

# Criptografia Aplicada

---

Funções de hash

# Sumário

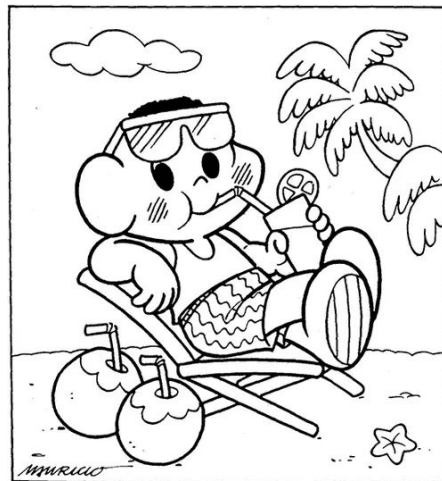
- Definições básicas
- Requisitos e segurança
- O algoritmo SHA
- Funções de hash na prática

# Recapitulando..

- Cifragem de mensagens provê *confidencialidade*
  - protegendo a mensagem de um atacante passivo
- Um atacante ativo pode *modificar* o conteúdo de uma mensagem
  - às vezes, não conseguimos evitar esse tipo de ataque
- Como a criptografia pode nos ajudar a identificar se uma modificação aconteceu?
  - esse objetivo é chamado de *integridade de dados*

# Integridade de dados

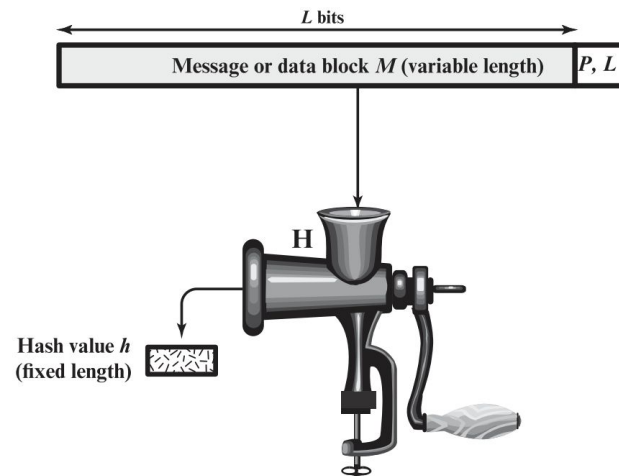
- Se eu tenho o dado original salvo em um local seguro, posso usar ele para comparação e identificação de modificações
- **Problema:** se o dado é muito grande, preciso de muito espaço de armazenamento
- **Solução:** funções de hash



Fonte: <https://tinyurl.com/7-erros>

# Funções de hash

- Uma *função de hash* recebe como entrada uma mensagem  $M$  de tamanho variável e produz uma saída  $H(M)$  de tamanho fixo
  - chamamos a saída  $h = H(M)$  de *hash*
- São conhecidas como funções de caminho único.
- As saídas devem ser bem distribuídas e com aparência aleatória.
- Uma mudança em qualquer bit de uma mensagem  $M$  deve resultar em um valor  $H(M)$  diferente.
  - conhecido como efeito avalanche



$P, L$  = padding plus length field

**Figure 11.1** Cryptographic Hash Function;  $h = H(M)$

Imagens: W. Stallings. *Cryptography and network security*. Cap 11.1

# Funções de hash criptográficas

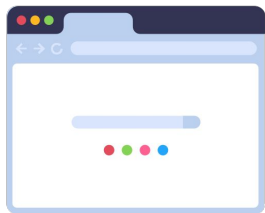
- Uma *função de hash criptográfica* é utilizada para prover garantia da integridade de dados
  - Possui certos requisitos de segurança
- Ela é utilizada para construir um "resumo" do dado
  - Se o dado foi alterado, o seu resumo se torna inválido

# Exemplo

- Para verificar a integridade de um dado:
  - calculamos seu resumo/*hash* e armazenamos em um local seguro
  - o dado pode ser armazenado ou transmitido através de um canal inseguro
  - a verificação posterior consiste em recalculer o resumo/*hash* e verificar se o valor é igual ao armazenado



