1.Introdução à criptografia simétrica

1.1 Componentes e funcionamento da criptografia simétrica

A criptografia simétrica envolve três componentes principais:

- O texto claro (***plaintext***).
- A chave simétrica (***symmetric key***).
- O texto cifrado (***ciphertext***).

1.1.1 Texto claro (plaintext)

O texto claro refere-se aos dados originais que se deseja proteger. Pode ser qualquer forma de informação, como mensagens de texto, arquivos, documentos, imagens, entre outros. Antes de serem criptografados, esses dados estão em formato legível e compreensível.

1.1.2 Chave simétrica

A ***chave simétrica*** é um valor secreto, compartilhado entre o remetente e o destinatário da mensagem. É com base nessa chave que ocorre a transformação dos dados originais em formato ilegível. A chave simétrica é utilizada tanto para ***criptografar*** (***processo de transformação do texto claro em texto cifrado***) quanto para ***descriptografar*** (***processo de reconverter o texto cifrado em texto claro***) os dados. É crucial que a chave simétrica seja mantida em segredo e seja conhecida apenas pelas partes autorizadas.

1.1.3 Texto cifrado

O texto cifrado é o resultado da aplicação do algoritmo de criptografia a chave simétrica e ao texto claro. Após a criptografia, os dados originais são transformados em uma forma ilegível e aparentemente aleatória. O texto cifrado é o que é transmitido ou armazenado de forma segura, uma vez que não pode ser compreendido por terceiros que não possuam a chave correta para desfazer a criptografia.

A criptografia simétrica utiliza o texto claro, a chave simétrica e o texto cifrado para proteger a confidencialidade dos dados. O remetente utiliza a chave simétrica para criptografar o texto claro, gerando o texto cifrado, que pode ser transmitido ou armazenado com segurança. O destinatário, por sua vez, utiliza a mesma chave simétrica para descriptografar o texto cifrado e recuperar o texto claro original. Ela pode utilizar ***cifras de fluxo*** ou ***cifras de bloco***.

Os algoritmos simétricos usam a mesma chave pré-compartilhada para criptografar e descriptografar dados. Uma chave pré-compartilhada, também chamada de chave secreta, é conhecida pelo remetente e pelo receptor antes que qualquer comunicação criptografada possa ocorrer.

Hoje, algoritmos de criptografia simétrica são comumente usados com o tráfego VPN. Isso ocorre porque os algoritmos simétricos usam menos recursos da CPU do que os algoritmos de

criptografia assimétrica. Isso permite que a criptografia e a descriptografia de dados sejam rápidas ao usar uma VPN. Ao usar algoritmos de criptografia simétrica, como qualquer outro tipo de criptografia, quanto maior a chave, mais tempo levará para alguém descobrir a chave. A maioria das chaves de criptografia tem entre 112 e 256 bits.

1.2 Cifra de fluxo

Uma cifra de fluxo é um tipo de algoritmo de criptografia simétrica que opera em tempo real, transformando dados em um fluxo contínuo de caracteres cifrados. Ao contrário das cifras de bloco, que operam em blocos fixos de dados, as cifras de fluxo criptografam os dados em um fluxo contínuo de bits ou bytes.

Nas cifras de fluxo, a criptografia é realizada combinando os bits do texto claro com uma sequência de bits gerada pela chave simétrica. Essa sequência de bits é chamada de "fluxo de chave" (keystream) e é gerada pelo algoritmo de cifra de fluxo a partir da chave simétrica.

O fluxo de chave é combinado com os bits do texto claro por meio de uma operação lógica, como a operação XOR (OU exclusivo). Cada bit do texto claro é combinado com o bit correspondente do fluxo de chave, gerando o bit cifrado correspondente. Esse processo é repetido para cada bit ou byte dos dados a serem criptografados.

