Introdução à Criptografia

Resumo Criptográfico
Função Hash

Prof. Rodrigo Minetto
rminetto@dainf.ct.utfpr.edu.br
Universidade Tecnológica Federal do Paraná
Baseado em: Understanding Cryptography by Paar e Pelzl

Sumário

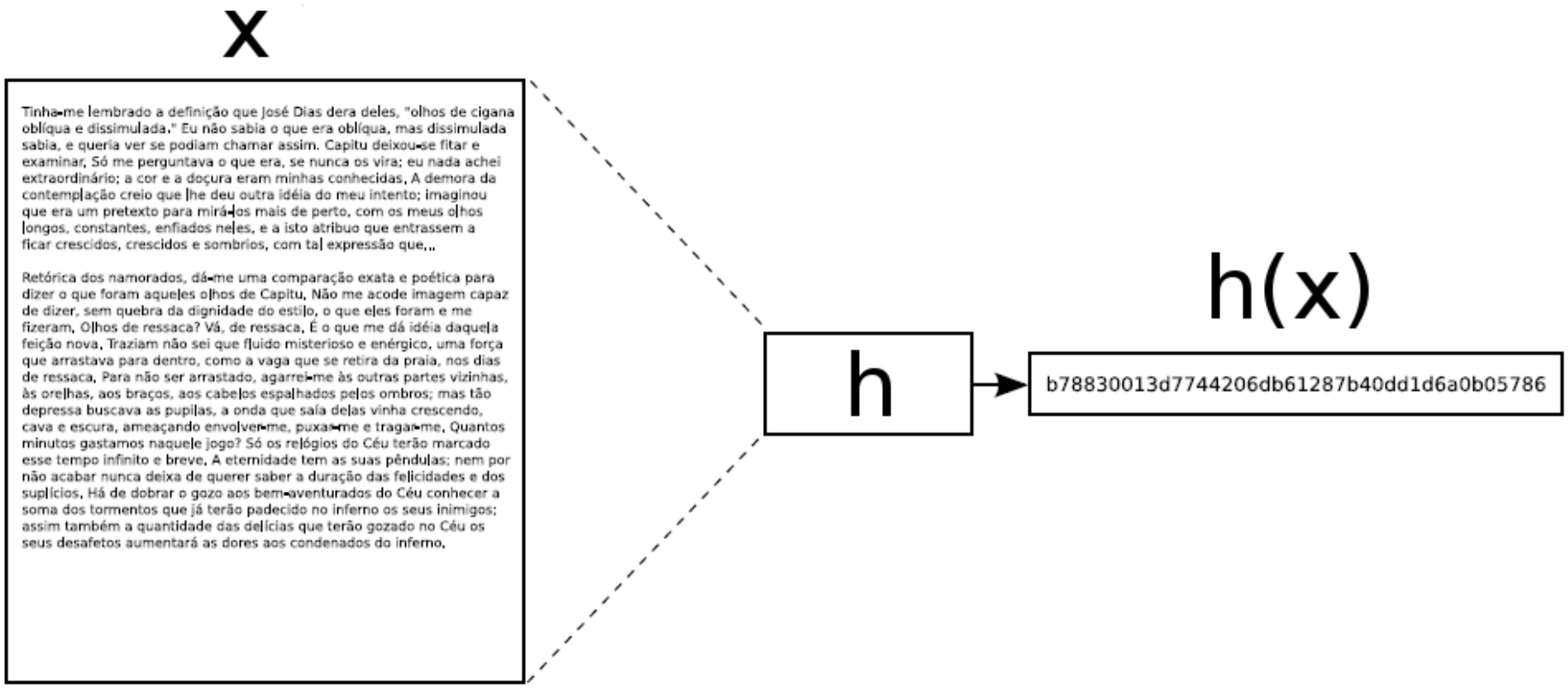
Introdução

Uma **função de resumo criptográfico** ou **função hash h** permite mapear informações de tamanho arbitrário $*$, como mensagens ou arquivos, em uma cadeia de bits de comprimento fixo m, o resultado também chamado de **resumo**, é geralmente de tamanho pequeno (128-512 bits). Formalmente

