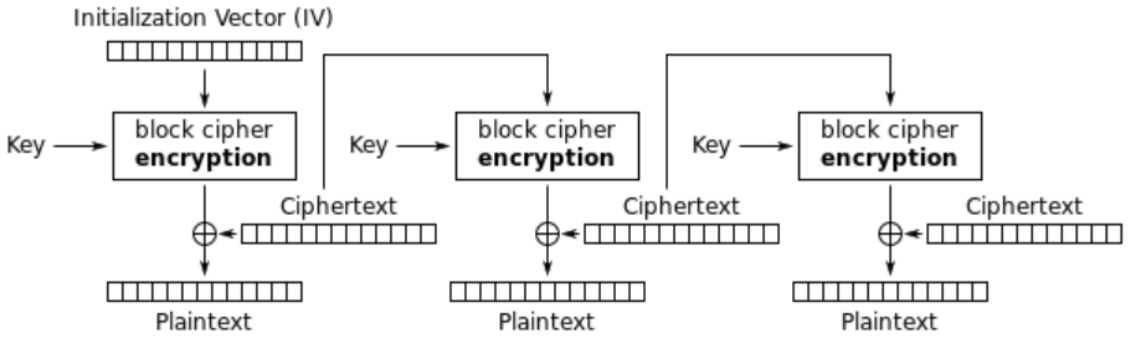

Figura 9: Descriptografia utilizando o modo CFB.

Outro modo de operação dos algoritmos simétricos é o *Output Feedback* ou OFB. Neste modo o algoritmo de criptografia em bloco também é convertido em um algoritmo de criptografia em fluxo e funciona basicamente como o CFB sendo a única diferença que a entrada do bloco é somente o resultado da criptografia do bloco anterior sem ter sido operada (XOR) com o texto pleno. As Figuras 10 e 11 apresentam o processo de criptografia e descriptografia, respectivamente, utilizando o modo OFB.

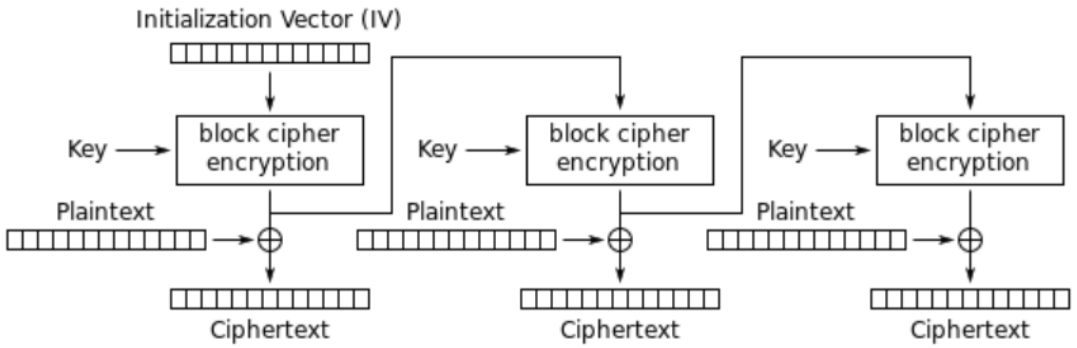

Figura 10: Criptografia utilizando o modo OFB.

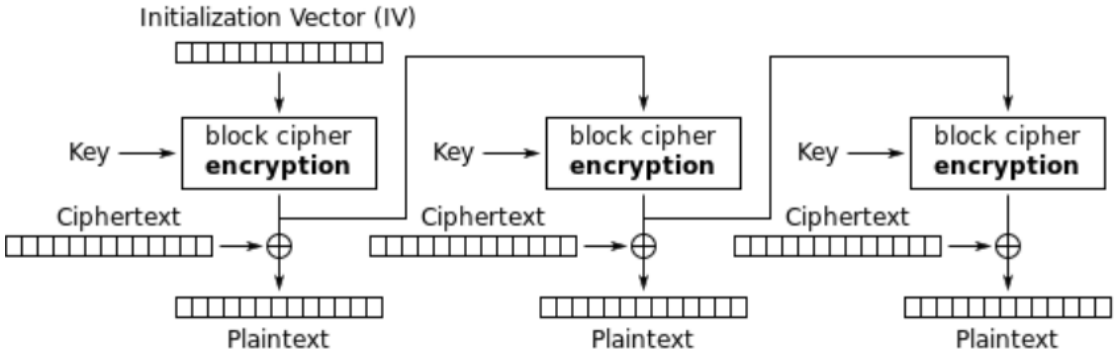

Figura 11: Descriptografia utilizando o modo OFB.