A principal vantagem das cifras de fluxo é que elas são geralmente rápidas e eficientes em termos de recursos computacionais, o que as torna adequadas para aplicação em comunicações em tempo real, como transmissões de dados contínuas.

No entanto, é importante garantir que o fluxo de chave seja gerado de forma segura e aleatória, pois qualquer falha na geração ou no uso do fluxo de chave pode comprometer a segurança dos dados criptografados.

1.3 Cifra de blocos

Uma cifra de blocos é um tipo de algoritmo de criptografia simétrica que opera em blocos fixos de dados. Ao contrário das cifras de fluxo, que criptografam os dados em um fluxo contínuo, as cifras de blocos dividem o texto claro em blocos de tamanho fixo e criptografam cada bloco independentemente.

Em uma cifra de blocos, cada bloco de dados é processado individualmente usando a chave simétrica e o algoritmo de criptografia. O tamanho do bloco pode variar dependendo do algoritmo específico, mas geralmente é de 64 bits (8 bytes) ou 128 bits (16 bytes).

Os blocos são criptografados em uma ordem sequencial. Os modos de funcionamento das cifras de bloco são:

- **Electronic Codeblock (EBC):** No modo ECB, cada bloco de dados é criptografado de forma independente, usando a mesma chave. Isso significa que blocos idênticos no texto claro produzirão blocos cifrados idênticos. Isso pode resultar em vulnerabilidades,

pois um adversário pode identificar padrões no texto cifrado e realizar substituições ou reordenamentos. O ECB é geralmente considerado inseguro para criptografia de longos volumes de dados ou dados com padrões repetitivos.

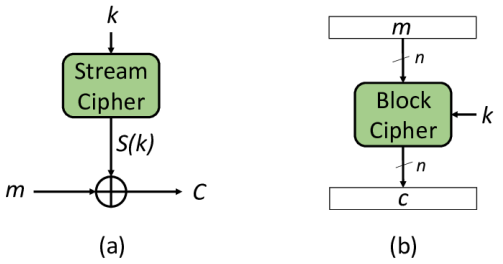
- **Cipher Block Chaining (CBC):** No modo CBC, cada bloco de dados é criptografado combinando-o com o bloco cifrado anterior antes de ser processado. Isso introduz uma dependência entre os blocos e torna o texto cifrado dependente de todos os blocos anteriores. Além disso, um vetor de inicialização (IV) é usado para iniciar o processo de encadeamento. O CBC fornece confidencialidade e integridade, pois qualquer alteração em um bloco afeta todos os blocos subsequentes. No entanto, não é paralelizável, pois cada bloco depende do bloco anterior.
- **Cipher Feedback (CFB):** No modo CFB, os blocos de dados são tratados como um fluxo de bits, em vez de blocos independentes. É utilizado um tamanho de deslocamento menor que o tamanho do bloco para criar um fluxo de bits. O fluxo de bits é então combinado com o texto claro por meio de uma operação XOR, e o resultado é o texto cifrado. O CFB é adequado para criptografia síncrona, onde os blocos de texto claro são criptografados um a um. Ele permite o uso de cifras de bloco como cifras de fluxo.
- **Output Feedback (OFB):** No modo OFB, o bloco cifrado anterior é usado para gerar uma sequência de bits pseudoaleatória, que é combinada com o texto claro por meio de uma operação XOR para produzir o texto cifrado. O OFB transforma uma cifra de bloco em uma cifra de fluxo, permitindo a criptografia de fluxos contínuos de dados. Também é adequado para transmissões assíncronas, pois não requer blocos sequenciais.
- **Counter mode (CTR):** No modo CTR, um contador é usado para gerar uma sequência de valores (normalmente chamados de nonce) que são combinados com a chave para gerar um fluxo de chave pseudoaleatório. Esse fluxo de chave é, então, combinado com o texto claro por meio de uma operação XOR para produzir o texto cifrado. Cada bloco de dados é criptografado independentemente, tornando o modo CTR altamente paralelizável.

