Módulo 8 - Aulas 1 e 2

Módulo 8: Conceitos de criptografia

Aula 1: Propriedades da criptografia

Objetivos

- [ ] Compreender as propriedades da comunicação criptografada.
- [ ] Conhecer os conceitos fundamentais da criptografia em dados.
- [ ] Explorar técnicas de ocultação de dados.

Conceitos

- [ ] Confusão.
- [ ] Difusão.
- [ ] Esteganografia.

Introdução

Nesta aula, vamos explorar o emocionante mundo da criptografia e técnicas de ocultação de dados. Prepare-se para adentrar em um universo onde segredos são protegidos, mensagens são codificadas e a arte de esconder informações se revela. Neste ambiente, encontraremos personagens fascinantes como Alice, Bob, Mallory e outros, cada um com seu papel na dança criptográfica.

Ao mergulhar nesse tópico, vamos desvendar as propriedades dos algoritmos criptográficos, que são a base de toda a segurança da informação. Entenderemos conceitos fundamentais como confusão, difusão e colisão, que moldam a maneira

como os dados são transformados e protegidos. Exploraremos o intrigante mundo da criptografia em dados, descobrindo como a informação é codificada de forma segura e tornada inacessível a olhares indiscretos.

Também abordaremos técnicas de ocultação de dados, como a esteganografia e a ofuscação. Aprenderemos como mensagens podem ser escondidas dentro de arquivos aparentemente inofensivos, ou como programas podem ser alterados para dificultar sua compreensão. Prepare-se para uma jornada cheia de mistérios, desafios e segredos ocultos.

Propriedades da comunicação com criptografia

Criptografia

Criptografia é a prática de transformar informações legíveis em um formato ilegível, chamado de texto cifrado, com o objetivo de proteger a confidencialidade, integridade e autenticidade dessas informações. Ela envolve o uso de algoritmos matemáticos e chaves criptográficas para realizar a transformação dos dados. A criptografia é amplamente utilizada para proteger dados sensíveis durante a transmissão pela internet, armazenamento em dispositivos e comunicação entre sistemas. Apenas aqueles que possuem a chave correta podem decifrar o texto cifrado e obter acesso às informações originais. A criptografia desempenha um papel fundamental na segurança da informação e é uma das principais técnicas para garantir a privacidade e segurança dos dados.

Alice, Bob, Mallory e outros

O uso dos protocolos de criptografia sempre considera personagens que participam de alguma maneira no tratamento da informação. Esses personagens são:

- Alice: É geralmente a entidade que deseja enviar uma mensagem de forma segura. Ela pode representar um remetente legítimo ou uma parte autorizada.
- Bob: É o destinatário da mensagem enviada por Alice. Bob também pode ser um receptor legítimo ou uma parte autorizada.
-

Mallory: É um personagem mal-intencionado que tenta interceptar ou manipular a comunicação entre Alice e Bob. Mallory é frequentemente usado como um exemplo de um atacante na criptografia.

- Eve: É uma personagem que representa um observador externo, geralmente conhecido como "espião" ou "eavesdropper". Eve tenta interceptar as mensagens enviadas entre Alice e Bob para obter acesso não autorizado às informações.
- Trent: É um personagem neutro e confiável que atua como uma autoridade de certificação ou intermediário confiável. Trent pode autenticar as identidades de Alice e Bob e facilitar a comunicação segura entre eles.
- Oscar: Também conhecido como "Oscuro", é um personagem que representa um adversário avançado ou hacker experiente. Oscar usa técnicas sofisticadas para tentar quebrar a segurança dos sistemas criptográficos.
- Charlie: É um personagem adicional que representa um intermediário ou canal de comunicação. Charlie pode ser um servidor de mensagens, roteador ou qualquer entidade que facilite a transmissão das mensagens entre Alice e Bob.
- Carol: Carol é outro personagem que pode ser usado em exemplos de criptografia. Assim como Alice e Bob, Carol pode ser uma parte legítima envolvida na troca de informações seguras.

Alice e Bob

Confusão

Em criptografia, o conceito de confusão refere-se a uma propriedade dos algoritmos criptográficos que visa tornar a relação entre a chave de criptografia e o texto cifrado tão complexa quanto possível. A confusão é alcançada por meio do

embaralhamento e da dispersão das características dos dados de entrada, dificultando a identificação de padrões ou informações úteis pelos adversários.

Ao aplicar técnicas de confusão, um algoritmo criptográfico busca garantir que pequenas mudanças nos dados de entrada ou na chave resultem em grandes alterações nos dados de saída, tornando a relação entre eles imprevisível. Isso torna mais difícil para um adversário analisar o texto cifrado e deduzir informações sobre a chave ou os dados originais.

A confusão contribui para a segurança global do sistema criptográfico, tornando a análise criptográfica mais desafiadora e exigindo um grande esforço computacional para quebrar a criptografia. É uma das propriedades fundamentais para garantir a resistência da criptografia a ataques de força bruta, análise estatística e outras técnicas de criptoanálise.

Um exemplo prático de confusão na criptografia é o uso de tabelas de substituição, como a Tabela de Substituição S-Box no algoritmo de criptografia simétrica AES (Advanced Encryption Standard). A S-Box é uma tabela de substituição não linear que é aplicada aos dados durante o processo de criptografia.

Nesse exemplo, cada byte do texto de entrada é substituído por outro byte específico da tabela S-Box, de acordo com uma determinada lógica. Essa substituição confunde a relação entre o byte de entrada e o byte de saída, tornando a análise do texto cifrado mais difícil.

Por exemplo, se o byte de entrada for "10100110", a S-Box pode substituí-lo pelo byte "01101000". Essa substituição não é óbvia e não segue uma relação linear direta. Portanto, mesmo que um adversário conheça o byte de entrada, é difícil deduzir o byte de saída ou a tabela S-Box sem o conhecimento da chave de criptografia.

Difusão

Em criptografia, o conceito de difusão refere-se a um dos princípios fundamentais dos algoritmos criptográficos. A ideia por trás da difusão é garantir que qualquer alteração mínima nos dados de entrada cause uma mudança significativa nos dados de saída. Isso significa que uma pequena modificação nos bits de entrada

deve ser propagada para vários bits diferentes no texto cifrado, espalhando os efeitos e tornando a relação entre os dados originais e os dados criptografados o mais complexa possível.

A difusão é alcançada por meio de uma série de transformações e operações aplicadas aos dados durante o processo de criptografia. Essas operações podem incluir substituições, permutações, misturas e outras técnicas para garantir que cada bit de entrada tenha um impacto significativo em múltiplos bits de saída. O objetivo é distribuir as propriedades estatísticas dos dados originais de maneira uniforme em todo o texto criptografado, o que dificulta a detecção de padrões e relações entre os bits.

Por exemplo, considere um algoritmo criptográfico que opera em blocos de 128 bits. Se um único bit de entrada for alterado, o processo de difusão garantirá que a alteração se propague para vários bits diferentes nos blocos de saída. Essa propagação é geralmente obtida por meio de funções de substituição não lineares, que transformam os bits de entrada em novos valores com base em tabelas ou caixas de substituição. Além disso, são aplicadas permutações e transformações matemáticas complexas para garantir que os bits de saída sejam totalmente diferentes dos bits de entrada, dificultando a reconstrução dos dados originais.