Outro modo de operação fundamental na atualidade é o *Counter* (CTR) criado por Whitfield Diffie e Martin Hellman em 1979. A principal característica deste modo de operação quando comparado com o OFB e o CFB é o fato de o processo de criptografia e descriptografia ser totalmente paralelizável. Neste modo um número aleatório denominado Nonce (equivalente ao VI) é combinado com um contador, por meio de concatenação, adição ou operação ou-exclusivo, e então criptografado utilizando o algoritmo simétrico e sua chave. O resultado deste processo é então combinado ao texto plano utilizando a operação de ou-exclusivo. A cada novo bloco, o contador aumenta seu valor e a criptografia de todos os trechos de informação pode acontecer simultaneamente em um processo paralelo. As Figuras 12 e 13 apresentam o processo de criptografia e descriptografia do modo CTR.

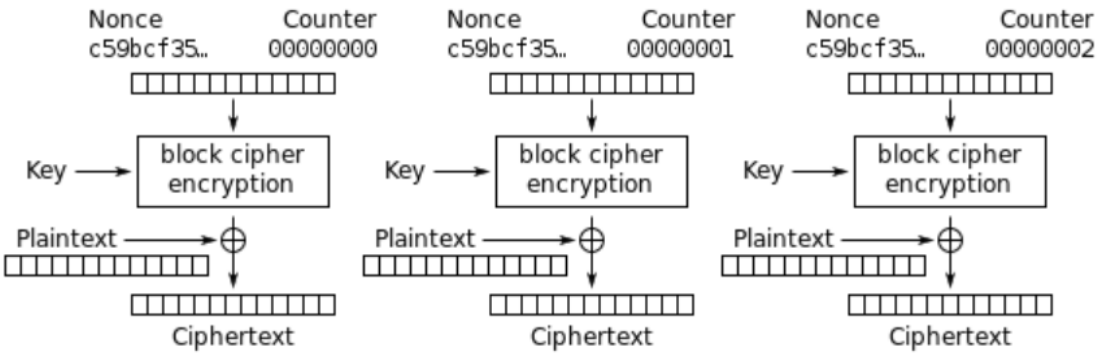

Figura 12: Processo de criptografia utilizando o modo CTR.

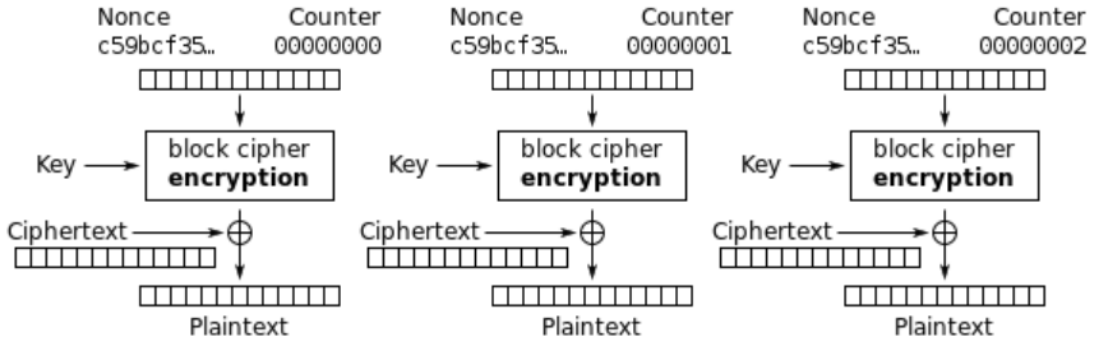

Figura 13: Processo de descriptografia utilizando o modo CTR.