É importante destacar que alguns modos de operação apresentam propriedades de segurança adicionais em relação a outros. Por exemplo, o modo ECB é considerado menos seguro devido à sua falta de confidencialidade quando blocos idênticos são repetidos no texto claro. O modo CBC, CFB, OFB e CTR são projetados para mitigar essa vulnerabilidade, tornando-os mais seguros em certos aspectos.

O modo ***CBC*** (***Cipher Block Chaining***) oferece confidencialidade e integridade devido à dependência entre os blocos. Ele é amplamente utilizado e oferece uma boa segurança quando implementado corretamente. No entanto, ele não é paralelizável, o que pode ser uma desvantagem em algumas aplicações.

O modo ***CTR*** (***Counter***) é altamente paralelizável, o que o torna eficiente em termos computacionais e adequado para criptografia em tempo real. Ele também evita a repetição de

blocos, desde que o contador seja usado corretamente. O CTR é amplamente adotado e considerado seguro quando implementado adequadamente.



2.Distribuição de chaves

A distribuição de chaves é um dos desafios fundamentais na criptografia simétrica. Na criptografia simétrica, a mesma chave é usada tanto para criptografar quanto para descriptografar os dados. Portanto, para garantir a confidencialidade e a segurança dos dados, é crucial que a chave seja mantida em sigilo e compartilhada apenas entre as partes autorizadas.

Existem várias abordagens para a distribuição de chaves em criptografia simétrica, e a escolha depende do contexto e dos requisitos específicos do sistema. Alguns métodos comuns de distribuição de chaves incluem:

- **Distribuição direta:** Neste método, a chave é compartilhada diretamente entre as partes autorizadas por meio de um canal seguro. Isso pode envolver a entrega física da chave em um dispositivo de armazenamento seguro ou a transmissão eletrônica da chave por meio de um canal de comunicação seguro, como criptografia de chave pública.
- **Uso de uma chave mestra:** Uma abordagem comum é o estabelecimento de uma chave mestra compartilhada entre as partes autorizadas. Essa chave mestra é usada para gerar chaves de sessão exclusivas para cada comunicação. As chaves de sessão são geradas por algoritmos de derivação de chaves, que utilizam a chave mestra e outros parâmetros (como um número de sequência ou um valor de inicialização) para gerar uma nova chave em cada sessão.

Apesar de ser amplamente utilizado, a distribuição de chaves na criptografia simétrica também enfrenta alguns problemas:

- **Chaves secretas compartilhadas:** A distribuição segura de uma chave simétrica requer um canal confiável e seguro para compartilhamento. Se um adversário interceptar ou comprometer a chave durante a distribuição, a segurança dos dados é comprometida. O desafio reside em garantir que a chave seja entregue apenas às partes autorizadas e que a chave seja mantida em sigilo.
- **Número de chaves:** Em um ambiente com várias partes comunicantes, cada par de partes deve compartilhar uma chave exclusiva. Isso implica na necessidade de gerenciar um grande número de chaves, especialmente quando o número de pares de comunicação aumenta. O gerenciamento adequado dessas chaves, incluindo sua

geração, armazenamento e atualização, pode ser complexo e requer uma infraestrutura robusta.

- **Escalabilidade:** À medida que o número de participantes aumenta, a distribuição segura de chaves se torna cada vez mais desafiadora. Aumentar a quantidade de chaves compartilhadas pode aumentar a complexidade e os custos operacionais. A distribuição eficiente de chaves em redes maiores é um problema complexo que requer soluções escaláveis e seguras.

Para mitigar esses problemas, são empregadas várias técnicas e protocolos, como a ***criptografia de chave pública*** (***assimétrica***) para estabelecer chaves compartilhadas de forma segura ou o uso de protocolos de troca de chaves como o Diffie-Hellman. Além disso, o uso de algoritmos de gerenciamento de chaves, como sistemas de gerenciamento de chaves (Key Management Systems - KMS) e infraestruturas de chave pública (Public Key Infrastructures - PKIs), pode facilitar a distribuição, armazenamento e atualização das chaves.