A difusão desempenha um papel crucial na segurança dos algoritmos criptográficos, pois garante que até mesmo uma pequena mudança nos dados de entrada cause uma mudança drástica nos dados criptografados. Isso torna a tarefa de descobrir padrões ou relações entre os dados extremamente difícil para um adversário. A difusão é um dos pilares da criptografia moderna e contribui para a resistência dos algoritmos contra ataques de criptoanálise, tornando a recuperação dos dados originais a partir do texto criptografado praticamente impossível sem a chave correta.

Um exemplo prático e simples de difusão em criptografia pode ser ilustrado pelo algoritmo conhecido como "XOR". Considere dois bits de entrada, A e B, e uma operação de difusão realizada através do operador XOR (ou exclusivo).

Suponha que temos os seguintes valores para os bits de entrada: A = 1 B = 0

A operação XOR compara os bits A e B e produz um resultado que é 1 se os bits forem diferentes e 0 se forem iguais. Nesse caso, temos: A XOR B = 1 XOR 0 = 1

Agora, vamos supor que alteramos o valor de apenas um dos bits de entrada. Se modificarmos o bit B para 1, teremos: A = 1 B = 1

Novamente, aplicamos a operação XOR: A XOR B = 1 XOR 1 = 0

Perceba que uma pequena mudança em um dos bits de entrada (de 0 para 1) resultou em uma mudança completa no resultado (de 1 para 0). Essa é a difusão em ação.

Confusão e Difusão

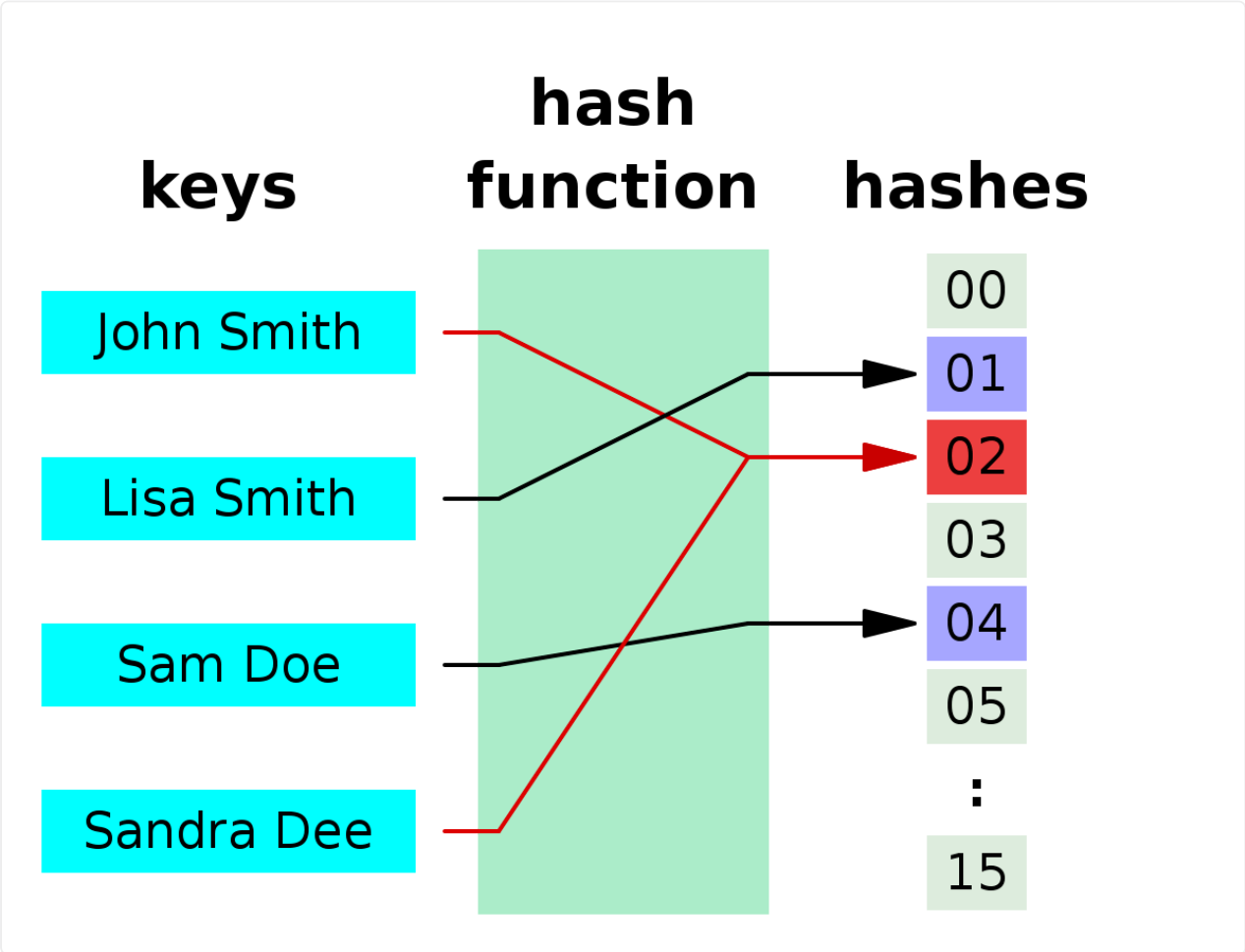
Colisão

O conceito de "colisão" em criptografia está relacionado às funções hash, que são algoritmos que transformam um conjunto de dados em um valor de tamanho fixo. Uma colisão ocorre quando dois conjuntos de dados diferentes produzem o mesmo valor de hash. Em outras palavras, é quando duas entradas distintas geram o mesmo resumo criptográfico.

Para entender melhor, vamos considerar um exemplo simplificado. Suponha que temos uma função hash que recebe como entrada uma sequência de números e produz um resumo de tamanho fixo. Se tivermos duas sequências diferentes, por exemplo, [1, 2, 3] e [4, 5, 6], e ambas produzirem o mesmo resumo de hash, temos uma colisão.

A colisão é indesejável na criptografia, pois compromete a integridade dos dados. Se um invasor puder encontrar duas entradas diferentes que geram o mesmo valor de hash, ele pode substituir os dados originais por dados maliciosos sem que seja detectado pelo resumo criptográfico. Isso pode levar a sérias vulnerabilidades em sistemas que dependem da integridade dos dados.

Os algoritmos de função hash modernos, como o SHA-256 (Secure Hash Algorithm 256 bits), foram projetados para ter uma probabilidade muito baixa de colisões, tornando-as extremamente improváveis. No entanto, é importante ressaltar que não há garantia matemática de que uma colisão nunca ocorrerá. A segurança das funções hash é baseada na dificuldade prática de encontrar uma colisão, levando em consideração o tamanho do resumo e a qualidade do algoritmo utilizado.



Colisão na saída 02

Criptografia em dados em repouso

A criptografia em dados em repouso é uma técnica utilizada para proteger informações armazenadas em dispositivos de armazenamento, como discos rígidos, servidores, bancos de dados, pen drives, entre outros. O objetivo é garantir que mesmo que um invasor tenha acesso físico aos dispositivos, os dados permaneçam inacessíveis e não possam ser compreendidos ou utilizados sem a chave de criptografia adequada.

