Segurança da Tecnologia da Informação

UNIDADE 05

Criptografia

Nesta Unidade, iremos estudar as funções de hash criptográficas. Conhecer os princípios e as propriedades fundamentais para implementação das funções hash criptográficas. Ainda, iremos nos aprofundar nos principais algoritmos de função hash criptográficas, tais como o SHA, SHA-1, SHA-2, MD5 entre outros. Adicionalmente, apresentar as funções hash baseadas em Cipher Block Chaining, verificar como que ela é estruturada, quais os benefícios e ameaças associados a esta abordagem. Também, iremos demostrar algumas das aplicações das funções hash criptográficas, em destaque o uso de autenticação de mensagens e assinaturas digitais. Apresentaremos, os requisitos de segurança para implementação de função hash criptográficas. Abordaremos o impacto das colisões em funções hash criptográficas, como que o adversário pode explorar esta e outras fraquezas. Por fim, iremos

expor os algoritmos de função hash estabelecidos pelos padrões internacionais, ou seja, mostraremos algumas estratégias e boas práticas para implementação das funções hash criptográficas.

Funções Hash Criptográficas

Uma função *hash* mapeia uma mensagem de entrada de tamanho variável em um bloco de dados de tamanho fixo, denominado de código *hash.* O *Hash* também é conhecido como *checksum.* O termo *checksum* é bastante utilizado para se referir ao código usado para verificar a integridade de uma mensagem transmitida ou que foi armazenada em algum meio por determinado tempo.

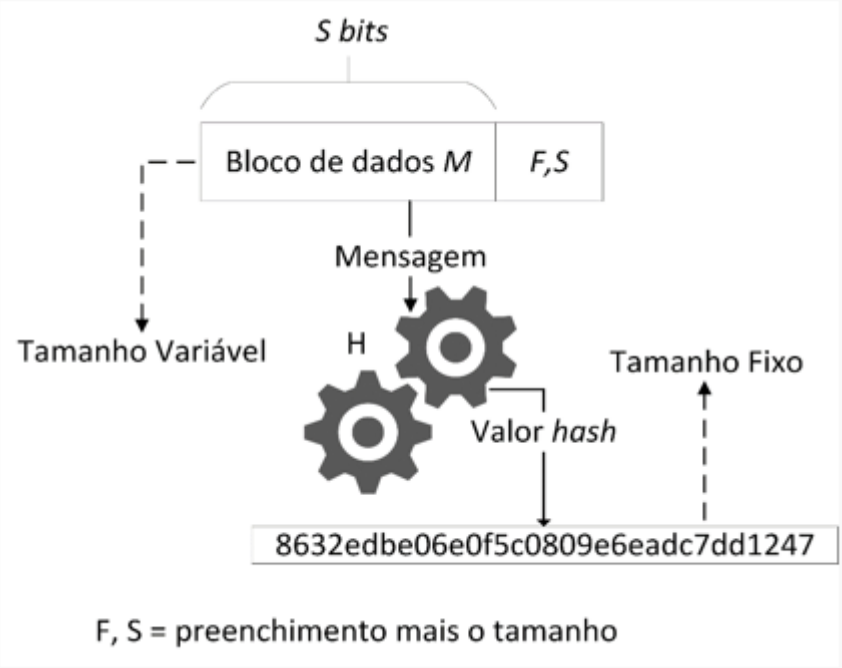
Diferente dos algoritmos de criptografia convencional, o algoritmo de uma função *hash* não é reversível, ou seja, você não consegue retornar ao valor original. As funções *hashes* fornecem a garantia de que a mudança de qualquer bit na mensagem produzirá um código *hash* diferente.

A função *hash* destinada às aplicações de segurança é denominada como função *hash* criptográfica. O algoritmo de uma função *hash* criptográfica estabelece dois princípios:

- A propriedade de mão única.
- A propriedade livre de colisão.

A propriedade de mão única garante que a partir do código *hash* você não consegue retornar ao seu valor inicial, sendo computacionalmente inviável. Por sua vez, a propriedade livre de colisão garante que dois objetos de dados diferentes não serão mapeados com o mesmo resultado *hash.* Dado essas duas propriedades, as funções de *hash* criptográficas são amplamente utilizadas para determinar se os dados foram alterados ou não.

A função *hash* disposta na fórmula h=H(M), onde a variável M corresponde ao valor de entrada para a função H, tendo como retorno h como sendo o valor do *hash* que possui um tamanho fixo. Esta função é representada pela figura a seguir, a entrada da função *hash* é preenchida por um número inteiro, em geral múltiplo de 1024bits, incluindo também como entrada o tamanho em bits da entrada original. Tal estratégia permite dificultar que um adversário consiga reproduzir uma mensagem com o mesmo valor de *hash.*



Fonte: Autor

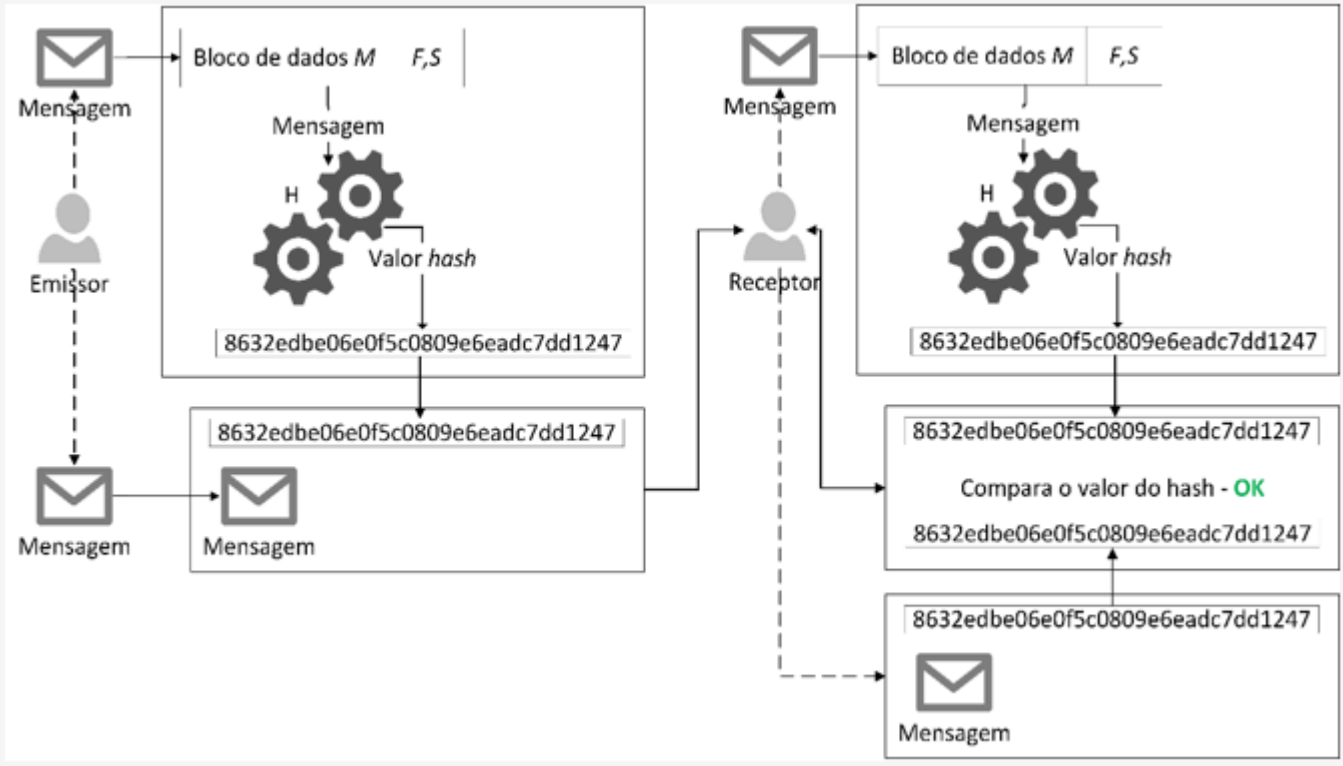
Aplicações de Funções Hash Criptográficas