3.Principais algoritmos modernos de criptografia simétrica

Os principais algoritmos modernos de criptografia simétrica são projetados para garantir a confidencialidade e a integridade dos dados por meio do uso de chaves compartilhadas. Esses algoritmos, como o Advanced Encryption Standard (AES), o Twofish e o Serpent, são amplamente utilizados em diversos setores, incluindo comunicações seguras, transações financeiras e proteção de dados sensíveis.

Eles se destacam por sua eficiência computacional, resistência a ataques criptográficos e capacidade de suportar diferentes tamanhos de chave, permitindo uma adaptação flexível às necessidades de segurança de cada aplicação.

3.1 DES

O algoritmo ***DES*** (***Data Encryption Standard***) é um algoritmo de criptografia simétrica de blocos que foi amplamente utilizado durante muitos anos. Ele foi desenvolvido nos anos 1970 pela IBM em conjunto com a NSA (Agência de Segurança Nacional dos Estados Unidos) e posteriormente adotado como padrão pelo governo dos EUA.

O DES opera em blocos de dados de tamanho fixo de 64 bits e utiliza uma estrutura de rede Feistel, composta por 16 rounds de operações. Cada round consiste em uma série de etapas, incluindo permutação, substituição e combinação linear.

A etapa de permutação no DES envolve a reorganização dos bits do bloco de dados de entrada. Essa permutação, conhecida como permutação inicial, é realizada antes dos rounds e é seguida pela permutação final após a conclusão dos rounds. Essas permutações têm como objetivo garantir uma distribuição uniforme dos bits e aumentar a confusão dos dados.

A etapa de substituição é realizada usando ***S-boxes*** (***tabelas de substituição***) não lineares. **O DES utiliza um conjunto de oito S-boxes, cada uma mapeando 6 bits de entrada para**

4 bits de saída. Essas S-boxes introduzem uma operação não linear e ajudam a confundir os padrões nos dados.

A combinação linear no DES envolve operações matemáticas, como XOR e operações de deslocamento de bits, para combinar os resultados das etapas anteriores e gerar a saída final do algoritmo.

Uma característica importante do DES é o uso de uma chave de 56 bits. Durante a criptografia, a chave é expandida para gerar 16 subchaves de 48 bits cada uma, uma para cada round. Essas subchaves são derivadas por meio de um processo de permutação e rotação da chave original.

3.2 3DES

O algoritmo ***3DES*** (***Triple Data Encryption Standard***), também conhecido como TDEA, é uma extensão do algoritmo DES (Data Encryption Standard) que visa aumentar a segurança usando múltiplas aplicações do DES em sequência.

O 3DES opera em blocos de dados de tamanho fixo de 64 bits, assim como o DES, e utiliza uma estrutura de rede Feistel. Ele consiste em três etapas principais: criptografia, descriptografia e criptografia novamente (ou seja, EDE - Encrypt, Decrypt, Encrypt).

Na primeira etapa, o bloco de dados é criptografado usando uma chave de criptografia primária (K1). O algoritmo executa uma série de rounds, semelhante ao DES, que inclui permutações, substituições e combinações lineares. Isso resulta em um bloco criptografado intermediário.

Na segunda etapa, o bloco criptografado intermediário é descriptografado usando uma chave de descriptografia (K2). O algoritmo realiza o processo inverso da criptografia, desfazendo as etapas de substituição e permutação aplicadas anteriormente. Isso resulta em um bloco descriptografado.

Finalmente, **na terceira etapa**, o bloco descriptografado é criptografado novamente usando uma segunda chave de criptografia (K3). Essa etapa adiciona outra camada de segurança ao aplicar novamente as permutações, substituições e combinações lineares.