Ao aplicar a criptografia em dados em repouso, os dados são transformados em um formato ilegível chamado de texto cifrado. Esse processo de transformação é realizado por meio de algoritmos criptográficos, que são projetados para serem reversíveis, ou seja, é possível decifrar os dados utilizando a chave correta.

A criptografia em dados em repouso pode ser implementada de diferentes maneiras. Uma abordagem comum é a criptografia de disco, em que todo o conteúdo do disco é criptografado, incluindo o sistema operacional, arquivos e estruturas de diretório. Isso garante que todos os dados no disco permaneçam protegidos, independentemente de onde estejam armazenados.

Outra técnica é a criptografia de arquivos ou pastas específicas, em que apenas determinados arquivos ou diretórios são criptografados. Isso pode ser útil quando se deseja proteger informações confidenciais específicas, enquanto outros dados permanecem acessíveis de forma convencional.

A segurança da criptografia em dados em repouso depende de vários fatores, como a robustez dos algoritmos criptográficos utilizados, o comprimento e a complexidade das chaves de criptografia, e a proteção adequada das chaves de criptografia. É importante também implementar práticas adequadas de gerenciamento de chaves, como o armazenamento seguro das chaves e a rotação regular das mesmas para garantir a segurança contínua dos dados criptografados.

Criptografia em dados em trânsito

A criptografia em dados em trânsito refere-se à aplicação de técnicas criptográficas para proteger informações enquanto são transferidas ou transmitidas através de redes de comunicação. O objetivo é garantir que os dados não sejam

interceptados ou alterados por pessoas não autorizadas durante a transmissão, mantendo sua confidencialidade e integridade.

Quando os dados são transmitidos em redes, como a internet, eles geralmente passam por vários pontos intermediários, como roteadores, servidores e provedores de serviços. Esses pontos intermediários podem representar possíveis pontos de vulnerabilidade, onde um invasor poderia interceptar ou manipular os dados em trânsito. A criptografia em dados em trânsito atua como uma camada de proteção para mitigar essas ameaças.

Existem diferentes protocolos e algoritmos de criptografia projetados para proteger a comunicação em rede. Um exemplo comum é o protocolo HTTPS (Hypertext Transfer Protocol Secure), que utiliza o SSL/TLS (Secure Sockets Layer/Transport Layer Security) para criptografar a comunicação entre um cliente (navegador) e um servidor da web. Ao estabelecer uma conexão HTTPS, os dados são criptografados antes de serem enviados e só podem ser decifrados pelo destinatário legítimo que possui a chave de criptografia correspondente.

A criptografia em dados em trânsito utiliza algoritmos criptográficos simétricos e assimétricos para proteger a comunicação. Algoritmos simétricos, como o AES (Advanced Encryption Standard), usam uma única chave compartilhada entre o remetente e o destinatário para criptografar e descriptografar os dados. Algoritmos assimétricos, como o RSA (Rivest-Shamir-Adleman), envolvem o uso de um par de chaves, uma pública e outra privada, onde a chave pública é usada para criptografar os dados e a chave privada correspondente é usada para descriptografá-los.

A criptografia em dados em trânsito oferece proteção contra diversos tipos de ataques, como a interceptação de dados (sniffing), onde um invasor captura os dados enquanto eles são transmitidos, e a modificação de dados (tampering), onde um invasor altera os dados durante a transmissão. Ao criptografar os dados em trânsito, mesmo que um invasor consiga interceptá-los, eles estarão em um formato ilegível sem a chave correta.

Criptografia em dados em uso

A criptografia em dados em uso refere-se à aplicação de técnicas criptográficas para proteger informações enquanto estão sendo processadas ou utilizadas por um sistema ou aplicativo. Ao contrário da criptografia em dados em repouso (armazenados) ou em trânsito (transmitidos), a criptografia em dados em uso visa proteger as informações enquanto estão em execução em um ambiente computacional.

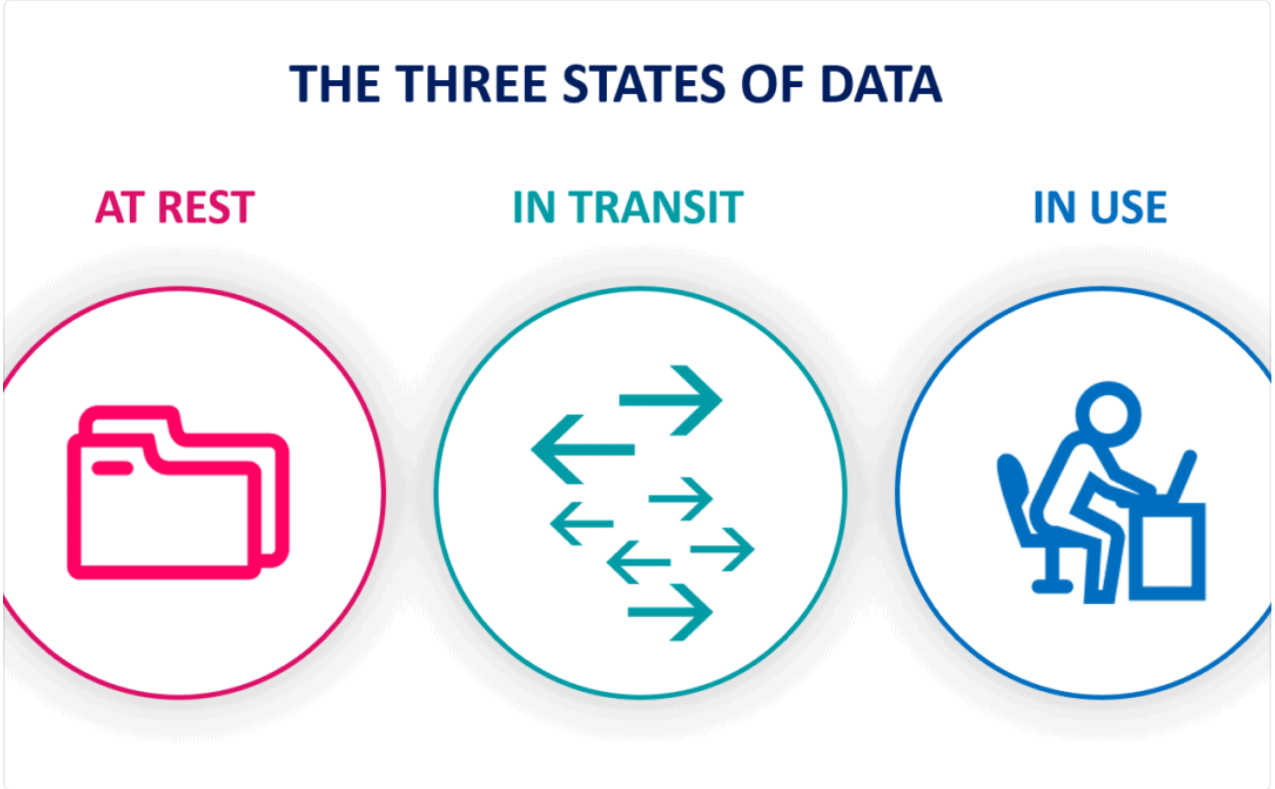
Quando os dados estão em uso, eles podem ser acessados e processados por diferentes componentes de um sistema, como processadores, memória, caches e dispositivos de entrada e saída. No entanto, esses componentes também podem representar possíveis pontos de vulnerabilidade, onde um invasor pode tentar obter acesso não autorizado aos dados em uso.