1.2. Advanced Encryption Standard (AES)

O algoritmo de criptografia simétrica em blocos mais utilizado atualmente é o AES que foi desenvolvido na década de 1990 pelos belgas Vincent Rijmen e Joan Daemen e cujo nome original é Rijndael. O algoritmos foi escolhido pelo Instituto Nacional de Padrões e Tecnologias (NIST) dos Estados Unidos como padrão de criptografia simétrica substituindo o algoritmo DES (*Data Encryption Standard*).

De forma geral, o AES é um algoritmo que funciona com blocos de 128 bits e pode utilizar chaves de 128, 192 ou 256 bits de tamanho. Sua estrutura é uma rede de substituição e permutação e 4 operações compõem a sua lógica: substituição de bytes, deslocamento de linhas, mistura de colunas e adição de chave (em inglês: *byte sub, shift row, mix column* e *add round key*).

O funcionamento do AES inicia pela definição da matriz de estado do algoritmo: os 128 bits de entrada são convertidos em uma matriz 4 por 4 onde cada

elemento é 1 byte (ou 8 bits) da entrada. Sobre essa matriz de estado serão realizadas as 4 operações listadas acima em um processo que se repete múltiplas vezes dependendo do tamanho da chave: quando a chave é de 128 bits, o processo se repete 10 vezes, quando é de 192 bits repete 12 vezes e quando é de 256 bits, 14 vezes.

Na etapa de *bytes sub* (substituição de bytes), cada elemento da matriz é dividido em 2 números de 4 bits, os 4 primeiros bits identificam a linha de uma matriz de substituição e os 4 últimos a coluna. Essa matriz de substituição é construída seguindo critérios matemáticos que permitem a sua inversão quando realizamos a descriptografia e ainda impede que bytes não sejam alterados, isto é, os 8 bits utilizados para designar a linha e coluna da matriz de substituição são alterados de forma que não há repetição de **bytes**.

O passo seguinte é o deslocamento das linhas. Na primeira linha da matriz de estado não há alterações. Na segunda linha há um deslocamento de 1 byte para esquerda, na terceira linha 2 bytes e na quarta linha 3 bytes. Essa etapa evita que as colunas da matriz de estado sejam criptografadas de forma independente.

O terceiro passo é a mistura de colunas (*mix column*): nesta etapa, cada coluna da matriz de estado é multiplicada (à esquerda) por uma matriz de transformação linear invertível.

Por fim, a quarta operação é a adição da chave da rodada (*add round key*): nesta etapa os elementos da matriz de estado são combinados com os elementos da chave da rodada utilizando a operação ou-exclusívo.

Como observado na quarta operação, a cada repetição do algoritmo (rodada) uma chave diferente é utilizada, todavia, apenas uma chave consta na entrada do algoritmo AES. Portanto, o algoritmo precisa derivar chaves diferentes para cada uma das repetições. Esse processo é denominado *key schedule.*

O processo de derivação de chave usa 3 recursos fundamentais: a rotação de palavra (*RotWord*) no qual o elemento da primeira linha em uma coluna de uma matriz é posicionado na última linha e todos os demais elementos são deslocados; a operação de substituição de palavra (*SubWord*) que funciona conforme a operação *bytes sub*; por fim, a combinação da primeira coluna de cada nova chave com uma coluna do vetor de constante da rodada (*round constant* ou *rcon*) por meio da operação de ou exclusivo. A Figura 14 apresenta parte do processo de criação das chaves de cada rodada.

A fim de elucidar a compreensão desse algoritmo recomenda-se a visualização do vídeo[1].