A função *hash* criptográfica é utilizada em diversas aplicações de segurança, sendo também utilizada em alguns dos protocolos da internet. Nesta seção veremos algumas das principais aplicações das funções *hash* criptográficas no âmbito de segurança da informação.

Autenticação de Mensagem

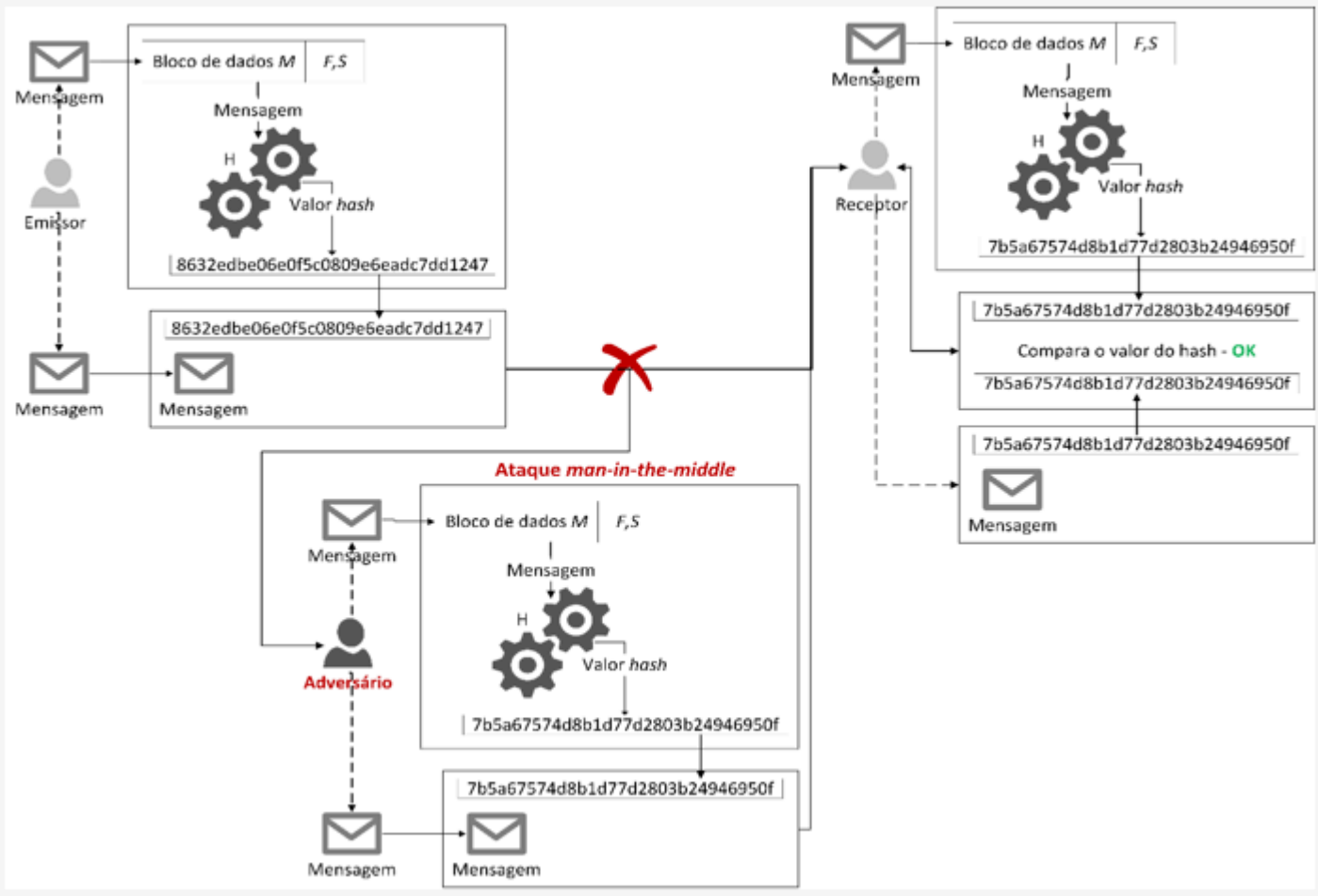
As funções hash criptográfica podem ser utilizadas para implementar os serviços de autenticação de mensagens. O mecanismo de autenticação de mensagens permite verificar a integridade de determinada mensagem. Este mecanismo é utilizado para garantir que uma mensagem transmitida não foi modificada, ou seja, não houve inserção ou exclusão de algum conteúdo da mensagem. Adicionalmente, o mecanismo de autenticação pode ser utilizado para fornecer garantia que a identidade de um emissor é válida. Ao utilizar uma função hash criptográfica para realizar autenticação de mensagem o valor decorrente deste processo é denominado resumo de mensagem.