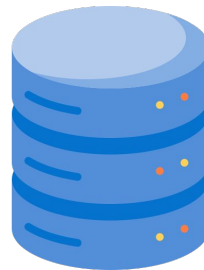
# Exemplo



minhasenha123

Função de  
hash

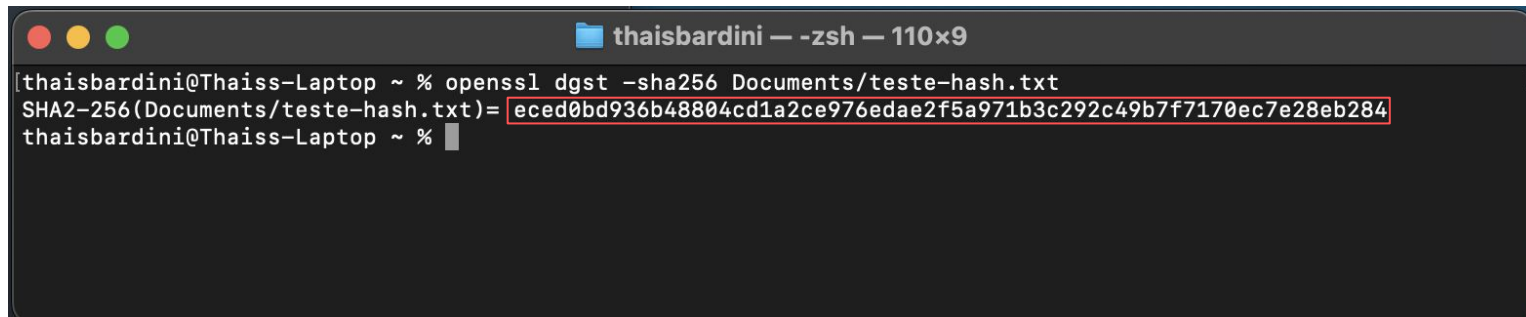
56b497f1a1929b





# Exemplo

*openssl dgst -sha256 nome-do-arquivo*



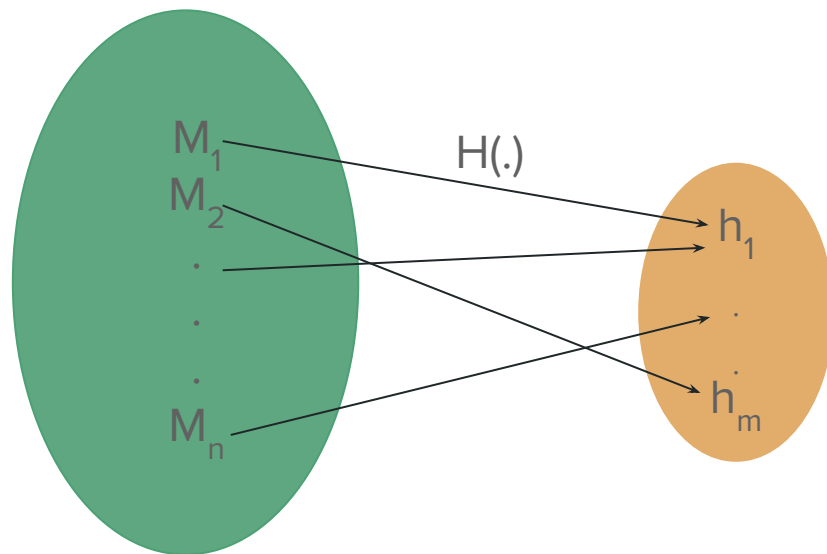
```
thaisbardini — -zsh — 110x9
[thaisbardini@Thaiss-Laptop ~ % openssl dgst -sha256 Documents/teste-hash.txt
SHA2-256(Documents/teste-hash.txt)= eced0bd936b48804cd1a2ce976edae2f5a971b3c292c49b7f7170ec7e28eb284
thaisbardini@Thaiss-Laptop ~ %
```

# Sumário

- Definições básicas
- **Requisitos e segurança**
- O algoritmo SHA
- Funções de hash na prática

# Colisão

- Como a entrada tem tamanho arbitrário e a saída tem tamanho fixo, valores diferentes de  $M$  resultam no mesmo hash  $h$ .
- Chamamos isso de *colisão de hash*.