O 3DES pode ser usado em diferentes modos de operação, como ECB (Electronic Codebook), CBC (Cipher Block Chaining), entre outros, para processar blocos de dados maiores ou fluxos contínuos de dados.

O uso do 3DES aumenta a segurança em relação ao DES original, pois adiciona uma camada adicional de criptografia. No entanto, devido ao seu processamento mais complexo e maior quantidade de operações, o 3DES é mais lento que o DES.

Nos últimos anos, tem sido gradualmente substituído pelo AES (Advanced Encryption Standard) devido ao seu desempenho superior e maior segurança oferecida pelo AES.

3.3 RC4

O algoritmo ***RC4*** (***Rivest Cipher 4***) é um algoritmo de criptografia de fluxo, também conhecido como cifra de fluxo. Ele é amplamente utilizado em diversas aplicações, como protocolos de segurança, redes sem fio e sistemas de segurança de dados.

O RC4 opera gerando uma sequência pseudoaleatória de bytes, conhecida como keystream, que é combinada com o texto claro por meio de uma operação XOR (OU exclusivo). O algoritmo possui duas partes principais: a etapa de inicialização e a geração do keystream.

Durante a etapa de inicialização, o RC4 requer uma chave de tamanho variável, geralmente entre 40 e 2048 bits. Essa chave é usada para inicializar um vetor de estado interno do RC4 e permutar os elementos do vetor com base na chave. Isso cria um estado interno inicializado para o algoritmo.

Após a inicialização, o algoritmo entra na fase de geração do ***keystream***. Nessa fase, o vetor de estado é percorrido e seus elementos são alterados com base em uma combinação complexa de trocas e permutações. Essa operação gera uma sequência pseudoaleatória de bytes que compõem o keystream.

Em seguida, o keystream é combinado com o texto claro por meio de uma operação XOR bit a bit. Cada byte do texto claro é combinado com o byte correspondente do keystream, resultando no texto cifrado. Essa operação é reversível, ou seja, aplicar a operação XOR novamente ao texto cifrado com o mesmo keystream produzirá o texto claro original.

É importante mencionar que o RC4 foi amplamente utilizado no passado, mas foram descobertos alguns problemas de segurança relacionados ao seu uso em determinadas situações. Por esse motivo, recomenda-se o uso de algoritmos mais modernos e seguros, como o Advanced Encryption Standard (AES), em vez do RC4, para aplicações criptográficas mais recentes.

3.5 RC6

O algoritmo ***RC6*** (***Rivest Cipher 6***) é um algoritmo de criptografia simétrica de blocos, desenvolvido por Ron Rivest em 1997. Ele é projetado para oferecer segurança, eficiência e flexibilidade em termos de tamanho de chave.

O RC6 opera em blocos de dados de tamanho fixo, normalmente de 128 bits, e **é baseado em uma estrutura de rede Feistel**, assim como o algoritmo Blowfish. O processo de criptografia/descriptografia envolve várias etapas, incluindo a expansão de chaves, a permutação, a substituição e a combinação linear.

A expansão de chaves no RC6 envolve a geração de uma matriz de subchaves a partir da chave original. Essas subchaves são derivadas por meio de uma série de transformações, como rotações de bits e operações de adição modular.

A etapa de permutação envolve a reorganização dos bits do bloco de dados de entrada. O RC6 usa permutações lineares para garantir uma difusão adequada dos dados e fornecer uma operação não linear.

A substituição é realizada usando ***S-boxes*** (***tabelas de substituição***) não lineares. Essas S-boxes mapeiam os valores de entrada para os valores de saída, introduzindo não linearidade e dificultando a análise estatística.

A combinação linear no RC6 envolve operações matemáticas, como XOR e operações de multiplicação em um corpo finito. Essas operações são aplicadas para combinar os resultados das etapas anteriores e gerar a saída final do algoritmo.