A função hash para autenticação de mensagens tem por base os seguintes passos. Primeiramente, o emissor utiliza a função hash para calcular o valor de hash dos bits da mensagem, então transmite a mensagem em conjunto com o valor do hash. Posteriormente, o receptor realiza o mesmo cálculo, utiliza a função hash sobre os bits da mensagem, então compara o hash obtido como o hash que foi enviado pelo emissor da mensagem. Por conseguinte, caso os hashes sejam diferentes, o receptor consegue identificar se a mensagem foi adulterada. Observe os passos de autenticação de mensagens apresentados na figura a seguir.



Fonte: Autor

O valor *hash* necessita ser transmitido de uma forma segura. É fundamental que o código hash seja protegido, pois se um adversário tentar modificar a mensagem, não terá como alterar o valor do *hash* para tentar enganar o receptor. Deste modo, evita-se um ataque de *man-in-the-middle*, conforme apresentado na figura a seguir.



Fonte: Autor

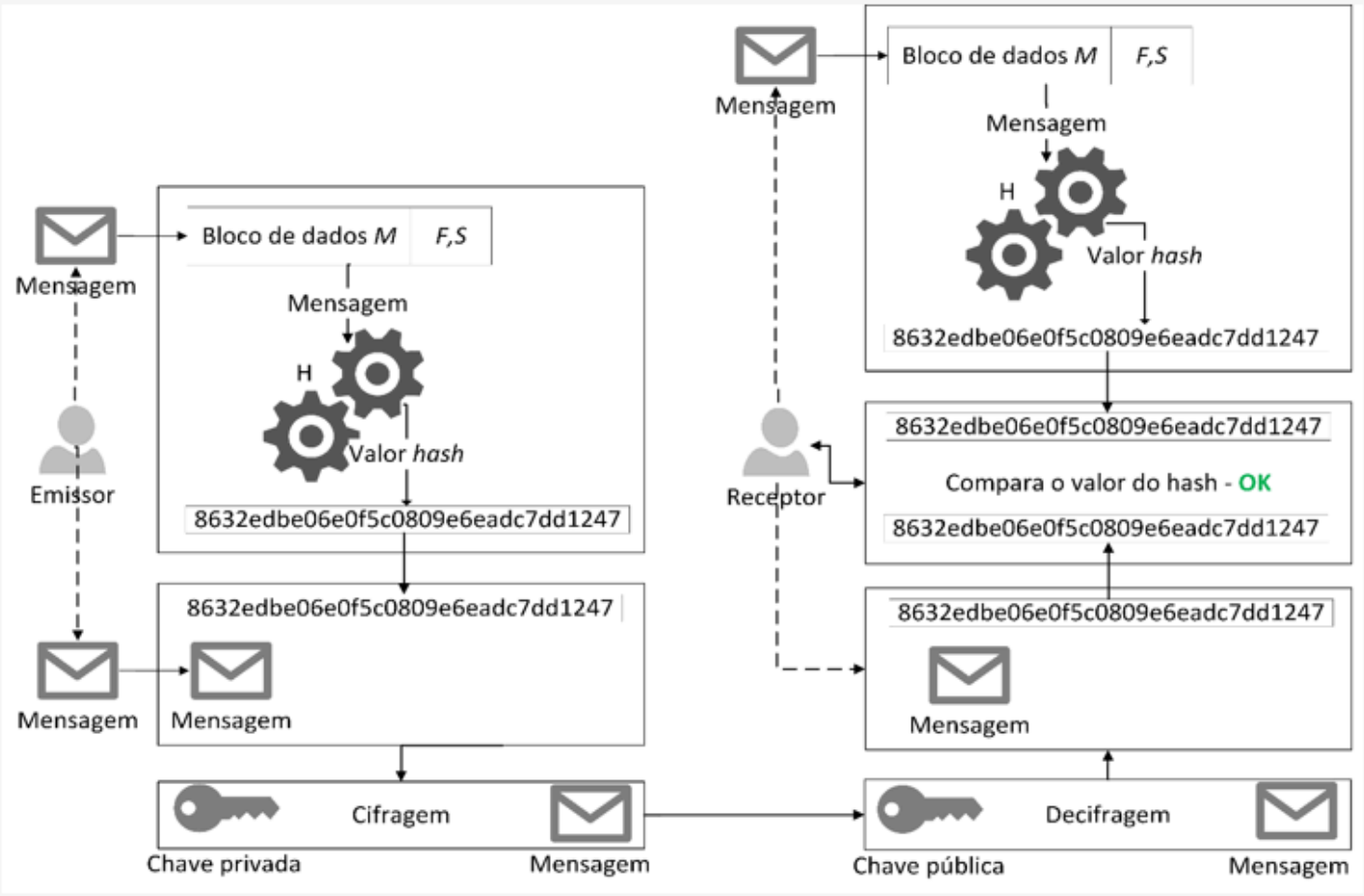
Neste tipo de ataque, o adversário se coloca entre o emissor e o receptor da mensagem. Quando o emissor transmite uma mensagem e adiciona um valor de *hash.* O adversário intercepta a mensagem, então modifica a mensagem e adiciona um novo valor de *hash* gerado a partir da mensagem que foi modificada. Neste sentido, quando o receptor receber a mensagem, irá checá-la, porém não conseguirá perceber se a mensagem foi modificada. Deste modo, para impedir este tipo de ataque, o código hash gerado pelo emissor deve ser protegido.

A autenticação de mensagem pode ser obtida utilizando um código de autenticação de mensagens, também conhecido como função de *hash* chaveada. A função de *hash* chaveada é normalmente usada entre duas partes (emissor e receptor) que compartilham uma chave secreta para trocar informações. Uma função de *hash* chaveada recebe como entrada uma chave secreta e uma mensagem e produz um código *hash,* conhecido como MAC (*Message Authentication Code*). O MAC é associado a mensagem a ser protegida. Caso seja necessário verificar a integridade da mensagem, a função chaveada pode ser aplicada à mensagem e o resultado é comparado com o valor MAC que foi associado. Desta forma, se um adversário alterar a mensagem não conseguirá alterar o valor do MAC que foi associado sem conhecer a chave secreta compartilhada entre as partes. Ainda, ressalta-se que parte do receptor que está verificando a mensagem tem certeza do emissor da mensagem, visto que apenas o emissor conhece a chave secreta.

Assinaturas Digitais

Uma outra aplicação de função *hash* criptográfica bastante importante é a assinatura digital. O funcionamento de operação da assinatura digital é bastante similar ao funcionamento da função *hash* chaveada. Retratando do mesmo processo da assinatura digital, o valor *hash* da mensagem é cifrado utilizando a chave privada de um determinado usuário. Neste sentido, qualquer indivíduo que tenha posse da chave pública do usuário poderá verificar a integridade da mensagem na qual está associada a assinatura digital. Assim, um adversário somente conseguirá modificar a mensagem se ele tiver posse da chave privada do usuário.