# Colisão



```
thaisbardini — -zsh — 80x24
[thaisbardini@Thaiss-Laptop ~ % openssl dgst -md5 Documents/ship.jpg
MD5(Documents/ship.jpg)= 253dd04e87492e4fc3471de5e776bc3d
[thaisbardini@Thaiss-Laptop ~ % openssl dgst -md5 Documents/plane.jpg
MD5(Documents/plane.jpg)= 253dd04e87492e4fc3471de5e776bc3d
thaisbardini@Thaiss-Laptop ~ % █
```

# Requisitos

## Requisitos de segurança de uma função de hash criptográfica:

- Tamanho de entrada variável e de saída fixo
- Fácil de computar para qualquer entrada
- **Resistência à pré-imagem:** dado  $H(x)$ , é inviável achar  $x$
- **Resistência à 2a pré-imagem:** dados  $x$  e  $H(x)$ , é inviável encontrar  $y \neq x$  com  $H(x) = H(y)$
- **Resistência à colisão:** inviável achar dois valores  $x \neq y$  com  $H(x) = H(y)$
- Saída passa nos testes de pseudo-aleatoriedade

# Segurança das funções de hash

- **Ataque de força bruta:** depende apenas do tamanho da saída da função, e não do algoritmo.
- Assuma uma função  $H$  com saída de  $m$  bits.
- Para encontrar  $x$  dado  $h = H(x)$ , o esforço é proporcional a  $2^m$  tentativas
  - em média,  $2^{m-1}$  (metade das possibilidades)
  - primeira e segunda pré-imagens
- Para encontrar dois valores diferentes  $x$  e  $y$  tal que  $H(x) = H(y)$ , precisamos de um esforço muito menor, graças ao **paradoxo do aniversário**.
  - de apenas  $2^{m/2}$  tentativas
  - colisão

# Paradoxo do aniversário

- Probabilidade de colisão pode ser avaliada usando o paradoxo do aniversário
  - Em um grupo de 23 pessoas aleatoriamente selecionadas, duas delas têm a mesma data de aniversário com probabilidade de pelo menos 50%
  - 365 dias do ano,  $\sqrt{365} = 22.3$
  - Encontrar duas pessoas com o mesmo aniversário é como achar uma colisão no hash
- Segurança da função SHA256:
  - Espaço de  $2^{256}$  possibilidades de hash
  - Paradoxo:  $\sqrt{2^{256}}$  para 50% de chances
  - $2^{256/2} = 3.4028237e+38$  tentativas

As { the } Dean of Blakewell College, I have { had the pleasure of knowing } Cherise  
known  
Rosetti for the { last } four years. She { has been } { a tremendous } { asset to }  
past { was } { an outstanding } { role model in }  
{ our } { school. I { would like to take this opportunity to } recommend Cherise for your  
the } { wholeheartedly }  
{ school's } graduate program. I { am } { confident } { that } { she } will  
{ } { feel } { certain } { } { Cherise }  
{ continue to } succeed in her studies. { She } is a dedicated student and  
{ } { Cherise }  
{ thus far her grades } { have been } { exemplary } In class,  
{ her grades thus far } { are } { excellent }  
{ she } { has proven to be } a take-charge { person } { who is } able to  
{ Cherise } { has been } { individual } { }  
successfully develop plans and implement them.  
{ She } has also assisted { us } in our admissions office. { She } has  
{ Cherise }  
{ successfully } demonstrated leadership ability by counseling new and prospective students.  
{ }  
{ Her } advice has been { a great } help to these students, many of whom  
{ Cherise's } { of considerable }  
have { taken time to share } their comments with me regarding her pleasant and  
shared  
{ encouraging } attitude. { For these reasons } I  
{ reassuring } { It is for these reasons that }  
{ highly recommend } Cherise { without reservation } Her { ambition } and  
{ offer high recommendations for } { unreservedly } { drive }  
{ abilities } will { truly } be an { asset to } your { establishment }  
{ potential } { surely } { plus for } { school }

Figure 11.7 A Letter in  $2^{38}$  Variations

# Sumário

- Definições básicas
- Requisitos e segurança
- **O algoritmo SHA**
- Funções de hash na prática



# Funções de hash

- As funções mais utilizadas hoje em dia são as da família Secure Hash Algorithm (SHA)
- Desenvolvidas pelo Instituto Nacional de Padrões e Tecnologia (NIST) e padronizadas em 1993
  - Em 1995, a versão SHA-0 foi comprometida e substituída pela nova versão SHA-1
  - Em 2002, três novas versões foram criadas e chamadas de SHA-2: SHA-256, SHA-384 e SHA-512
  - Em 2008, foi incluída a versão SHA-224
  - Em 2015, foram adicionadas a SHA-512/224 e SHA-512/256
  - Foi encontrado um ataque de colisão no SHA-1 e hoje ele não é mais recomendado.
  - Uma competição do NIST anunciada em 2007 produziu a nova geração de funções de hash, a SHA-3. Ela foi padronizada em 2012.
- A especificação dos algoritmos é apresentada atualmente na [FIPS PUB 180-4](#), conhecida como Secure Hash Standard (SHS)

**Table 11.3** Comparison of SHA Parameters

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
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

*Note:* All sizes are measured in bits.