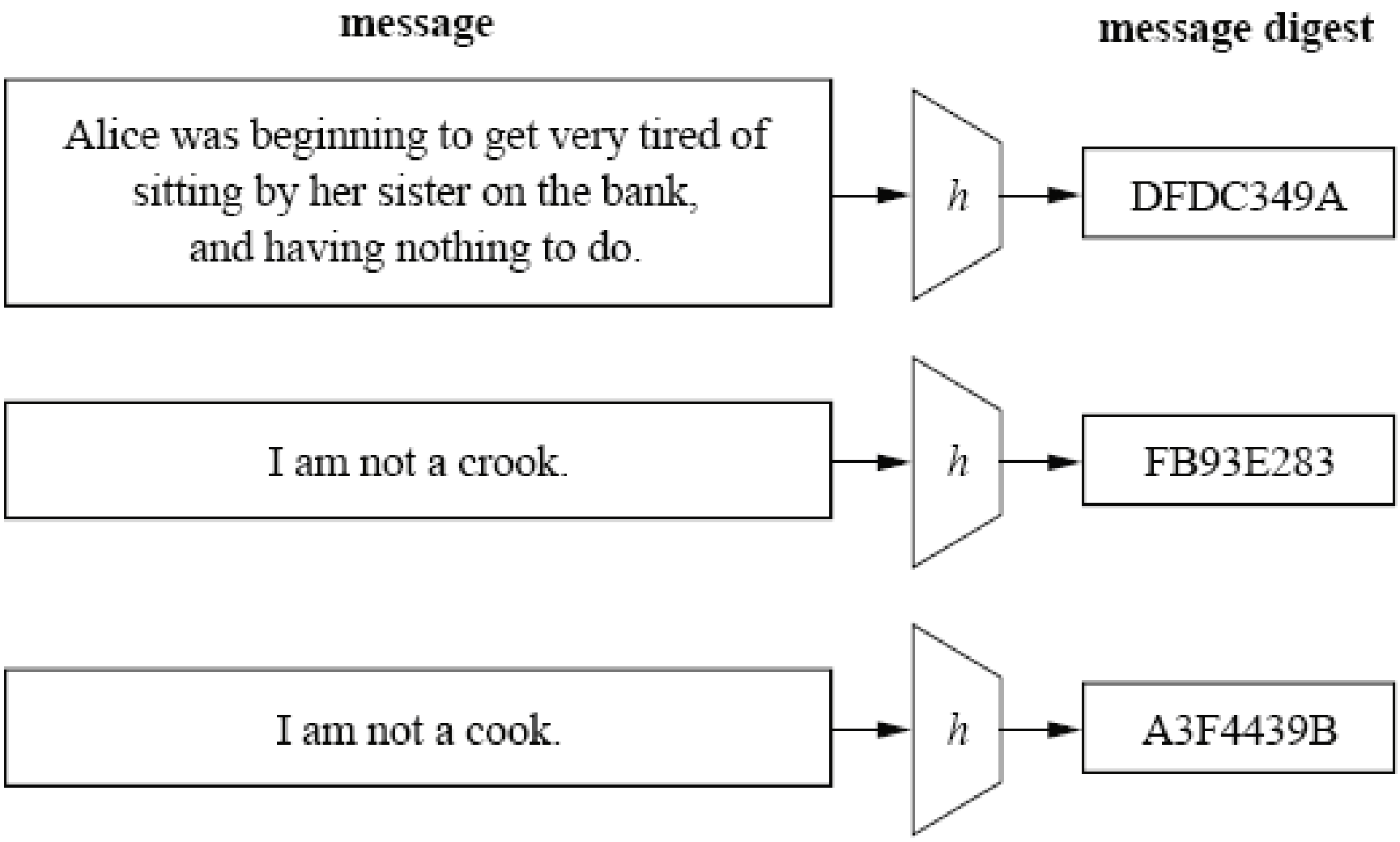
$$\mathbf{h} : \{0,1\}^* \rightarrow \{0,1\}^m$$