Na figura a seguir é apresentado de maneira simplificada o processo de assinatura digital utilizando o código *hash.* Observe que o código *hash* é cifrado utilizando a chave privada do emissor, decifrado com a chave pública. Na próxima unidade nos aprofundaremos nos conceitos de criptografia de chave pública e privada. Neste momento, é suficiente compreender que a criptografia de chaves pública e privada utiliza duas chaves distintas, uma para codificar e outra para decodificar mensagens.



Fonte: Autor

Outras aplicações

Entre outras aplicações, as funções *hash* criptográficas são utilizadas para criar os arquivos de senha de mão única conforme visto na Unidade 3 – Mecanismos de Autenticação, à medida que nos aprofundamos no mecanismo de autenticação, descobrimos que o sistema operacional ao invés de armazenar diretamente o valor da senha, armazena o *hash* correspondente a senha. Logo, o adversário que conseguir acesso ao arquivo de senha não terá acesso efetivo a senha real, dado que a partir do valor do *hash* ele não conseguirá retornar ao valor original da senha. De maneira simplificada, quando um usuário solicitado acesso ao sistema e fornece sua senha, o mecanismo de autenticação imediatamente aplica a função *hash* criptográfica sobre a senha digitada pelo usuário, então é realizado o processo de verificação, onde o *hash* obtido é comparado com valor do *hash* que está armazenado no arquivo de senha. Destaca-se que este procedimento é realizado na grande maioria dos sistemas operacionais.

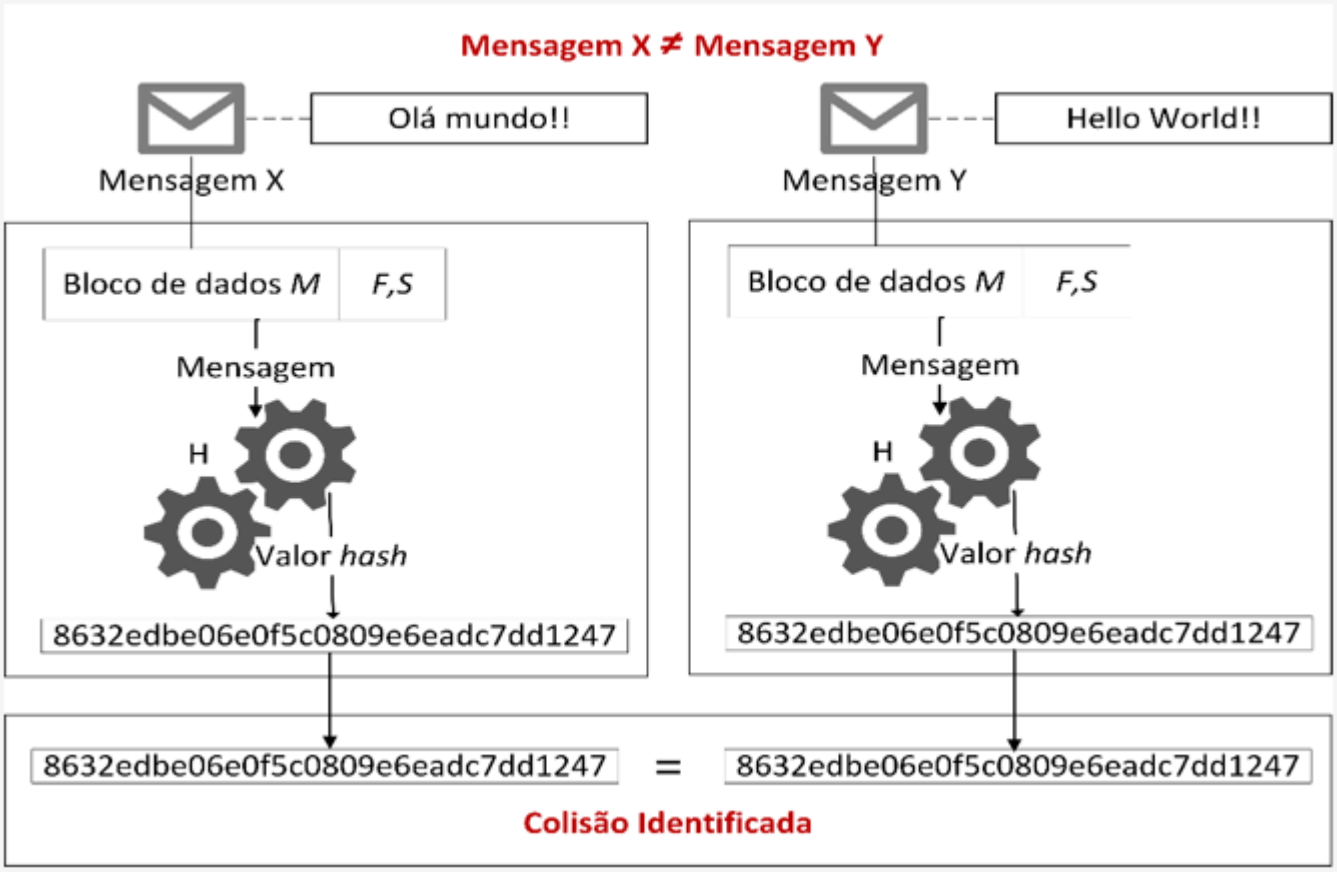
Uma outra aplicação bastante interessante é a utilização da função *hash* criptográfica para detecção de intrusão e detecção de vírus. Neste caso, a função *hash* criptográfica é aplicada sobre cada arquivo do sistema, gerando um valor *hash* para cada arquivo, então é armazenado uma cópia de segurança dos valores de *hash* - de preferência em uma mídia externa. Posteriormente,

você poderá utilizar os valores de *hash* armazenados para determinar se um arquivo foi modificado no sistema, ou seja, aplica-se a função *hash* criptográfica sobre o arquivo que se deseja analisar, então compara-se esse *hash* com o valor armazenado anteriormente.

Ainda, outra aplicação para a função *hash* criptográfica é utilizá-la para construir um gerador de número pseudoaleatório, cujo uso é bastante comum em aplicações que geram as chaves simétricas.