Imagens: W. Stallings. *Cryptography and network security*. Cap 11.5

# SHA-512

- **Entrada:** mensagem  $< 2^{128}$  bits
- Entrada processada em blocos de 1024 bits cada
- **Saída:** resumo de 512 bits

Imagem: W. Stallings. *Cryptography and network security*. Cap 11.5

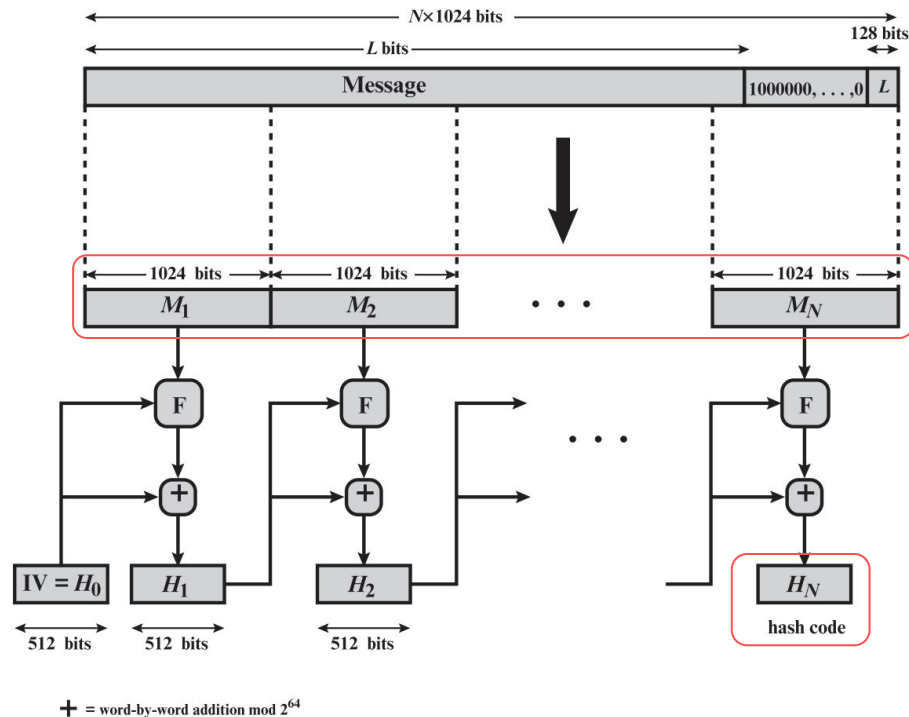


Figure 11.9 Message Digest Generation Using SHA-512

# SHA-512

- **Entrada:** mensagem  $< 2^{128}$  bits
- Entrada processada em blocos de 1024 bits cada
- **Saída:** resumo de 512 bits
- Padding 10...0 é adicionado para garantir comprimento específico