Introdução

$$\mathbf{h}(x) : \{0,1\}^{2^{64}-1} \rightarrow \{0,1\}^{160} \quad (\text{SHA-1})$$



Introdução

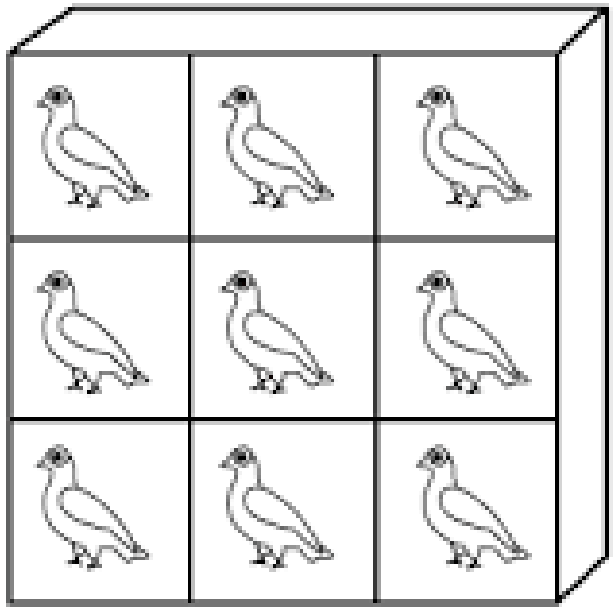

message
message digest
Alice was beginning to get very tired of sitting by her sister on the bank, and having nothing to do.
h
DFDC349A
I am not a crook.
h
FB93E283
I am not a cook.
h
A3F4439B

Introdução

De forma diferente de outros algoritmos para criptografia, funções hash **não** trabalham com **chaves** e não são consideradas funções de criptografia. Note que funções hash são funções de mão única, assim, não é possível recuperar totalmente (ou até mesmo parcialmente) a informação a partir do resumo. Essa premissa de cifragem e decifragem é essencial para um algoritmo de criptografia.

Introdução

Princípio da casa dos pombos (teorema de Dirichlet): se n pombos devem ser postos em m casas, e se $n > m$, então pelo menos uma casa irá conter mais de um pombo.

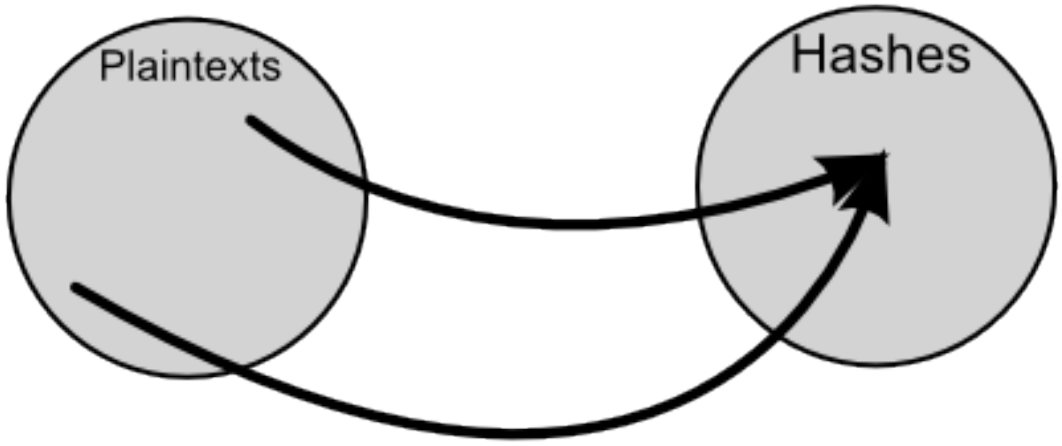


Introdução

Ou seja, dado que o domínio de entrada de um hash é maior do que sua imagem

$$\mathbf{h} : \{0,1\}^* \rightarrow \{0,1\}^m$$

é fácil ver que mais de uma mensagem será mapeada para o mesmo resumo.