Colisões em Funções Hash Criptográficas

Dado o valor de *hash* h = H(x), definimos x como sendo a pré-imagem de h. Em outras palavras, significa que x é um bloco de dados cuja função *hash* H obtém o *hash* h. Como a função H é mapeada em uma cardinalidade de muitos para um, para qualquer valor existente de h, existirá várias pré-imagens. Neste sentido, uma colisão ocorre quando temos a situação que x ≠ y e H(x) = H(y). Esta notação denota o conceito de colisão, onde ao fornecer duas entradas diferentes para função *hash* é obtido dois códigos de *hash* idênticos. Tratando-se de função *hash* criptográfica a colisão é algo extremamente indesejado, principalmente quando precisamos garantir a integridade dos dados.



Fonte: Autor

A fim de tentar identificar potenciais colisões para um valor de *hash,* devemos considerar a quantidade de pré-imagens existentes para este valor. Para tal, definimos um tamanho de código *hash* de n *bits* e utilizamos a função *hash* H que recebe como entrada uma mensagem contendo um tamanho de b *bits,* sendo $b > n$. Assim, podemos calcular o total de mensagens possíveis, como sendo 2^b, consequentemente teremos um total de 2^n valores de *hash* possíveis. Deste modo, para cada valor de *hash* teremos 2^{b-n} pré-imagens. Então, se a função *hash* H distribuir de maneira uniforme os valores de *hash,* cada valor de *hash* possuirá um valor aproximado de 2^{b-n} pré-imagens. Ainda, devemos destacar que caso seja autorizado entradas de qualquer tamanho, não limitando o tamanho da entrada com um comprimento fixo, teremos uma grande variação no número de pré-imagens por valor de *hash.*

Contudo, exploramos os riscos de segurança sempre no pior caso. Para compreender com uma melhor clareza o impacto das colisões em termos de segurança das funções *hash* criptográficas, devemos definir com maior precisão os seus requisitos de segurança.

Requisitos de Segurança das Funções Hash Criptográficas

Existem ao todo basicamente sete requisitos que devem ser observados na implementação de funções Hash criptográficas, seguem dispostos abaixo:

1. Tamanho de entrada variável
2. Tamanho de saída fixo
3. Eficiência
4. Propriedade de mão única
5. Resistência à colisão fraca
6. Resistência à colisão forte
7. Pseudoaleatoriedade

Em geral, tais requisitos são amplamente aceitos para implementar uma função *hash* criptográfica. Destaca-se que as três primeiras propriedades são utilizadas em aplicações práticas de função *hash.* O **tamanho de entrada variável** é usado para definir um bloco de dados de qualquer tamanho para função *hash* H. Por sua vez, o **tamanho de saída fixo** determina uma saída com um tamanho de código *hash* fixo, indiferentemente do tamanho da

mensagem de entrada. Adicionalmente, utilizamos a propriedade da **eficiência** para calcular o valor do *hash* a partir da função *hash* H(x), obtendo facilmente o valor de *hash* para qualquer que seja o valor de x, aplicado tanto para implementação de *hardware* como por *software.*

A **propriedade de mão única** é definida como resistência à pré-imagem. Esta propriedade estabelece que para qualquer valor de *hash* h que seja informado, é computacionalmente impossível encontrar y, tal que H(y) = h. Em outras palavras, é fácil gerar o código *hash* a partir da mensagem, porém é praticamente impossível gerar a mensagem a partir do código *hash.* Esta é uma das principais propriedades da função *hash* criptográfica, técnica bastante utilizada associada ao mecanismo de autenticação que adota o uso de valor secreto. Reforçando que neste caso o valor secreto não é enviado. Contudo, considerando uma função *hash* que não possua a propriedade de mão única, um adversário poderia explorar facilmente o valor secreto.

A **resistência à colisão fraca** é definida como resistência à segunda pré-imagem. Esta propriedade garante que é impossível obter uma mensagem alternativa que possua o mesmo valor de *hash* de uma determinada mensagem. A propriedade de resistência à colisão fraca estabelece a seguinte premissa, para qualquer bloco de dados x informado é computacionalmente impossível encontrar um valor de $y \neq x$ que atenda a sentença H(y) = H(x). Tal propriedade protege contra a falsificação de uma *hash* cifrado. Considerando um cenário onde esta propriedade não exista, um adversário poderia interceptar uma mensagem com o código *hash* cifrado, então gerar um código de *hash* decifrado de uma mensagem, por fim, gerar uma mensagem diferente reaproveitando o mesmo código *hash* para validar a mensagem modificada.

As funções *hash* que atendem as cinco primeiras propriedades mencionadas anteriormente são denominadas como função *hash* fraca. Porém, a função *hash* pode possuir uma sexta propriedade, denominada de **resistência à colisão forte**, tal propriedade confere proteção contra um ataque onde um adversário gera uma mensagem para ser assinada por outra parte. Esta propriedade estabelece a premissa que é computacionalmente impossível encontrar qualquer par (x, y), tal que H(x) = H(y). Consequentemente, a função que implementa esta propriedade é chamada de função *hash* forte.

Adicionalmente, existe um último requisito denominado como a propriedade da **pseudoaleatoriedade**. Apesar desta propriedade não ser amplamente citada, este requisito da função *hash* criptográfica está relativamente implícita. As funções *hash* criptográficas são normalmente utilizadas para auxiliar no processo de derivação de chaves e ainda geração de número pseudoaleatório. Além disso, nas aplicações que visam integridade de mensagens, as três propriedades de resistência que consistem na saída da função *hash* aparenta ser aleatório. Neste sentido, é possível afirmar que a função *hash* produz uma saída pseudoaleatória.

Message Digest Algorithm 5

O MD5 (Message Digest Algorithm 5) é uma função hash amplamente utilizada que produz um valor de hash de 128 bits declarado em uma string de 32 caracteres. O MD5 foi desenvolvido por Ronald Rivest da RSA Data Security, Inc. em 1991. Esta função hash foi desenvolvida para substituir a função hash anterior MD4 que apresentava algumas vulnerabilidades de segurança. Então, em 1992 o MD5 foi especificado como padrão na RFC 1321.

Destaca-se que o MD5 é um algoritmo unidirecional (algoritmo de mão única), a partir do valor hash gerado no MD5 não é possível retornar ao valor da mensagem. A partir de uma mensagem de tamanho variável (entrada de qualquer tamanho), o MD5 produz um valor hash fixo, correspondente a 128 bits.