Imagem: W. Stallings. *Cryptography and network security*. Cap 11.5

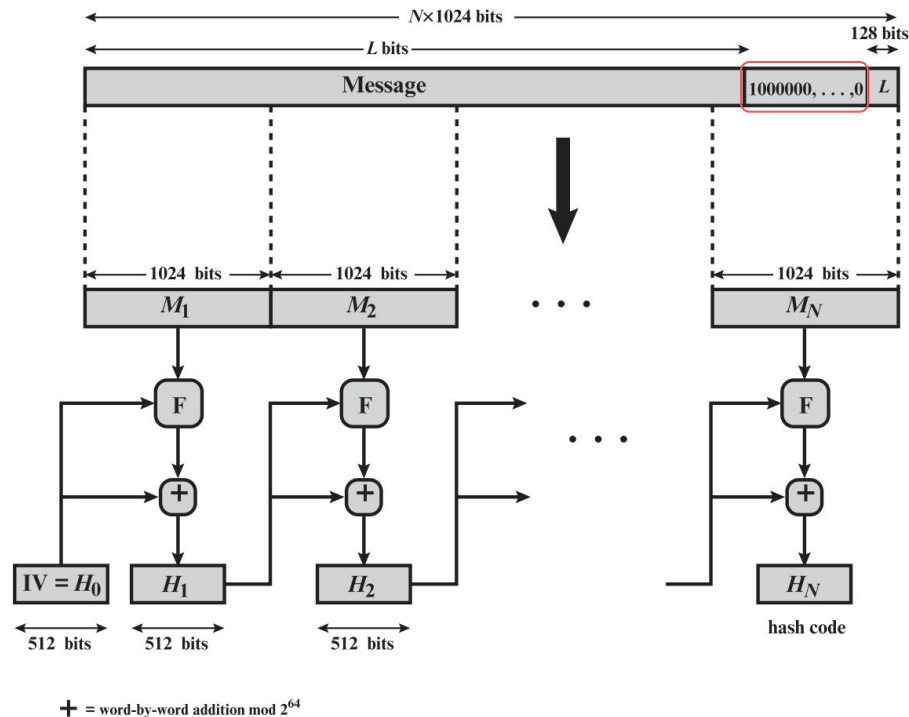


Figure 11.9 Message Digest Generation Using SHA-512

# SHA-512

- **Entrada:** mensagem  $< 2^{128}$  bits
- Entrada processada em blocos de 1024 bits cada
- **Saída:** resumo de 512 bits
- Padding 10...0 é adicionado para garantir comprimento específico
- Bloco de 125 bits contém o tamanho da mensagem original (antes do padding)

Imagem: W. Stallings. *Cryptography and network security*. Cap 11.5

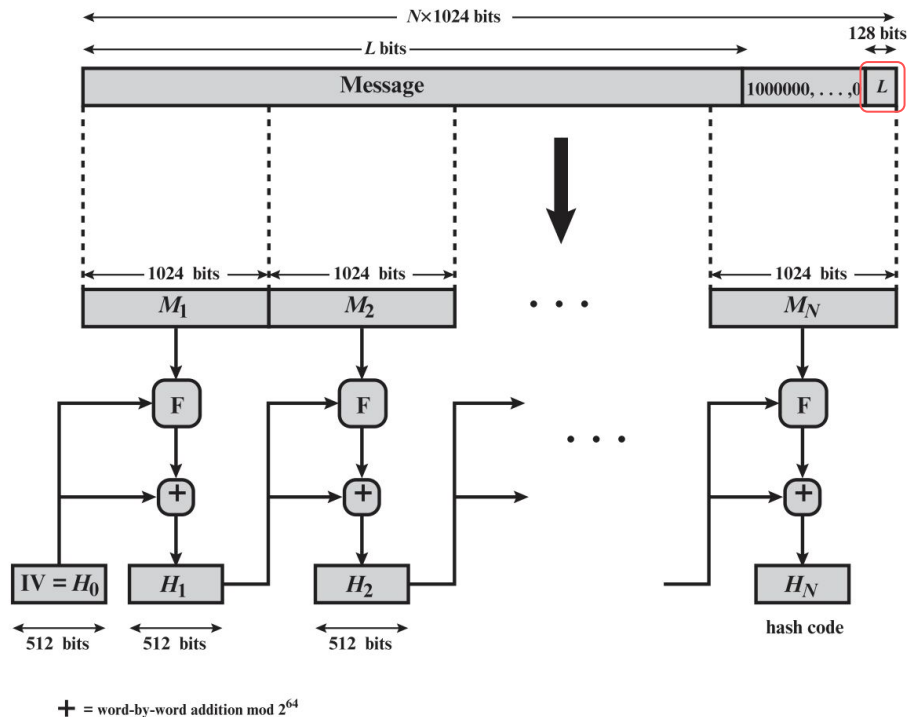


Figure 11.9 Message Digest Generation Using SHA-512

# SHA-512

- **Entrada:** mensagem  $< 2^{128}$  bits
- Entrada processada em blocos de 1024 bits cada
- **Saída:** resumo de 512 bits
- Padding 10...0 é adicionado para garantir comprimento específico
- Bloco de 125 bits contém o tamanho da mensagem original (antes do padding)
- Inicialização do  $H_0$  com valores pré-determinados ([FIPS PUB 180-4](#))