A criptografia em dados em uso utiliza técnicas para proteger os dados sensíveis enquanto estão sendo processados. Isso envolve a aplicação de algoritmos criptográficos que garantem a confidencialidade e integridade dos dados, mesmo quando estão sendo manipulados ou processados por diferentes partes do sistema.

Uma abordagem comum na criptografia em dados em uso é a utilização de técnicas de criptografia homomórfica, que permitem a execução de operações sobre os dados criptografados sem a necessidade de descriptografá-los. Com a criptografia homomórfica, é possível realizar cálculos e processamentos em dados criptografados, preservando sua confidencialidade e protegendo as informações contra vazamentos durante o processamento.

Além disso, a criptografia em dados em uso também pode envolver o uso de técnicas de isolamento e proteção de memória para evitar vazamentos de informações sensíveis. Mecanismos de isolamento, como contêineres e máquinas virtuais, podem ser empregados para criar ambientes seguros onde os dados em uso são protegidos contra acessos não autorizados.

A criptografia em dados em uso desempenha um papel importante na proteção da privacidade e segurança das informações sensíveis durante o processamento. Ela garante que os dados permaneçam protegidos, mesmo quando estão sendo utilizados por sistemas, aplicativos ou algoritmos, minimizando o risco de vazamento de informações confidenciais.



Os 3 estados dos dadots: repouso, em trânsito e em uso

Perfect Forward Secrecy (PFS)

Também conhecido como Sigilo Adiante Perfeito, é um conceito na criptografia que garante a confidencialidade dos dados mesmo que as chaves de criptografia sejam comprometidas no futuro. É um recurso importante para proteger a comunicação online e evitar que um atacante obtenha acesso aos dados criptografados.

Em uma comunicação segura, normalmente são usadas chaves criptográficas para criptografar e descriptografar os dados. O problema é que se a chave criptográfica for comprometida, todos os dados criptografados com essa chave podem ser decifrados pelo atacante. O PFS resolve esse problema ao gerar chaves efêmeras, também chamadas de chaves de sessão, para cada comunicação individual.

O PFS garante que mesmo que um atacante comprometa uma chave criptográfica usada em uma comunicação específica, as chaves de sessão anteriores e futuras permanecerão seguras. Isso é possível graças a um acordo de chaves efêmeras feito durante o processo de estabelecimento da comunicação. Essas chaves efêmeras são usadas apenas para essa comunicação específica e não são armazenadas ou reutilizadas posteriormente.

O uso do PFS adiciona uma camada adicional de segurança, pois mesmo que um atacante consiga comprometer as chaves de criptografia usadas em um determinado momento, não poderá decifrar as comunicações anteriores ou futuras. Isso é especialmente relevante em cenários em que o armazenamento de dados criptografados é prolongado, como em servidores ou logs de comunicação.

Uma implementação comum do PFS é feita usando protocolos de troca de chaves Diffie-Hellman, como o Diffie-Hellman de curvas elípticas (ECDH). Esses protocolos permitem que as partes envolvidas em uma comunicação estabeleçam uma chave compartilhada sem revelar a chave real durante o processo de negociação.

Perfect Forward Secrecy

Paradoxo do aniversário

O Paradoxo do Aniversário é um fenômeno matemático que envolve a probabilidade de duas ou mais pessoas em um grupo compartilharem a mesma data de aniversário. Apesar de ser chamado de paradoxo, ele recebe esse nome devido à surpreendente conclusão de que a probabilidade é maior do que muitas pessoas imaginam.

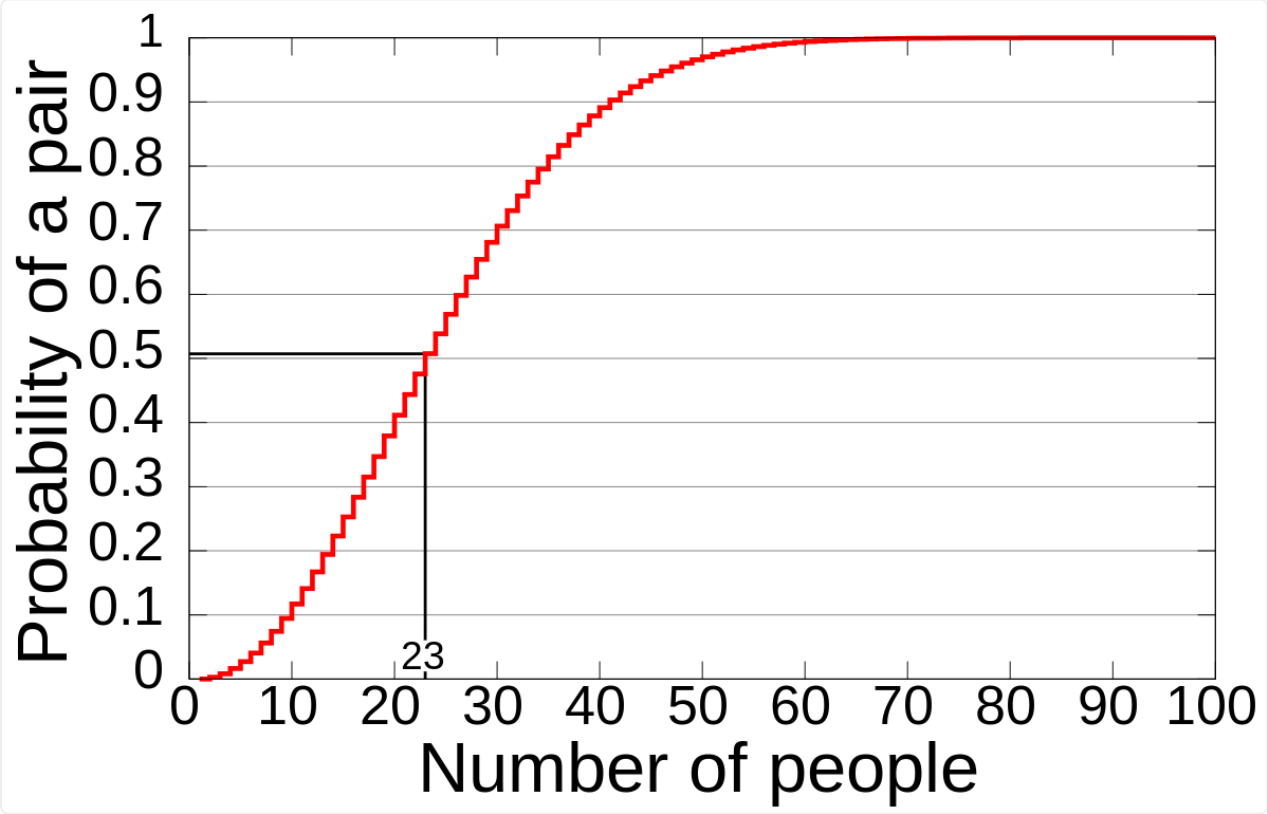
Para entender o Paradoxo do Aniversário, é importante considerar o número total de dias em um ano, que é 365. No entanto, em grupos relativamente pequenos de pessoas, a probabilidade de que duas delas compartilhem o mesmo dia de aniversário é maior do que o esperado.