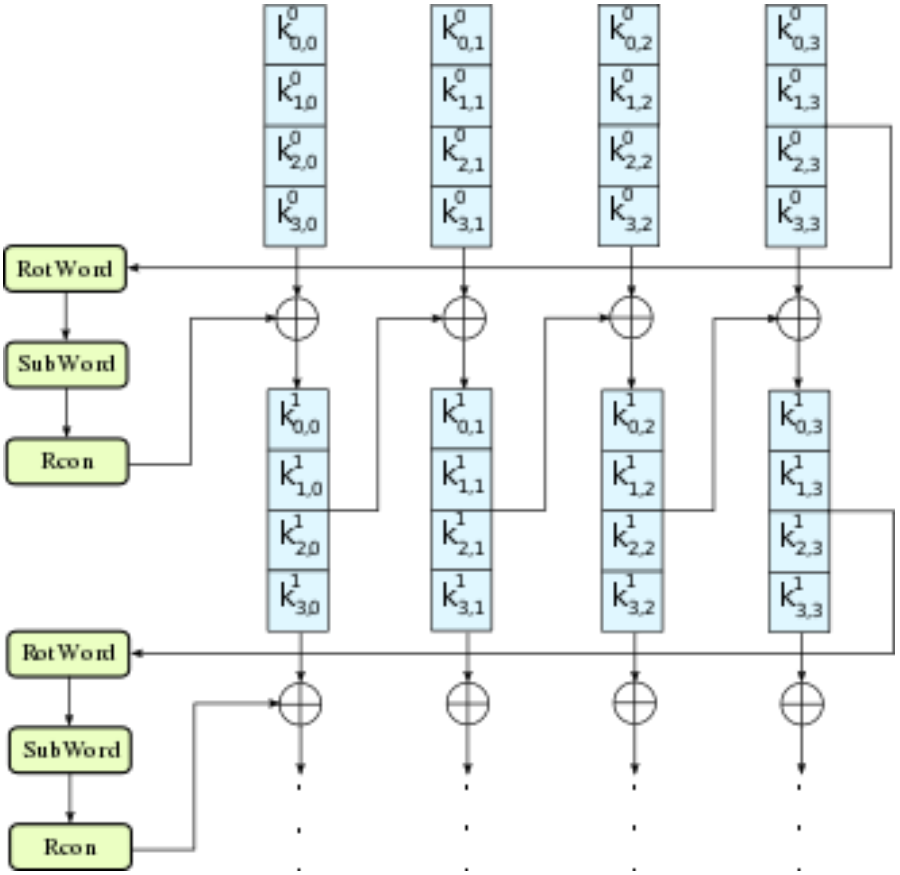

Figura 14: Processo de derivação das chaves de cada rodada no algoritmo AES.

Neste material não cobrimos as razões e fundamentos que explicam o funcionamento do algoritmo uma vez que não faz parte do programa da disciplina e exige o conhecimento profundo de conceitos matemáticos. Todavia, ficará disponível no Moodle links para aulas que explicam matematicamente o funcionamento do algoritmo AES.

1.3. ChaCha20

O algoritmo de criptografia simétrica ChaCha foi desenvolvido em 2008 por Daniel J. Bernstein e é uma derivação do algoritmo Salsa20 desenvolvido pelo mesmo autor. Trata-se de um algoritmo de criptografia em fluxo que combinado com o algoritmo de autenticação Poly1305 passou a ser utilizado no protocolo

[1] AES Rijndael Cipher explained as a Flash animation

QUIC e no HTTP/3. Adicionalmente, ChaCha20 pode ser utilizado junto ao protocolo IKE (*Internet Key Exchange),* a suíte de protocolos IPsec, no TLS (*Transport Layer Security*) bem como no protocolo WireGuard VPN.

O algoritmo pode funcionar com chaves de 128 ou 256 bits e utiliza o mesmo conceito de estado que tem-se no AES, com a diferença da matriz 4 por 4 ser composta por palavras de 32 bits, logo o estado tem 512 bits. Os 512 bits do estado na saída do algoritmo são usados para criptografar o fluxo utilizando a operação de ou-exclusivo. A Tabela 1 apresenta como é organizado o estado inicial do ChaCha20.

Tabela 1: Organização inicial da matriz 4 por 4 de estado do algoritmo ChaCha20. Cada elemento da matriz tem 32 bits.

“expa”	**“nd 3”**	**“2-by”**	**“te k”**
chave	**chave**	**chave**	**chave**
chave	**chave**	**chave**	**chave**
contador	**contador**[2]	**nonce**	**nonce**

Para gerar o fluxo de chave é necessário alterar a entrada do algoritmo, daí a existência do Nonce e do contador dentro da matriz de estados. O algoritmo segue uma estrutura ARX (*addition, rotation, xor*), ou seja, utiliza somente as operações de adição, rotação (deslocamento de bits) e ou-exclusivo. Além disso durante as 20 repetições do algoritmo ele alterna as entradas de sua lógica: nas rodadas ímpares as entradas são as colunas da matriz de estado (coluna 1 = a, coluna 2 = b, coluna 3 = c e coluna 4 = d). Já nas rodadas pares as entradas são as linhas (linha 1 = a, linha 2 = b, linha 3 = c, linha 4 = d). O estado do algoritmo após 20 rodadas é utilizado para criptografar 512 bits de informação utilizando a operação ou-exclusivo. A Figura 15 apresenta o funcionamento de cada rodada do ChaCha20.