Imagem: W. Stallings. *Cryptography and network security*. Cap 11.5

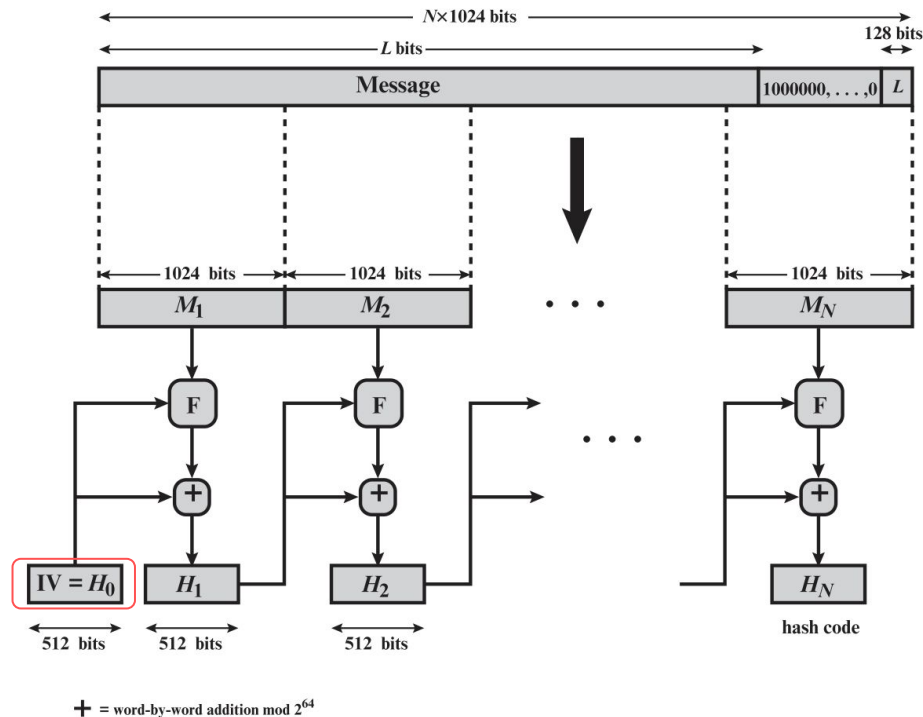


Figure 11.9 Message Digest Generation Using SHA-512

# SHA-512

- **Entrada:** mensagem  $< 2^{128}$  bits
- Entrada processada em blocos de 1024 bits cada
- **Saída:** resumo de 512 bits
- Padding 10...0 é adicionado para garantir comprimento específico
- Bloco de 125 bits contém o tamanho da mensagem original (antes do padding)
- Inicialização do  $H_0$  com valores pré-determinados ([FIPS PUB 180-4](#))
- Processamento de cada bloco utilizando o valor intermediário  $H_{i-1}$