Isso ocorre devido ao conceito de combinações. À medida que o número de pessoas em um grupo aumenta, o número de possíveis combinações de pares de pessoas também aumenta exponencialmente. Por exemplo, em um grupo de 23 pessoas, existem 253 combinações possíveis de pares.

A probabilidade de que todas as pessoas tenham datas de aniversário diferentes é de aproximadamente 0,4927 (ou seja, cerca de 49,27%). Portanto, a probabilidade de que pelo menos duas pessoas compartilhem o mesmo dia de aniversário é de aproximadamente 0,5073 (ou seja, cerca de 50,73%). Esse resultado é surpreendente, pois muitas pessoas tendem a pensar que a probabilidade é significativamente menor.

Essa probabilidade aumenta à medida que o número de pessoas no grupo aumenta. Por exemplo, com 50 pessoas, a probabilidade de compartilhar o mesmo dia de aniversário é de cerca de 97%. Com 70 pessoas, a probabilidade aumenta para quase 100%.

O Paradoxo do Aniversário tem aplicações na criptografia ao lidar com o conceito de colisão. Em criptografia, colisão refere-se à ocorrência de duas ou mais mensagens diferentes que resultam no mesmo valor de hash. O Paradoxo do Aniversário ajuda a destacar a importância de algoritmos criptográficos que possam resistir a ataques de colisão, garantindo que a probabilidade de colisão seja extremamente baixa.



Probabilidade de acontecimento do Paradoxo do aniversário

Técnicas de ocultação de dados

As técnicas de ocultação de dados são métodos utilizados para esconder informações sensíveis em outros tipos de arquivos ou meios de comunicação, de modo que essas informações não sejam facilmente detectadas por terceiros. O objetivo principal dessas técnicas é garantir a confidencialidade e a integridade dos dados, tornando-os menos suscetíveis a interceptação ou descoberta por indivíduos não autorizados.

Diferentemente da criptografia, que se concentra na transformação dos dados em um formato ilegível por meio do uso de algoritmos matemáticos, as técnicas de ocultação de dados buscam camuflar as informações sensíveis dentro de arquivos ou estruturas aparentemente comuns. Isso pode incluir a inserção de dados em áreas não utilizadas de um arquivo, modificação de bits específicos em arquivos de imagem ou áudio, ou até mesmo a utilização de esteganografia, que é a prática de esconder dados dentro de outros tipos de arquivos sem que isso seja perceptível.

Embora a criptografia e as técnicas de ocultação de dados compartilhem o objetivo comum de proteger informações sensíveis, elas são abordagens distintas. Enquanto a criptografia se concentra em transformar os dados em um formato ilegível para terceiros, as técnicas de ocultação de dados procuram tornar as informações menos detectáveis ou disfarçadas dentro de outros arquivos ou meios de comunicação.

É importante ressaltar que, embora as técnicas de ocultação de dados possam fornecer uma camada adicional de segurança, elas não substituem a criptografia em si. Ambas as abordagens podem ser usadas em conjunto para fornecer uma proteção mais robusta aos dados sensíveis, garantindo não apenas a confidencialidade, mas também a integridade e autenticidade das informações.

Esteganografia

A esteganografia é uma técnica que se concentra em ocultar informações dentro de outros tipos de arquivos, sem levantar suspeitas de que esses dados estejam presentes. Diferente da criptografia, que transforma os dados em uma forma

ilegível, a esteganografia busca esconder os dados de forma que eles passem despercebidos.

O objetivo principal da esteganografia é garantir a confidencialidade das informações, tornando-as imperceptíveis a olho nu ou a métodos convencionais de detecção. Essa técnica utiliza arquivos de mídia, como imagens, áudios ou vídeos, como meio para ocultar os dados. Os dados são inseridos de forma que não alterem a aparência visual ou auditiva do arquivo, tornando-se indistinguíveis para um observador desavisado.

Um exemplo prático de esteganografia é a técnica de ocultação de dados em uma imagem. Suponha que você tenha uma imagem em formato JPEG e deseje ocultar uma mensagem dentro dela. O processo envolve a alteração de bits específicos na imagem para representar a mensagem oculta. Os bits menos significativos de certos pixels podem ser modificados para armazenar os dados da mensagem, enquanto a alteração não será perceptível ao visualizar a imagem.

Ao compartilhar a imagem com outra pessoa, ela pode recuperar a mensagem oculta usando um software ou algoritmo específico. Para um observador comum, a imagem parecerá normal e não haverá indícios de que há dados ocultos nela.

A esteganografia é uma técnica poderosa para comunicação secreta, pois disfarça a própria existência dos dados ocultos. No entanto, é importante notar que a esteganografia não fornece proteção contra interceptação ou alteração dos dados. Ela se concentra apenas em ocultar a presença dos dados, não em criptografá-los. Portanto, a combinação de esteganografia com criptografia pode fornecer uma camada adicional de segurança para proteger as informações sensíveis.

Ofuscação

A ofuscação é uma técnica de ocultação de dados que tem como objetivo dificultar a compreensão do conteúdo por parte de terceiros não autorizados. Ao contrário da criptografia, que utiliza algoritmos matemáticos para transformar os dados em um formato incompreensível, a ofuscação se concentra em tornar o código ou programa mais complexo e difícil de entender, sem alterar sua funcionalidade.

Na ofuscação, técnicas como renomeação de variáveis, substituição de trechos de código por versões equivalentes mais obscuras, inserção de instruções irrelevantes ou redundantes e reorganização da estrutura do código são aplicadas. Essas medidas visam tornar o código fonte ilegível ou confuso, dificultando a análise reversa e a compreensão das funcionalidades e lógicas implementadas.

Existem várias ferramentas e programas disponíveis para a ofuscação de código, sendo que a escolha depende da linguagem de programação e dos requisitos específicos do projeto. A seguir, vou listar alguns dos programas mais utilizados para a ofuscação de código em diferentes linguagens:

- Java: ProGuard, Allatori, DashO, DexGuard.
- JavaScript: UglifyJS, Closure Compiler, JavaScript Obfuscator.
- .NET: Dotfuscator, Eazfuscator.NET, ConfuserEx.
- Python: PyArmor, Pyminifier, Pyobfuscate.
- C/C++: Themida, UPX, GNU obfuscator.

Ofuscação não é criptografia.

Fragmentação

Refere-se à divisão de um dado em partes menores, conhecidas como fragmentos, antes de aplicar a criptografia. Essa técnica é utilizada principalmente para melhorar a segurança e proteção dos dados, especialmente em ambientes onde a transmissão ou armazenamento dos dados pode ser sujeita a interceptação ou ataques.