Observa-se que como qualquer função hash criptográfico um dos requisitos fundamentais é garantir que não exista colisões entre os hash (conforme estudando anteriormente, encontrar duas mensagens distintas com mesmo hash). Considerando uma das principais fraquezas do MD5, as pseudocolisões foram exploradas no algoritmo MD5. Em destaque, o malware denominado Flame em 2012 explorou as vulnerabilidades do MD5 para falsificar assinaturas digitais da Microsoft.

Este algoritmo é de domínio público, pode ser utilizado para quaisquer fins. Apesar do MD5 ter sido projetado para ser utilizado como função hash criptográfica, foram identificadas algumas vulnerabilidades de segurança no algoritmo. Ainda assim, o MD5 é útil para ser utilizado para fins não criptográficos, como determinar uma partição para uma chave específica em um banco de dados particionado. Também pode ser utilizado para checar a integridade dos dados contra corrupção não intencional. Além, de ser utilizado como mecanismo de integridade em vários protocolos de padrão Internet.

Após a exposição das fraquezas do MD5, Roland Rivest em conjunto com outros pesquisadores publicaram em 2008 uma nova versão do algoritmo, com hash de tamanhos de 224, 256, 384 ou 512 bits. O novo algoritmo foi denominado como MD5. Este algoritmo foi cogitado para ser utilizado como o novo algoritmo SHA-3, porém acabou não sendo aprovado por ser considerado um algoritmo muito lento para os computadores da época. Apesar da fraqueza e depreciação dos pesquisadores da área de segurança, o MD5 ainda continua sendo amplamente utilizado.

Funções Hash Baseadas em Cipher Block Chaining

Na literatura foram apresentadas diversas propostas a fim de aperfeiçoar as funções hash criptográficas, uma das estratégias foi o uso da técnica de Cipher Block Chaining, não adotando uma chave secreta. Entre as primeiras propostas destaca-se a proposta apresentada por Rabin [Rabin, 1978]. Esta proposta consiste em dividir a mensagem (M) em blocos de tamanho fixo (M_1 , M_2 , ... M_n). Posteriormente, utilizar um sistema de cifragem simétrica para calcular o código hash G, utiliza-se a seguinte notação:

Ho=valor inicial

Hi=E(Mi,Hi−1)

G=Hn

Semelhante a qualquer código de hash, esta abordagem também está sujeita ao ataque de dia do aniversário, observando que caso seja utilizado um algoritmo de cifragem DES com um código hash com somente 64 bits, o sistema será vulnerável. O ataque do aniversário consiste em um tipo de ataque criptográfico que explora o cálculo matemático por trás do paradoxo do aniversário na teoria da probabilidade.

O paradoxo do aniversário afirma que dado um grupo de 23 sujeitos selecionados aleatoriamente, a chance de dois desses sujeitos terem a mesma data de aniversário é de mais de 50%. Ainda, estendendo este paradoxo, dado 57 ou mais sujeitos, a probabilidade chega a ser maior do que 99%, contudo, não pode ser considerado exatamente 100%, exceto que haja no mínimo 367 pessoas. Este tipo de ataque pode ser utilizado para explorar a comunicação entre duas ou mais partes. Onde o ataque depende da maior probabilidade de colisão localizadas entre as tentativas de ataque aleatório e a quantidade fixa de permutações.

Um ataque derivado do ataque de dia do aniversário pode ser aplicado mesmo que o adversário tenha acesso somente a uma das mensagens e a assinatura válida, e não tenha acesso para obter outras assinaturas. Então, vamos explorar um pouco este cenário, considerando que o adversário consiga interceptar uma mensagem com uma assinatura no formato de um código hash cifrado e que o código de hash não cifrado tenha uma quantidade exata de m bits em sua extensão [Stallings,2008], deve se considerar:

1. Utilizar o algoritmo citado anteriormente (código de hash G) para realizar o cálculo do código de hash não cifrado G.

2. Construir qualquer mensagem desejada norteado pelo formato

   Q1, Q2,…Qn−2

3. Calcular $H_i = E(Q_1, H_{i-1})$, para $1 \leq i \leq (n-2)$.

4. Gerar $2^{m/2}$ blocos aleatórios; para cada bloco X, calcular $E(X, H_{n-2})$. Gerar $2^{m/2}$ blocos aleatórios adicionais; para cada bloco Y, calcular

   D(Y,G) , onde D é a função de cifragem que corresponde a E.

5. Baseado no paradoxo do dia do aniversário, considerando alta probabilidade, haverá um X e Y tal que blocos aleatórios adicionais; para cada bloco Y, calcular

   $E(X, H_{n-2}) - D(Y, G)$.

6. Formando a mensagem Q 1, Q2,…Qn−2, X,Y . Conferir que esta mensagem tem o código de *hash* G. Consequentemente, poderá ser utilizada com a assinatura cifrada que foi interceptada.

Este tipo de ataque é conhecido como ataque de *meet-in-the-middle*. Vários autores na literatura propuseram otimizações para fortalecer a técnica tradicional de *Cipher Block Chaining*. Em contrapartida, os adversários também exploraram as novas técnicas a fim de encontrar vulnerabilidades. Algumas técnicas de *Cipher Block Chaining* são vulneráveis a uma série de tipos de ataques. Em geral, alguma variante do ataque de dia do aniversário terá sucesso contra uma técnica que utilize *Cipher Block Chaining* sem o uso de chave secreta, no caso de utilizar um código de *hash* de 64 bits ou menos, ou ainda utilizar um código maior que possa ser decomposto em unidades independentes de código [JUEM,1987]. Deste modo, é importante ter uma certa atenção em relação a outras propostas para o *hashing,* sendo que muitas delas também possuem seus prontos fracos [MITC, 1992].

Secure Hash Algorithm (SHA)

Atualmente, a função *hash* criptográfica mais amplamente utilizada tem sido o *Secure Hash Algorithm* (SHA). Este algoritmo foi desenvolvido pelo NIST *(National Institute of Standards and Technology*) e publicado como um padrão de processamento de informações federais (FIPS 180) em 1993. Este algoritmo foi introduzido visando substituir as funções *hashes* anteriores que apresentaram vulnerabilidades de criptoanálise críticas. Destaca-se que em 2005 era um dos últimos algoritmos padronizado, restante, que ainda não havia sido encontrado vulnerabilidades. Então, assim que foram encontrados alguns pontos fracos no SHA,

uma outra versão foi lançada como padrão no FIPS 180-1 em 1995. Esta nova versão do SHA passou a ser chamada de SHA1, diante disto, a versão anterior ficou conhecida como SHA-0. Ressalta-se que o documento de padrões oficial é intitulado como *Secure Hash Standard*, ainda que a função hash do SHA seja baseada na função hash do MD4, o que confere uma modelagem do projeto bem próxima.