Imagem: W. Stallings. *Cryptography and network security*. Cap 11.5

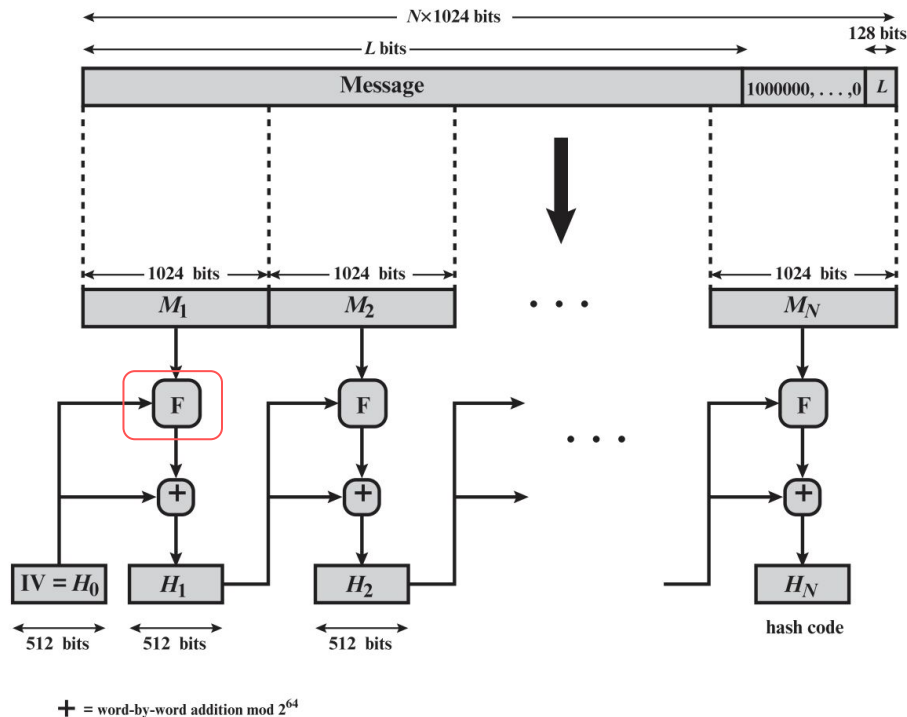


Figure 11.9 Message Digest Generation Using SHA-512

# SHA-512

- A função F consistem em 80 rounds para cada bloco de texto a ser processado
- **Entrada:** bloco  $M_i$  e valor intermediário  $H_{i-1}$
- **Saída:** valor intermediário  $H_i$

Imagem: W. Stallings. *Cryptography and network security*. Cap 11.5

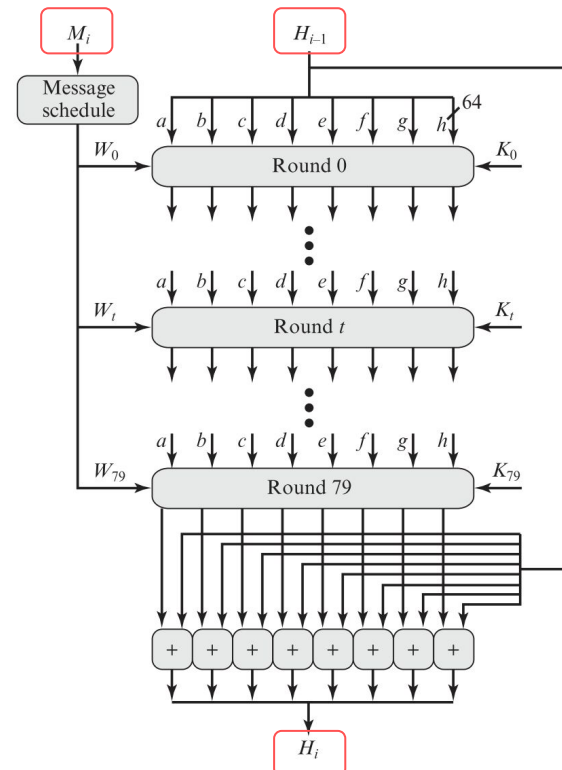


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

# SHA-512

- A função F consistem em 80 rounds para cada bloco de texto a ser processado
- **Entrada:** bloco  $M_i$  e valor intermediário  $H_{i-1}$
- **Saída:** valor intermediário  $H_i$
- Cada  $W_t$  consiste em 64 bits derivados da  $M_i$
- Cada  $K_t$  é uma constante pré-determinada, que provê uma "aleatoriedade" à saída

Imagem: W. Stallings. *Cryptography and network security*. Cap 11.5

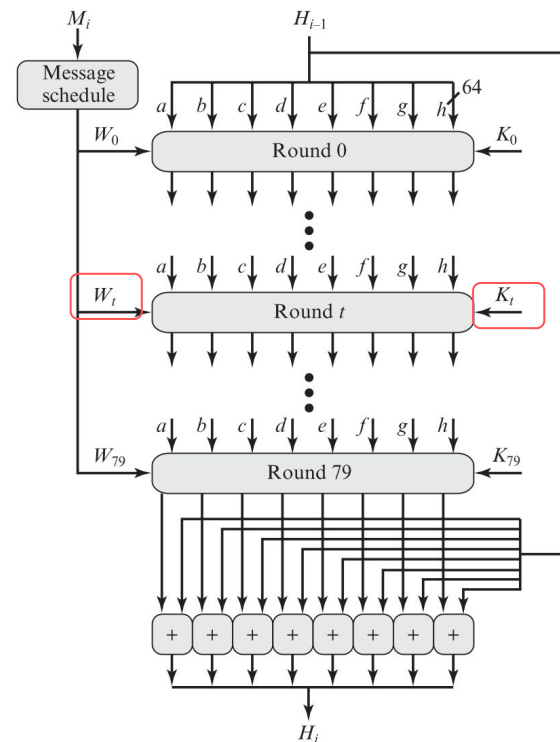


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block



# SHA-512

- A função F consistem em 80 rounds para cada bloco de texto a ser processado
- **Entrada:** bloco  $M_i$  e valor intermediário  $H_{i-1}$
- **Saída:** valor intermediário  $H_i$
- Cada  $W_t$  consiste em 64 bits derivados da  $M_i$
- Cada  $K_t$  é uma constante pré-determinada, que provê uma "aleatoriedade" à saída
- Cada round calcula operações elementares, como somas, operações binárias (and, xor, not) e rotações