Ao fragmentar os dados, o conteúdo original é dividido em várias partes menores, geralmente de tamanho fixo, e cada fragmento é criptografado separadamente. Isso dificulta a reconstituição dos dados originais sem ter acesso a todos os fragmentos e a chave de criptografia correta. Além disso, a fragmentação pode contribuir para uma distribuição mais uniforme dos dados criptografados, tornando mais difícil a análise e identificação de padrões pelos atacantes.

Um exemplo prático de fragmentação é a criptografia de um arquivo grande. Em vez de criptografar o arquivo como um todo, ele pode ser dividido em partes menores, como blocos de tamanho fixo. Cada bloco é criptografado separadamente e os fragmentos resultantes são transmitidos ou armazenados de forma separada. Assim, mesmo se um fragmento for interceptado por um atacante, ele terá apenas uma parte criptografada dos dados e não poderá obter acesso ao arquivo completo sem ter todos os fragmentos e a chave de criptografia adequada.

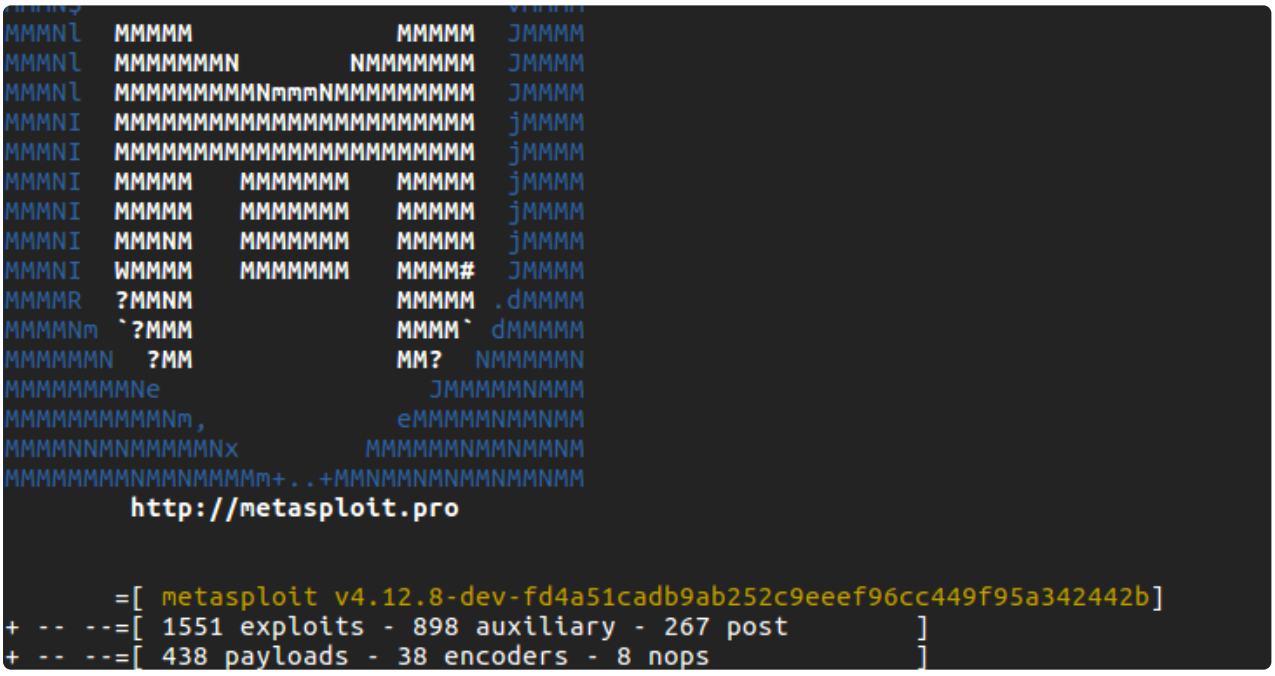
Marcação e ocultação em metadados

Refere-se a técnicas utilizadas para adicionar informações extra aos dados, conhecidas como metadados, a fim de fornecer contexto ou ocultar informações sensíveis. Os metadados são informações que descrevem outros dados, como informações sobre o autor, data de criação, localização, entre outros.

A marcação em metadados envolve adicionar informações relevantes aos dados de forma visível, que podem ser lidas e interpretadas por qualquer pessoa que tenha acesso a eles. Por exemplo, em um documento de texto, é possível adicionar metadados como o nome do autor, a versão do documento, as alterações realizadas, entre outros. Essas informações podem ser úteis para rastrear o histórico do documento e identificar a autoria.

Já a ocultação em metadados envolve a adição de informações ocultas nos dados, que não são visíveis inicialmente. Essas informações podem ser usadas para diversos fins, como marcar informações confidenciais, identificar o autor de um documento de forma anônima ou rastrear a origem de uma informação vazada. Esses metadados ocultos não são facilmente acessíveis e podem exigir ferramentas especiais para serem extraídos.

Um exemplo prático de marcação e ocultação em metadados é a adição de informações confidenciais em arquivos de imagem. Suponha que você deseje enviar uma imagem por e-mail, mas deseja ocultar informações sensíveis contidas nela. Você pode usar uma ferramenta de edição de imagens para adicionar metadados ocultos, como texto ou uma marca d'água, que não são visíveis na imagem, mas podem ser extraídos posteriormente. Dessa forma, você pode compartilhar a imagem sem revelar as informações confidenciais a olho nu, mas mantendo-as presentes para quem souber como acessá-las.



Ferramenta Metasploit.

Conclusão

A aula abordou diversos tópicos relacionados à criptografia e técnicas de ocultação de dados. Exploramos as propriedades dos algoritmos criptográficos, como confusão e difusão, que garantem a segurança dos dados por meio da

complexidade e dispersão das informações. Discutimos também a importância da criptografia em diferentes contextos, como em dados em repouso, em trânsito e em uso, para proteger a confidencialidade e integridade das informações.

Além disso, abordamos técnicas de ocultação de dados, como esteganografia, ofuscação, fragmentação e marcação e ocultação em metadados, que visam fornecer camadas adicionais de proteção e discrição. Essas técnicas permitem esconder informações sensíveis em meio a dados aparentemente comuns, dificultando sua detecção por pessoas não autorizadas.

> Parabéns pela conclusão da aula! Você dedicou tempo e esforço para absorver os conceitos e tópicos abordados, demonstrando seu comprometimento com o aprendizado. É gratificante ver seu progresso e o interesse que demonstrou durante a aula. Continue assim, buscando sempre expandir seus conhecimentos e aprimorar suas habilidades em segurança da informação. Estou aqui para ajudar no que precisar. Parabéns novamente!

Aula 2: Criptografia simétrica

Objetivos

- [ ] Compreender os princípios básicos da criptografia simétrica.
- [ ] Conhecer as cifras de fluxo e de bloco.
- [ ] Analisar a segurança da criptografia simétrica.

Conceitos

- [ ] Chave simétrica.
- [ ] Algoritmos de criptografia simétrica.
- [ ] Modos de operação de criptografia simétrica.