A função *hash* que implementa o SHA-1 produz uma saída de hash de 160 bits. Posteriormente, o NIST concebeu outras três novas versões do SHA (estabelecidas no FIPS 180-2), com os tamanhos de valor de *hash* de 256, 384 e 512 bits, respectivamente denominadas como SHA-256, SHA-384 e SHA-512. O conjunto desses três algoritmos de *hash* são também conhecidos como SHA-2. As novas versões utilizam a mesma estrutura básica do SHA-1, adotam os mesmos tipos de aritmética modular e realizam as mesmas operações lógicas binárias. Algum tempo depois, uma nova versão de 224 bits foi incluída no documento do FIP PUB 180-3. Ainda, é importante enfatizar que tanto o SHA-1 como o SHA-2 são especificados na RFC 6234, apesar de compor o mesmo conteúdo definido no material do FIPS 180-3, adicionalmente é incluído implementação do código C.

Na medida que um algoritmo de *hash* apresenta algum ponto fraco, deve ser imediatamente descontinuado. Neste sentido, novos algoritmos mais robustos são utilizados. Nesta linha de raciocínio, em 2005, o NIST informou a intenção de descontinuar a versão do SHA-1 e migrar para o SHA-2, fornecendo um prazo para transição ocorrer até 2010. Porém, por curiosidade uma equipe de pesquisa na área de segurança descreveu um ataque onde duas mensagens distintas poderiam gerar o mesmo valor *hash* no SHA-1 utilizando apenas 269 operações. Um número relativamente menor de operações que havia sido determinado que seria necessário para encontrar uma colisão em um *hash* SHA-1 [Wang,2005]. Esse resultado foi um dos grandes motivadores para acelerar o processo da transição do SHA-1 para o SHA-2. Na tabela abaixo é apresentado uma comparação entre os parâmetros do algoritmo SHA, exposto no FIPS 180-4, observe que o algoritmo SHA-512 possui outras duas versões que modificam o tamanho do resumo da mensagem, o SHA-512/224 e SHA-512/256.

Algoritmo	Tamanho da mensagem	Tamanho do bloco	Tamanho da palavra	Tamanho do resumo da mensagem
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224

SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Finalmente, destacamos que atualmente o FIPS 180-4 corresponde a última versão do documento disponibilizado pelo NIST, ou seja, a versão que está vigente.

Funções Hash Criptográficas No Banco de Tóquio

Analisando o estudo de caso do banco do Tóquio que foi apresentado na Unidade 1 – Introdução a Segurança da Informação, podemos trazer o incidente de segurança para ser avaliado sob a perspectiva do que estudamos aqui. Temos por objetivo que você reflita em alguns aspectos relacionados ao uso de funções *hash* criptográficas no banco. Primeiramente, o banco utiliza uma variedade de sistemas internos, em alguns desses sistemas a autenticação era realizado utilizando arquivos de senha de mão única. Devido serem sistemas legados existia uma certa resistência

para realizar modificações nos sistemas. Posterior ao incidente de segurança, foi realizado um inventário minucioso em cada um dos sistemas e foi identificado que grande parte desses sistemas ainda utilizavam o algoritmo de *hash* MD5.

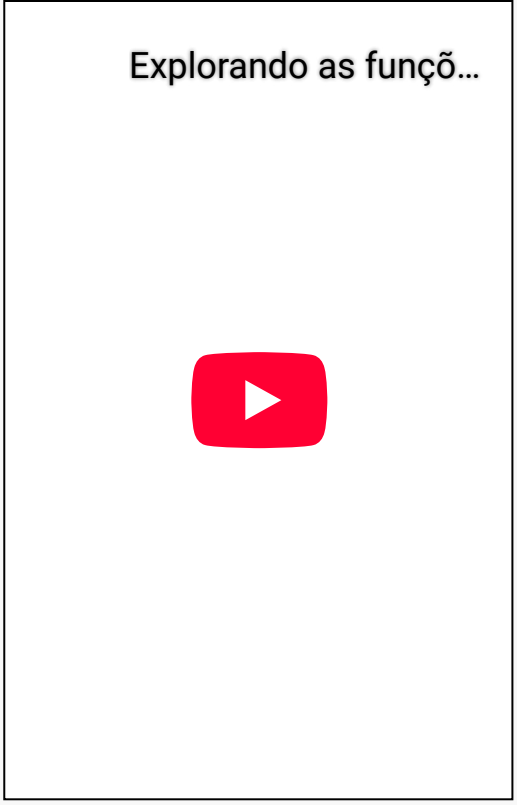
Nesta unidade, apresentamos algumas fraquezas das funções *hash* criptográficas, mas em destaque o algoritmo de hash MD5 que foi utilizado inclusive para falsificar assinaturas digitais da Microsoft. Ainda, está sendo investigado se o *malware* que foi disseminado no banco não acabou explorando este tipo de vulnerabilidade. Apesar não ter sido confirmado que a fraqueza do MD5 contribuiu com *malware* para realizar o ataque nos sistemas do banco, já estão sendo tomadas medidas para adotar um algoritmo que implementa funções *hash* criptográficas mais seguras. Estão sendo realizados testes de desempenho nos algoritmos *hash* mais atuais, está sendo estudado para que seja adicionado a versão do SHA-512.

As funções *hash* criptográficas são também utilizadas em outras aplicações no banco de Tóquio. A comunicação oficial entre os colaboradores do banco é realizada por meio de um sistema interno do banco e esta comunicação trafega em aberto. Deste modo, qualquer indivíduo mal-intencionado que capture os pacotes que trafegam na rede poderá ter acesso a informações sigilosas. Neste sentido, tais sistemas estão sendo modificados para implementar serviços de autenticação de mensagens. Cada colaborador será responsável em manter sua chave privada para realizar as trocas de mensagens no sistema, a criptografia de chave pública e privada serão alvo da nossa próxima unidade. Porém, para garantir a integridade da mensagem esta mensagem será cifrada utilizando uma função *hash.* Adicionalmente, está sendo verificado a possibilidade de adicionar a assinatura digital em alguns dos processos do banco.