Introdução

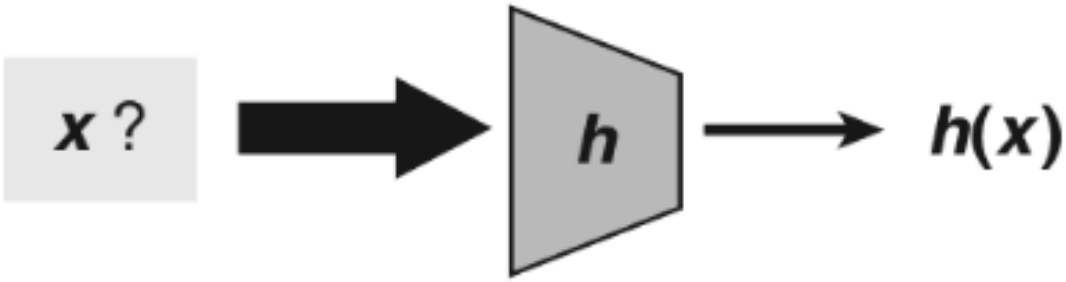
Algumas aplicações necessitam que seja computacionalmente inviável que um atacante encontre duas mensagens aleatórias que gerem o mesmo resumo (assinaturas digitais); outras que seja inviável encontrar uma mensagem dado o conhecimento do resumo. Desta forma, uma função hash precisa satisfazer algumas propriedades para que seja considerada segura:

- resistência à primeira pré-imagem;
- resistência à segunda pré-imagem;
- resistência a colisões.

Introdução

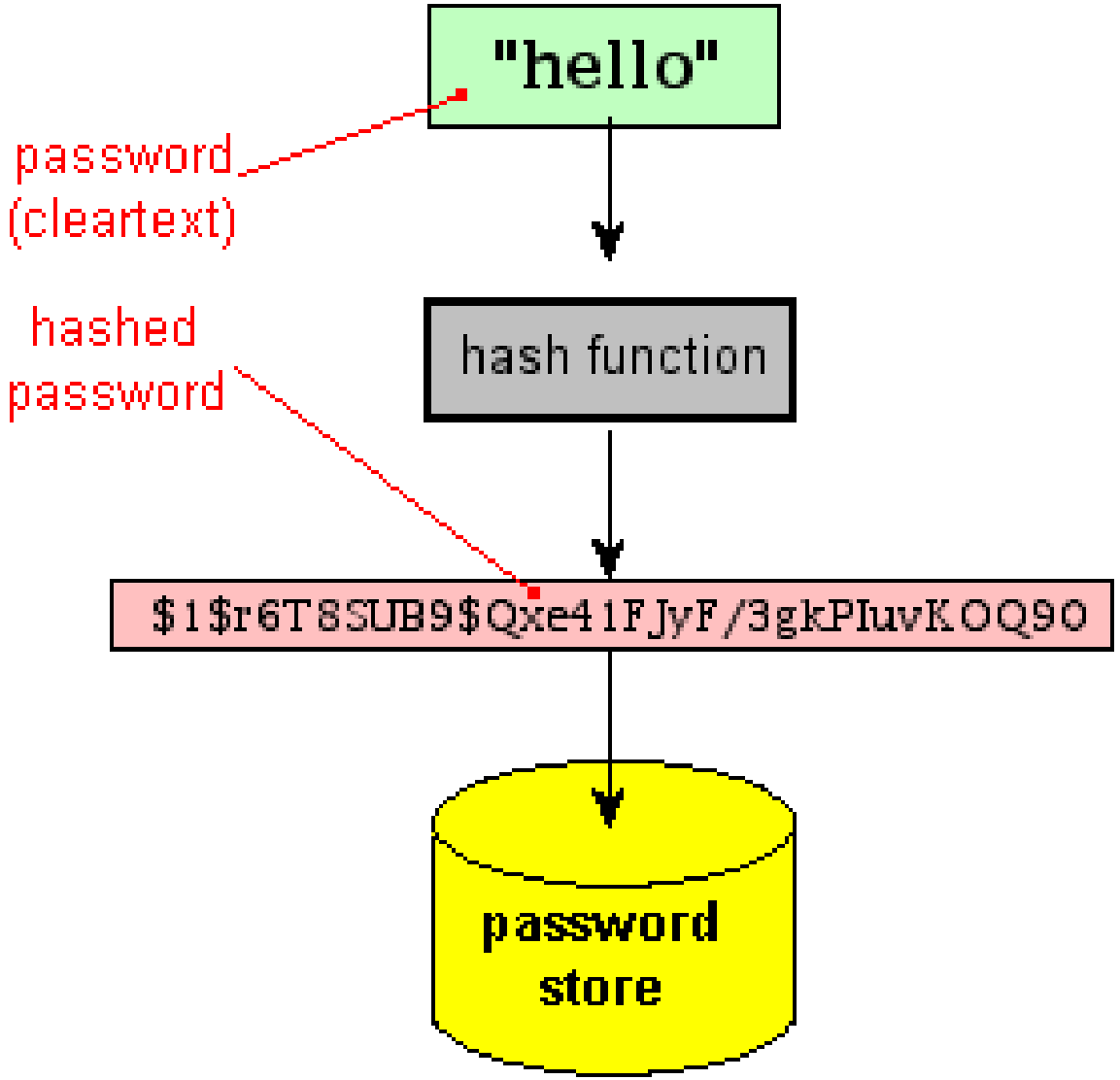
Resistência à pré-imagem (one-wayness): dado um resumo criptográfico por hash r, é computacionalmente inviável achar uma mensagem de entrada x tal que:

$$r = h(x)$$



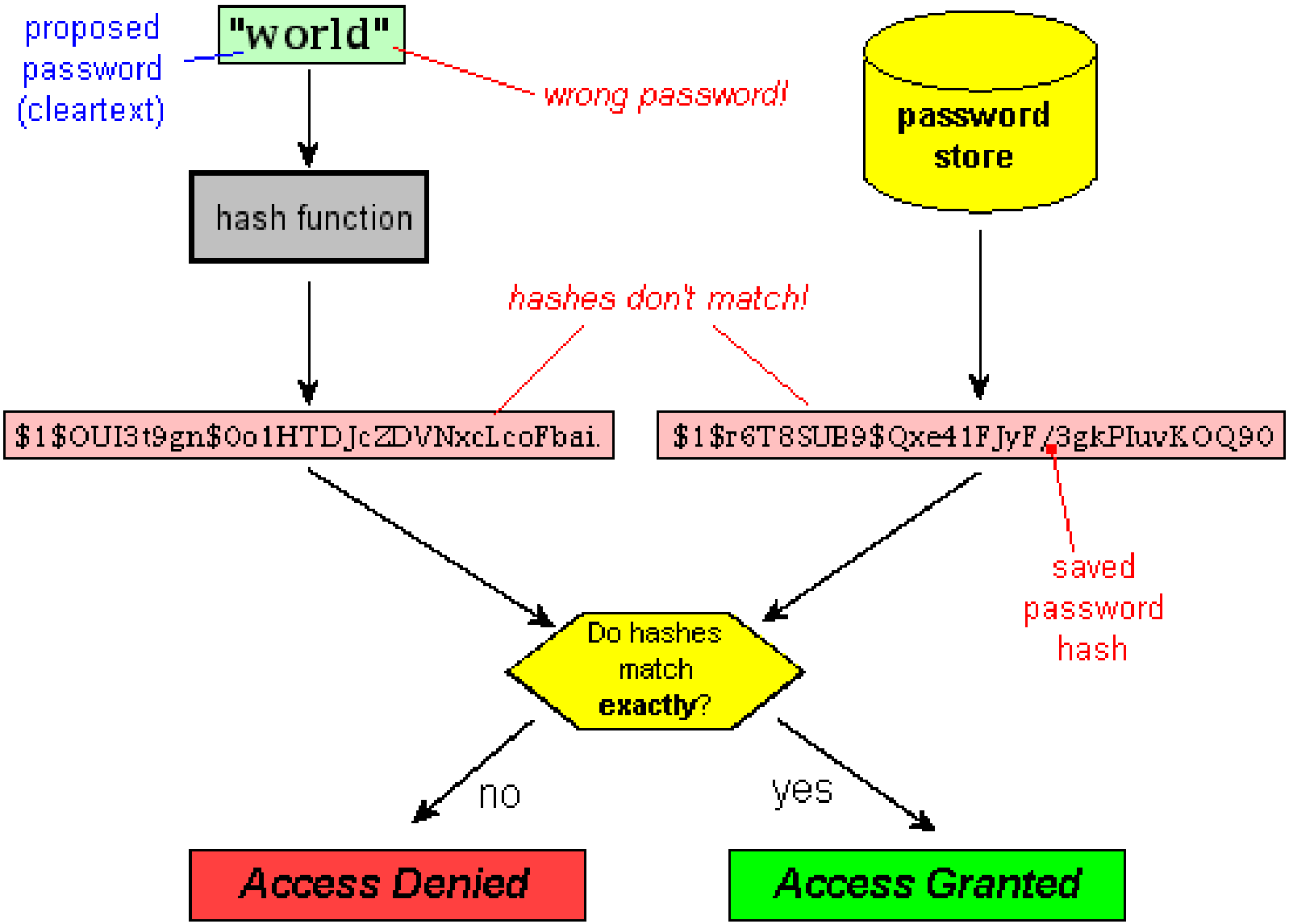
Introdução

Cenário: resistência à pré-imagem