O RC6 é considerado um algoritmo seguro e eficiente, projetado para resistir a diversos ataques criptográficos conhecidos. Ele suporta tamanhos de chave de 128, 192 e 256 bits, o que proporciona flexibilidade em termos de níveis de segurança e requisitos de aplicação.

3.6 BlowFish

O algoritmo Blowfish é um algoritmo de criptografia simétrica de blocos projetado por Bruce Schneier em 1993. É um algoritmo de chave variável, o que significa que pode trabalhar com chaves de tamanho variável, de 32 bits a 448 bits.

O Blowfish opera em blocos de dados de tamanho fixo (normalmente 64 bits) e utiliza uma estrutura de rede Feistel, que consiste em repetir uma série de transformações em cada bloco de dados. O algoritmo é dividido em duas partes principais: a etapa de inicialização e a etapa de criptografia/descriptografia.

Durante a etapa de inicialização, a chave é expandida em uma série de subchaves usando uma função chamada de ***Subkey Generation***. Essa função divide a chave em várias partes e aplica uma série de iterações complexas para gerar as subchaves que serão usadas nas transformações subsequentes.

Após a inicialização, o algoritmo entra na fase de criptografia/descriptografia. O bloco de dados de entrada é dividido em duas metades, e cada metade passa por uma série de iterações chamadas de Rounds. Durante cada Round, ocorre uma mistura das metades do bloco usando funções não lineares, combinações com as subchaves e operações XOR.

O número de Rounds realizados depende do tamanho da chave. Para chaves de 32 a 64 bits, são realizados 16 Rounds; para chaves de 80 a 128 bits, são realizados 18 Rounds; e para chaves de 136 a 448 bits, são realizados 20 Rounds.

Uma característica importante do Blowfish é que ele é um algoritmo rápido e eficiente em termos de desempenho, tornando-o adequado para uma ampla gama de aplicações. No entanto, devido ao avanço criptográfico, como o aumento do poder computacional, o Blowfish pode ser considerado relativamente menos seguro atualmente.

3.7 TwoFish

O algoritmo Twofish é um algoritmo de criptografia simétrica de blocos que foi finalista no concurso ***AES*** (***Advanced Encryption Standard***) em 2000. Ele foi projetado por Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall e Niels Ferguson. O Twofish é considerado um algoritmo altamente seguro e eficiente.

O Twofish opera em blocos de dados de tamanho fixo, normalmente de 128 bits, e usa uma estrutura de rede Feistel, semelhante ao algoritmo Blowfish. O processo de criptografia/descriptografia envolve quatro partes principais: expansão de chaves, permutação, substituição e combinação linear.

A expansão de chaves é realizada usando o algoritmo de chave estendida de Feistel, que transforma a chave original em várias subchaves. Essas subchaves são usadas em cada rodada do algoritmo para adicionar complexidade e segurança.

A permutação é uma etapa importante no Twofish e envolve reorganizar os bits do bloco de dados de entrada. A permutação é realizada usando permutações lineares para garantir uma difusão adequada dos dados e fornecer uma operação não linear.

A substituição é feita usando as chamadas "caixas-S" (***S-boxes***). As S-boxes são tabelas de substituição não lineares que mapeiam valores de entrada para valores de saída. O Twofish usa várias S-boxes para introduzir não linearidade e confundir os padrões de dados durante a criptografia.

A combinação linear envolve operações matemáticas como XOR e multiplicação em um corpo finito para combinar os resultados das etapas anteriores e produzir a saída final.

O Twofish é altamente considerado por sua segurança e resistência a ataques criptográficos conhecidos. Ele suporta tamanhos de chave de 128, 192 e 256 bits, o que o torna uma escolha flexível para diferentes níveis de segurança.