Introdução

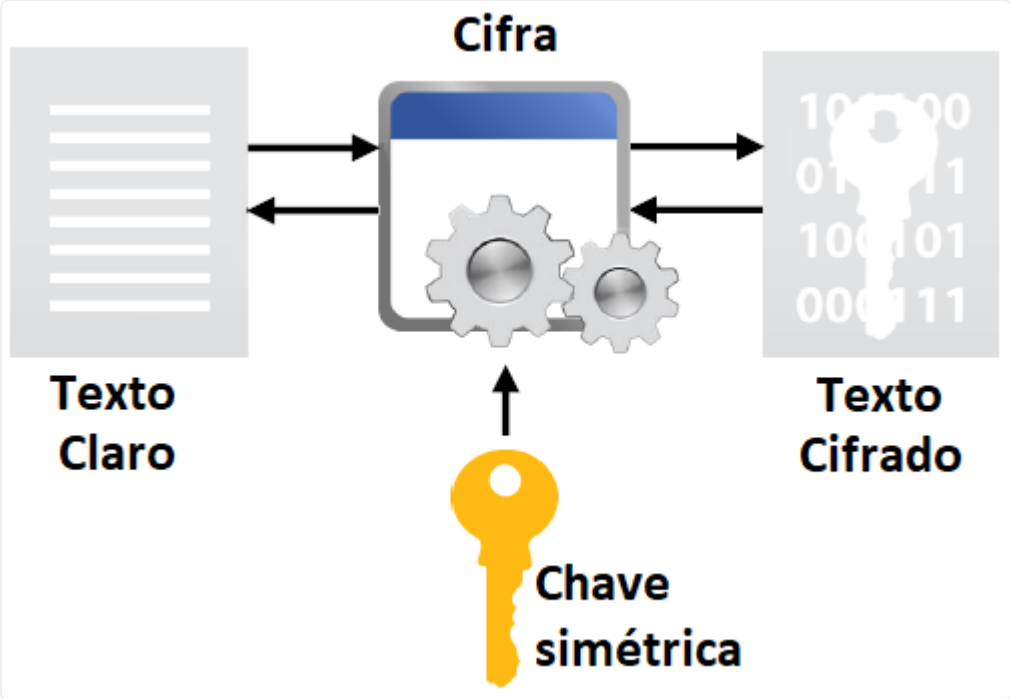
A criptografia simétrica é uma técnica poderosa para garantir a confidencialidade dos dados, usando uma única chave para criptografar e descriptografar as informações. A compreensão dos princípios básicos, algoritmos e modos de operação da criptografia simétrica é fundamental para a construção de sistemas de segurança robustos e protegidos contra ameaças cibernéticas.

Componentes e funcionamento da criptografia simétrica

A criptografia simétrica envolve três componentes principais: o texto claro (plaintext), a chave simétrica (symmetric key) e o texto cifrado (ciphertext).

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.
2. **Chave Simétrica (Symmetric Key)**: 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.
3. **Cifra (Cipher)**: algoritmo matemático que realiza a transformação dos dados em formato legível (texto claro) para uma forma ilegível (texto cifrado) e vice-versa. A cifra utiliza a chave de criptografia como parâmetro para determinar como os dados serão embaralhados ou substituídos durante o processo de criptografia. É importante mencionar que diferentes cifras podem ser utilizadas na criptografia simétrica, como o algoritmo DES, AES, RC4, entre outros.
4. **Texto Cifrado (Ciphertext)**: O texto cifrado é o resultado da aplicação do algoritmo de criptografia à 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.



Componentes da Criptografia Simétrica.

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.

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.

Veja um exemplo prático de Cifra de Fluxo!

O XOR (OU exclusivo) é um operador lógico utilizado em criptografia e outros campos da ciência da computação. Ele opera em dois bits de entrada e produz um bit de saída de acordo com as seguintes regras: * Se os dois bits de entrada forem iguais (0-0 ou 1-1), o resultado do XOR será 0. * Se os dois bits de entrada forem diferentes (0-1 ou 1-0), o resultado do XOR será 1.

Agora, vamos dar um exemplo prático da cifra de fluxo usando o operador XOR:

Suponha que temos um texto claro "10101010" (8 bits) e uma chave simétrica "11001100" (8 bits).

Passo 1: Geração do fluxo de chave: Usando a chave simétrica como base, o algoritmo de cifra de fluxo gera um fluxo de chave que será usado para combinar com o texto claro. Neste exemplo, o fluxo de chave gerado é "11001100" dado que corresponde ao tamanho do texto claro.

Passo 2: Criptografia usando XOR: Agora, aplicamos a operação XOR entre cada bit do texto claro e o bit correspondente do fluxo de chave:

- Texto claro: 10101010
-

- Fluxo de chave: 11001100
- Aplicando XOR: 01100110

O resultado é o texto cifrado resultante do XOR: "01100110".

Para descriptografar, basta aplicar novamente o XOR entre o texto cifrado e o fluxo de chave:

- Texto cifrado: 01100110
- Fluxo de chave: 11001100
- Aplicando XOR: 10101010

Obtemos o texto claro original "10101010".

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:

1. Electronic Codebook (ECB): 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.
2. 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.

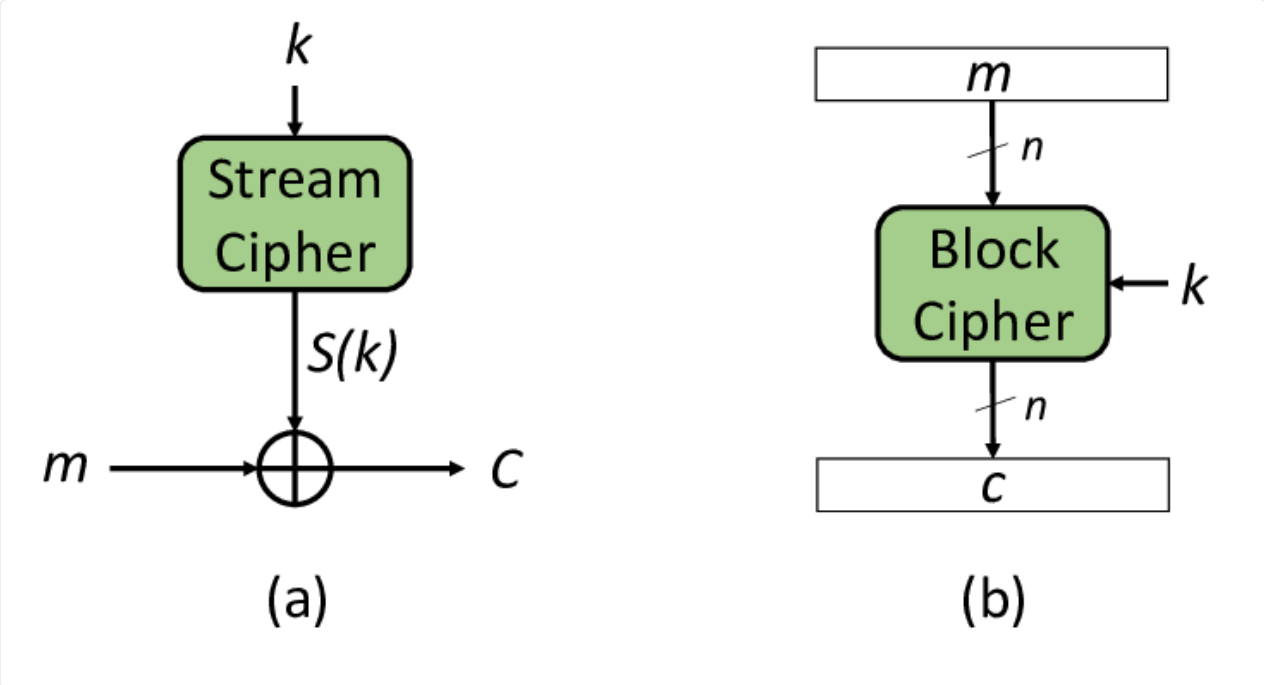
3. Cipher Feedback (CFB): No modo CFB, os blocos de dados são tratados como um fluxo de bits, em vez de blocos independentes. É utilizado um tamanho de deslocamento menor que o tamanho do bloco para criar um fluxo de bits. O fluxo de bits é então combinado com o texto claro por meio de uma operação XOR, 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.
4. 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.
5. 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.