[2] Na implementação do IETF (RFC 7539) o nonce tem 96 bits e ocupa este espaço também.

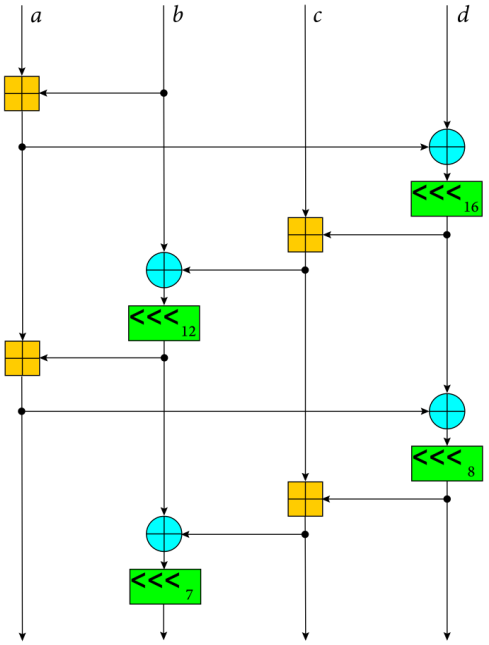

Figura 15: Uma repetição dentro do ChaCha20. Em amarelo as adições, azul as operações de ou exclusivo e em verde as rotações.

2. Sistemas de criptografia assimétrica

Na última aula estudamos o algoritmo de criptografia assimétrica Diffie-Hellman. Nesta semana vamos conhecer dois importantes algoritmos de criptografia assimétrica: o RSA e a Criptografia de Curvas Elípticas.

2.1. RSA

O RSA é um algoritmo de criptografia assimétrica e é um acrônimo para os sobrenomes de seus inventores: Ron Rivest, Adi Shamir e Leonard Adleman que desenvolveram o algoritmo em 1977. Como discutido anteriormente, os algoritmos de criptografia assimétrica tem sua segurança garantida por um problema difícil de ser resolvido, o que conhecemos por *trapdoor function*. No caso do RSA, esse problema difícil de resolver é a fatoração em primos.

A fatoração em primos é o processo de encontrar o conjunto de todos os números primos divisores de um número. Por exemplo, se o número que desejamos fatorar for 21, seus fatores são 3 e 7 - números primos. Embora pareça fácil, esse problema passa a ter um tempo de processamento muito grande quando os números passam a ser maiores e alguns casos poderiam levar milênios mesmo com o uso de supercomputadores e o uso de algoritmos de fatoração como o Rô de Pollard.

O processo de criptografia utilizando RSA é, na realidade, parecido com o processo que vimos no Diffie-Hellman. Considerando três inteiros muito grandes e, d e n, uma mensagem m pode ser criptografada e descriptografada como:

$$c = m^{e} (mod\, n)$$
$$m = c^{d}\ (mod\, n)$$

No caso acima, os números (e, n) formam a chave pública - e é o expoente público, e os números (d, n) a chave privada - d é o expoente privado. O processo de derivação dessas chaves segue o seguinte formato:

- **Passo 1:** escolha dois números primos grandes **p** e **q**. Preferencialmente os números devem ser escolhidos aleatoriamente e possuir uma grande diferença entre eles.
- **Passo 2:** Calcule $n = p\,q$. O comprimento de n é o tamanho da chave.
- **Passo 3**: Calcule $\lambda(n) = mmc(p-1, q-1)$ onde $\lambda()$ é a função totiente de Carmichael.
- **Passo 4:** Escolha e tal que $2 < e < \lambda(n)$ e $mmc(e, \lambda(n)) = 1$, isto é, e e n são coprimos. Geralmente o expoente público escolhido é o valor $2^{16} + 1 = 65537$.
- **Passo 5:** Determine d tal que $d \equiv e^{-1} (mod\, n)$, isto é d é o inverso multiplicativo de e módulo n, ou seja, $d\,e\ (mod\, n) = 1$.