As novas aplicações que utilizam as funções *hash* criptográficas propõem melhorias nos processos do banco, porém, devemos considerar a existência de vulnerabilidades em certos algoritmos de *hash.* No desenvolvimento deste processo, precisamos ficar atentos para que sejam utilizados algoritmos de *hash* adequados ao cenário de segurança do banco. Neste momento, o banco de Tóquio precisa implementar as novas aplicações, nossa missão é garantir que as propriedades de segurança da informação do banco sejam preservadas. Assim, devemos selecionar as funções *hash* criptográficas apropriadas, a fim de assegurar a propriedade de mão única e livre de colisão. Nas próximas unidades vamos conhecer outros mecanismos para garantir a segurança dos ativos de informação do banco, conhecer e saber como aplicá-los. Mas não se preocupe, vamos nortear você nesta caminhada, ao longo das próximas unidades vamos te preparar para lidar com este e outros problemas de segurança da informação no âmbito real.

Explorando as funções *hash* criptográficas

Este vídeo fornecerá uma visão mais ampla das funções *hash* criptográficas, vamos apresentar os elementos fundamentais associados a este algoritmo, quais as vantagens e desvantagens de utilizar esta abordagem, dicas e boas práticas para implementação deste mecanismo de segurança.



EXERCÍCIO

Exercícios de Fixação

1. Quais os dois princípios básicos de uma função hash criptográfica?
2. O que é o checksum e para que é utilizado?
3. Explique o conceito da propriedade mão única estabelecidas em funções hash.
4. Cite algumas aplicações de uso das funções hash criptográficas, descreva como são utilizadas.
5. O que é uma colisão em função hash criptográfica?
6. Cite os requisitos que devem ser observados para implementação de função hash criptográfica.
7. Descreva qual a principal diferença entre resistência à colisão fraca e a resistência à colisão forte.
8. Porque o algoritmo MD5 é considerado uma função hash criptográfica vulnerável?
9. Porque o algoritmo MD6 não foi adotado como padrão SHA-3?
10. Explique como o paradoxo do aniversário pode ser utilizado para efetuar ataques em funções hash criptográficas.
11. Qual a diferença entre o algoritmo SHA-0 e SHA-1?
12. Quais os algoritmos que compõe o SHA-2?

Conclusão

Esta unidade abordou as funções hash criptográficas. Descobrimos que a função hash mapeia uma mensagem de qualquer tamanho variável em um bloco de dados de tamanho fixo. Conforme discutido, qualquer função hash criptográfica é fundamentada sobre dois princípios básicos, a propriedade de mão única e a propriedade livre de colisão. Verificamos que a propriedade de mão única garante que a partir do código hash não será possível retornar ao seu valor inicial da mensagem. Por sua vez, a propriedade livre de colisão garante que duas mensagens distintas não serão mapeadas com o mesmo valor de hash.

Exploramos o uso das funções hash criptográficas, apresentamos algumas das principais aplicações. Entre as aplicações, demonstramos como as funções hash criptográfica podem ser utilizadas para implementar serviços de autenticação de mensagens. Ainda, exploramos as funções hash criptográficas para fornecer o processo de assinatura digital. Adicionalmente, apresentamos algumas outras aplicações das funções hash criptográficas, tais como gerador de números pseudoaleatórios, detecção de intrusão, detecção de vírus, alguns protocolos da internet entre outros.

Enfatizamos os requisitos de segurança das funções hash criptográficas, verificamos que existem basicamente sete requisitos que devem ser avaliados na implementação dos algoritmos de hash: tamanho de entrada variável; tamanho de saída fixo; propriedade de mão única; resistência à colisão fraca; resistência à colisão forte e pseudoaleatoridade. Estudamos cada um desses requisitos.

Por conseguinte, nos aprofundamos nos principais algoritmos que implementam as funções de hash. Primeiramente, apresentamos o algoritmo MD5, um dos algoritmos de hash mais utilizados. Abordamos as fraquezas do MD5 e como os adversários exploram tais vulnerabilidades. Conforme discutido, mesmo após a exposição das fraquezas e da depreciação dos especialistas de área de segurança o MD5 ainda continua a ser utilizado. Adicionalmente, apresentamos as funções hash baseadas em Cipher Block Chaining, mostramos quais os tipos de ataques utilizados pelo adversário, bem como algumas estratégias utilizadas para evitá-los. Na sequência, conhecemos a família do algoritmo SHA, exploramos sua evolução do primeiro algoritmo até os algoritmos utilizados atualmente.

Referências Bibliográficas

HANSEN, T. **RFC 6234 - US Secure Hash Algorithms.** Internet Engineering Task Force (IETF), 2011.

JUENEMAN, R. **Electronic Document Authentication.** IEEE Network Magazine, abr. 1987.

MITCHELL, C.; PIPER, F.; WILD, P. **Digital Signatures**, in SIMM, 1992.

NIST. FIPS PUB 180-1 - Federal Information Processing Standards Publication. National Institute of Standards and Technology (NIST), 2021. Disponível em: < https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub180-1.pdf>. Acesso em: 17 de jul. de 2021.

NIST. **FIPS PUB 180-2 - Federal Information Processing Standards Publication.** National Institute of Standards and Technology (NIST), 2021. Disponível em: <https://csrc.nist.gov/CSRC/media/Publications/fips/180/2/archive/2002-08-01/documents/fips180-2.pdf>. Acesso em: 17 de jul. de 2021.

NIST. **FIPS PUB 180-3 - Federal Information Processing Standards Publication**. National Institute of Standards and Technology (NIST), 2021. Disponível em: <https://csrc.nist.gov/csrc/media/publications/fips/180/3/archive/2008-10-31/documents/fips180-3_final.pdf>. Acesso em: 17 de jul. de 2021.

NIST. **FIPS PUB 180-4 - Federal Information Processing Standards Publication.** National Institute of Standards and Technology (NIST), 2021. Disponível em: < https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>. Acesso em: 17 de jul. de 2021.

RABIN, M. **Digitalized Signatures.** In. Foundations of Secure Computation, DeMillo, R.; Dobkin, D.; Jones, A. e Lipton, R., eds. New York: Academic Press, 1978.

RIVEST, R. **RFC 1321 - The MD5 Message-Digest Algorithm**. Network Working Group, 1992.

STALLINGS, William. **Criptografia e Segurança de Redes: Princípios e Práticas.** São Paulo: Pearson Prentice Hall, 2008.

STALLINGS, William; BROWN, Lawrie. **Segurança de computadores**: princípios e práticas. Rio de Janeiro: Elsevier Campus, 2014.

WANG, X.; YIN, Y.; YU, H. **Finding Collisions in the Full SHA-1**. Proceedings, Crypto'05, 2005. publicado por Springer-Verlag.