Cifra de fluxo x Cifra de bloco.

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:

- Chave secreta compartilhada: 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.

Grande quantidade de chaves compartilhadas.

- 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.

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.

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.

Embora o DES tenha sido amplamente utilizado no passado, ele é considerado atualmente inseguro para muitas aplicações devido ao seu tamanho de chave relativamente curto.

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).

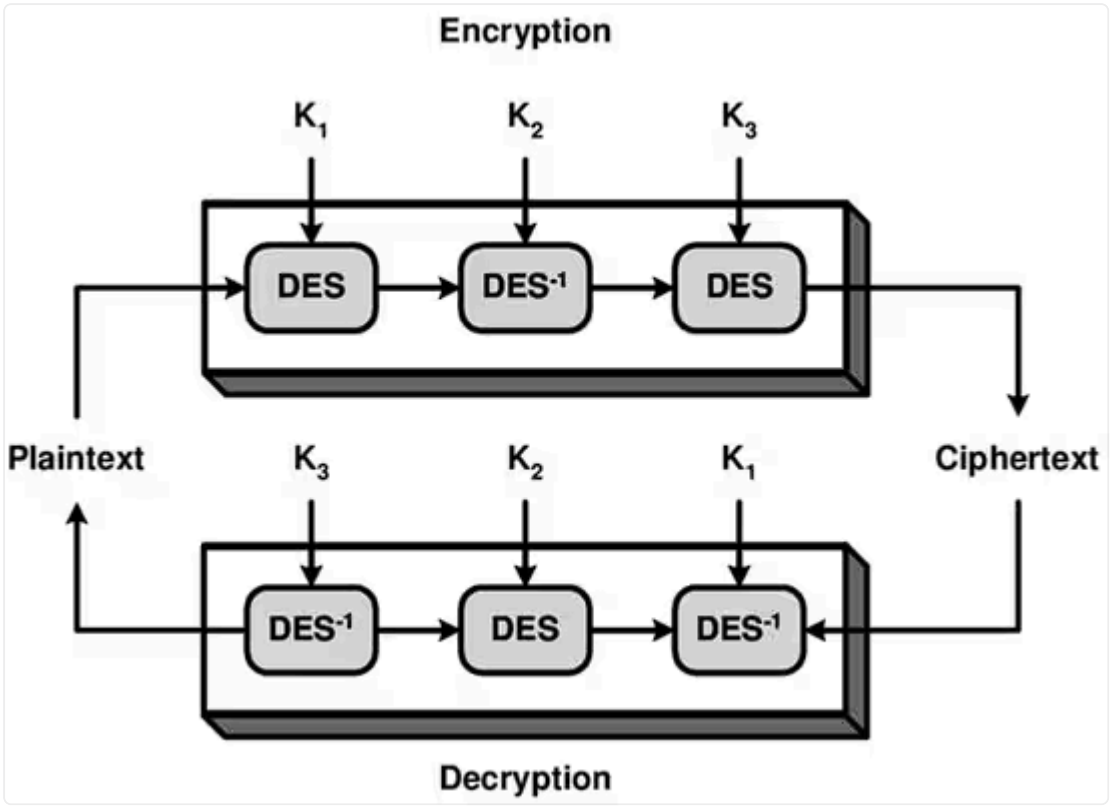
Na primeira etapa, o bloco de dados é criptografado usando uma chave de criptografia primária (K1). O algoritmo executa uma série de rounds, semelhante ao DES, que inclui permutações, substituições e combinações lineares. Isso resulta em um bloco criptografado intermediário.

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.



Algoritmo 3DES.

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.

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.

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.

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.

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.

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.

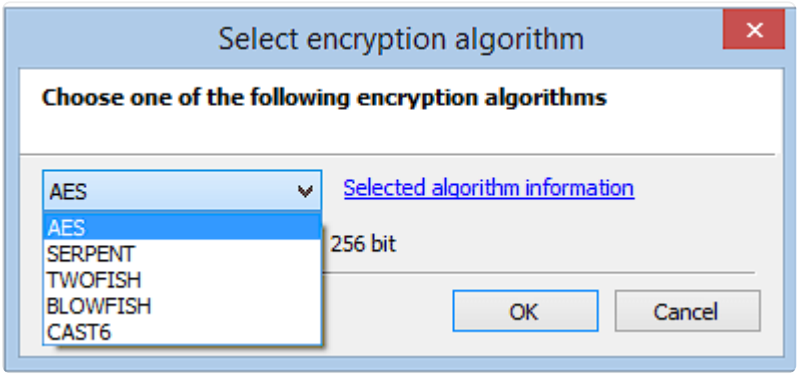
A etapa ShiftRows envolve o deslocamento dos bytes nas linhas do bloco de dados. Cada linha é deslocada para a esquerda, onde o primeiro byte permanece no lugar, o segundo é deslocado uma posição, o terceiro duas posições e o quarto três posições.

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.



Algoritmos simétricos.

Conclusão

A aula de criptografia simétrica abordou os principais conceitos e algoritmos relacionados à criptografia de chave única. Exploramos os componentes essenciais, como a chave de criptografia, o algoritmo de criptografia e o vetor de inicialização, e aprendemos sobre os diferentes modos de operação, como ECB, CBC, CFB, OFB e CTR. Também discutimos os desafios da distribuição segura de chaves e os algoritmos modernos, como RC4, Blowfish, TwoFish, Serpent, AES e RC6. Compreender os princípios e técnicas da criptografia simétrica nos permite proteger informações sensíveis, garantir a confidencialidade dos dados e mitigar riscos de segurança em diversas aplicações. A criptografia simétrica continua sendo uma base sólida para a segurança da informação, proporcionando uma camada de proteção essencial em nossos sistemas e comunicações digitais.

> Parabéns pelo seu empenho e por ter concluído a aula de criptografia simétrica! Você demonstrou dedicação em aprender sobre os conceitos e algoritmos fundamentais nessa área tão importante da segurança da informação. Agora você possui um conhecimento sólido sobre os componentes, os modos de operação e os algoritmos modernos da criptografia simétrica. Continue

> explorando e aprimorando seus conhecimentos na área da segurança cibernética!