Para compreender o funcionamento do RSA segue um exemplo numérico:

- **Passo 1:** Escolhemos $p = 23$ e $q = 47$.
- **Passo 2:** $n = 23 \times 47 = 1081$
- **Passo 3:** $\lambda(1081) = mmc(22, 46) = 506$
- **Passo 4:** $e = 17$
- **Passo 5:** $d = 387$, pois $387 \times 17 = 6579\, mod\, 506 = 1$
- **Criptografia:** supondo que $m = 65$ temos $c = 65^{17} mod\, 1081 = 849$
- **Descriptografia:** $m = 849^{387} mod\, 1081 = 65$

2.2. Criptografia por Curvas Elípticas

A criptografia por curvas elípticas (ECC) foi proposta originalmente em 1985 por Neal Koblitz e Victor S. Miller porém seu uso só foi popularizado após 2004. Uma curva elíptica é matematicamente definida como uma curva plana em um corpo finito - ou corpo de Galois e cujos pontos satisfazem a seguinte equação:

$$y^2 = x^3 + ax + b,$$

Uma das propriedades de curvas elípticas é que ao traçar uma reta em qualquer direção ela tocará a curva em até 3 pontos. A Figura 16 apresenta um exemplo de curva elíptica e uma reta tocando a curva em 3 pontos.

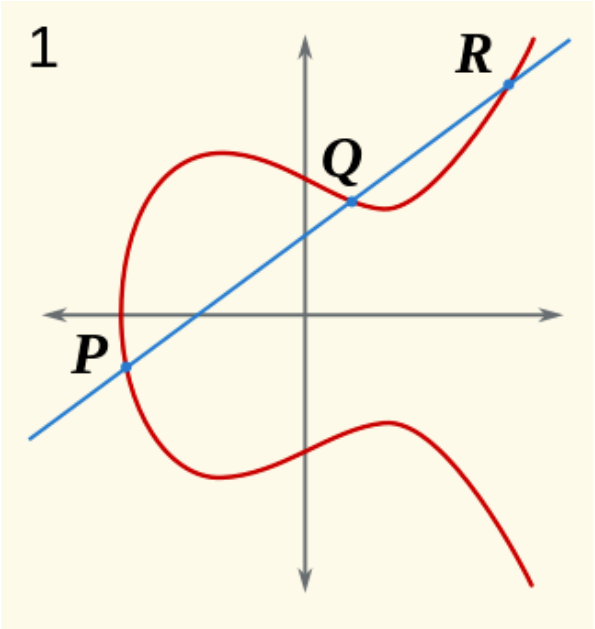

Figura 16: Curva elíptica (em vermelho) e reta (em azul) com três pontos (P,Q e R) tocando a curva.

Uma das principais vantagens da criptografia por curvas elípticas é a redução no tamanho da chave comparado com o RSA: para garantir a mesma segurança que uma chave RSA de 3072 bits podemos utilizar uma chave de 256 bits em criptografia por curvas elípticas.

Uma explicação mais detalhada do funcionamento do ECC está disponível na videoaula desta semana.

3. Limitações, Possibilidades e Problemas

Quando se trata de criptografia simétrica e assimétrica é fundamental compreender as limitações de cada uma: a criptografia simétrica necessita de um algoritmo assimétrico para compartilhar a sua chave, por outro lado, a criptografia assimétrica é mais complexa computacionalmente. Logo, utilizar criptografia assimétrica em todas as comunicações não é interessante pois pode introduzir atraso e comprometer a disponibilidade do sistema.

Um ponto fundamental a ressaltar é que em alguns casos a criptografia simétrica pode ser utilizada para prover autenticidade como no caso do algoritmo AES operando em modo GCM (*Galois Counter Mode*) que nada mais é que o modo CTR com um processo de autenticação alinhado.

Na próxima aula iremos abordar entre diferentes temas a criptoanálise: a criação de ferramentas capazes de comprometer a segurança dos sistemas criptográficos que estudamos. Adicionalmente, vamos estudar os *side channel attacks*, ou ataques de canal lateral, que buscam acessar informações no computador ou na implementação do algoritmo para comprometer sua segurança.