3.8 Serpent

O algoritmo Serpent é um algoritmo de criptografia simétrica de blocos, projetado por Ross Anderson, Eli Biham e Lars Knudsen. Foi um dos finalistas no concurso AES (Advanced Encryption Standard) em 2000. O Serpent é conhecido por sua segurança e robustez, sendo amplamente utilizado em várias aplicações criptográficas.

O Serpent opera em blocos de dados de tamanho fixo, geralmente de 128 bits, e também utiliza uma estrutura de rede Feistel, semelhante ao Blowfish e ao Twofish. O processo de criptografia/descriptografia envolve várias etapas, incluindo expansão de chaves, substituição, permutação e combinação linear.

A expansão de chaves no Serpent envolve a geração de um conjunto de subchaves a partir da chave original. Essas subchaves são derivadas por meio de uma série de transformações e iterações complexas, garantindo uma maior segurança e confidencialidade dos dados.

A etapa de substituição é realizada usando ***S-boxes*** (tabelas de substituição) não lineares. O Serpent utiliza um conjunto de S-boxes altamente não lineares e criptograficamente seguras para confundir os padrões dos dados e dificultar a análise estatística.

A permutação no Serpent é feita usando uma combinação de permutações lineares e não lineares. Essas permutações reorganizam os bits do bloco de dados para garantir uma difusão adequada e proporcionar uma operação não linear.

A combinação linear envolve operações matemáticas, como XOR e operações de multiplicação em um corpo finito. Essas operações são aplicadas para combinar os resultados das etapas anteriores e gerar a saída final do algoritmo.

O Serpent é conhecido por sua segurança e resistência a ataques criptográficos conhecidos. Ele suporta tamanhos de chave de 128, 192 e 256 bits, proporcionando um alto nível de segurança para diferentes aplicações.

3.9 AES

O algoritmo ***AES*** (***Advanced Encryption Standard***), também conhecido como Rijndael, é um dos algoritmos de criptografia simétrica mais amplamente utilizados e foi escolhido como o novo padrão de criptografia pelo NIST (Instituto Nacional de Padrões e Tecnologia dos Estados Unidos) em 2001.

O AES opera em blocos de dados de tamanho fixo de 128 bits e utiliza uma estrutura de rede Feistel, com algumas diferenças em relação aos algoritmos anteriores, como o DES. O processo de criptografia/descriptografia envolve quatro etapas principais: SubBytes, ShiftRows, MixColumns e AddRoundKey.

A etapa SubBytes envolve a substituição dos bytes do bloco de dados de entrada por meio de uma ***S-box não linear***. Essa S-box mapeia cada byte de entrada para um byte de saída usando uma tabela de substituição.

A etapa ***ShiftRows*** envolve o deslocamento dos bytes nas linhas do bloco de dados. Cada linha é deslocada para a esquerda, onde o primeiro byte permanece no lugar, o segundo é deslocado uma posição, o terceiro duas posições e o quarto três posições.

A etapa ***MixColumns*** envolve a combinação linear dos bytes em cada coluna do bloco de dados. Isso é feito multiplicando cada byte por uma matriz específica e reduzindo o resultado a um polinômio de grau menor.

A etapa ***AddRoundKey*** envolve a operação XOR entre cada byte do bloco de dados e uma chave de round correspondente. Essa operação combina os dados do bloco com a subchave do round atual.

Essas etapas são repetidas várias vezes, dependendo do tamanho da chave utilizada (10 rounds para uma chave de 128 bits, 12 rounds para uma chave de 192 bits e 14 rounds para uma chave de 256 bits). Cada round consiste nas quatro etapas mencionadas acima, exceto a etapa MixColumns, que é omitida no último round.

O AES é conhecido por sua segurança e resistência a ataques criptográficos conhecidos. Ele suporta tamanhos de chave de 128, 192 e 256 bits, proporcionando diferentes níveis de segurança para várias aplicações. Sua eficiência e ampla adoção em uma ampla gama de setores o tornam um dos algoritmos criptográficos mais confiáveis e amplamente utilizados atualmente.