Imagem: W. Stallings. *Cryptography and network security*. Cap 11.5

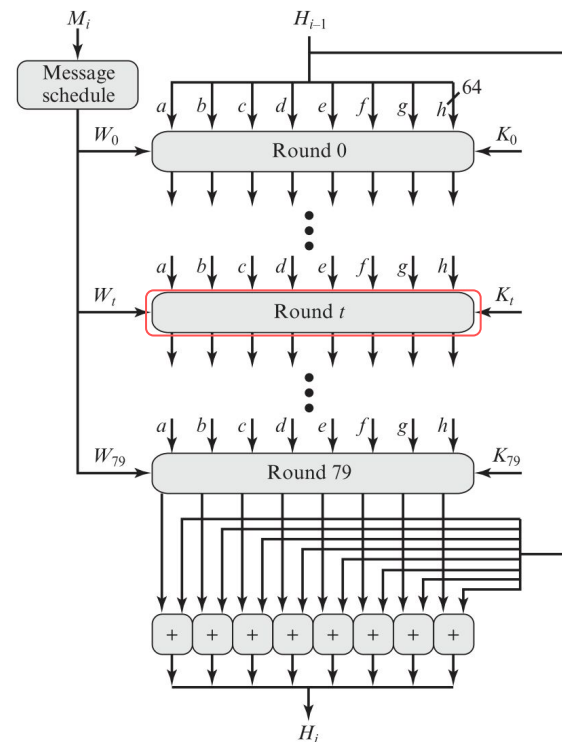


Figure 11.10 SHA-512 Processing of a Single 1024-Bit Block

# SHA-512

- Os passos do algoritmo adicionam redundância e interdependência
- Cada bit da saída depende de todos os bits da entrada
- Isso faz com que seja improvável que duas mensagens aleatórias mapeiem para o mesmo hash.
- Os outros algoritmos da família SHA-2 funcionam de maneira similar
  - ver mais em [FIPS PUB 180-4](#)

# Sumário

- Definições básicas
- Requisitos e segurança
- O algoritmo SHA
- **Funções de hash na prática**

# Aplicações

- As funções de hash são utilizadas em uma grande variedade de aplicações e protocolos
- Exemplos de aplicações:
  - Autenticação de mensagens
  - Assinaturas digitais
  - Armazenamento seguro de senhas
  - Geração de números pseudo-aleatórios (geração de parâmetros e chaves)
  - Imutabilidade de registros em uma blockchain
  - Esquemas de comprometimento
- O uso de uma função de hash insegura compromete a aplicação

# Atividade: calculando hashes

- Vamos praticar utilizando o openssl:

openssl dgst -algoritmo arquivo

- Calcule o hash do pdf da primeira aula usando o sha256 e compare com os colegas
- Calcule o hash do arquivo **teste-hash.txt** disponível no Canvas
  - utilize as funções SHA-256 e SHA-512 e observe a diferença nas saídas
- Modifique o arquivo e gere novamente o hash
  - observe o efeito avalanche ao se modificar o valor de entrada
- Calcule o hash das imagens disponíveis no canvas
  - utilize a função md5, o que acontece? E se utilizar o sha256?

# Resumo

- Funções de hash transformam dados de qualquer tamanho em "resumos" de tamanhos fixos;
- Utilizados na garantia de integridade de dados;
- Para serem seguros, as funções de hash criptográficas precisam seguir alguns requisitos;
  - caminho único, difícil encontrar colisões, efeito avalanche, saída com aparência aleatória
- Os principais algoritmos são os SHA, que processam a entrada em blocos e possuem diversos rounds de operações
- Escolha algoritmos seguros e mantenha-se atualizado sobre novas vulnerabilidades!

# Referências

- W. Stallings. *Cryptography and network security*. 7a edição.
  - Capítulos 11.1, 11.3, 11.5, 11.6
- D. Stinson e M. Paterson. *Cryptography: Theory and Practice*. 4a edição.
  - Capítulos 5.1, 5.2