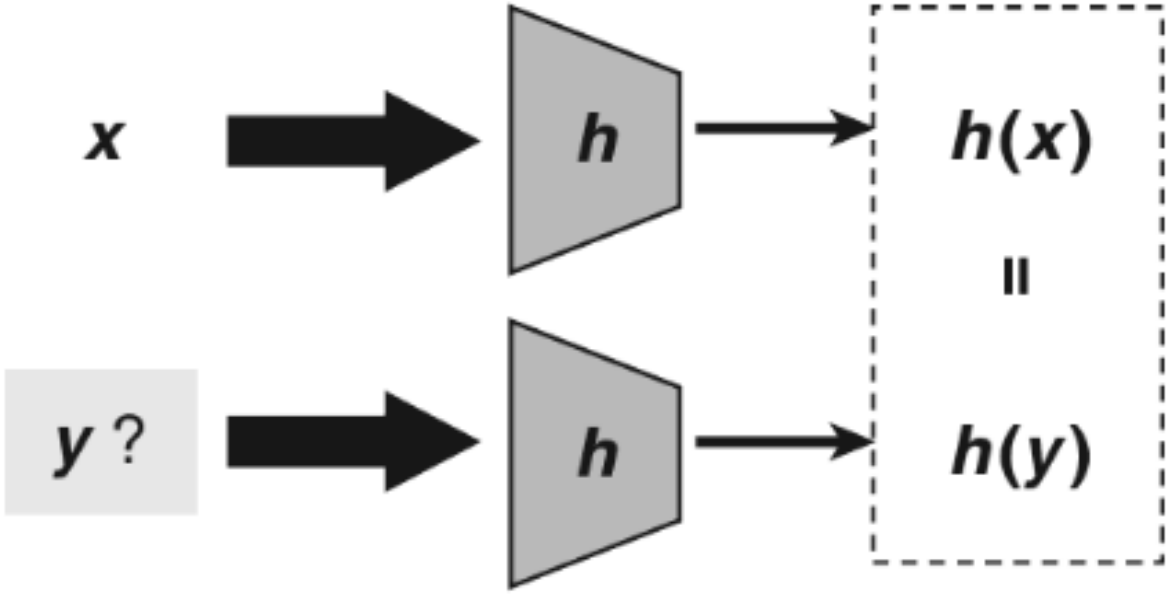
Introdução

Cenário: resistência à pré-imagem



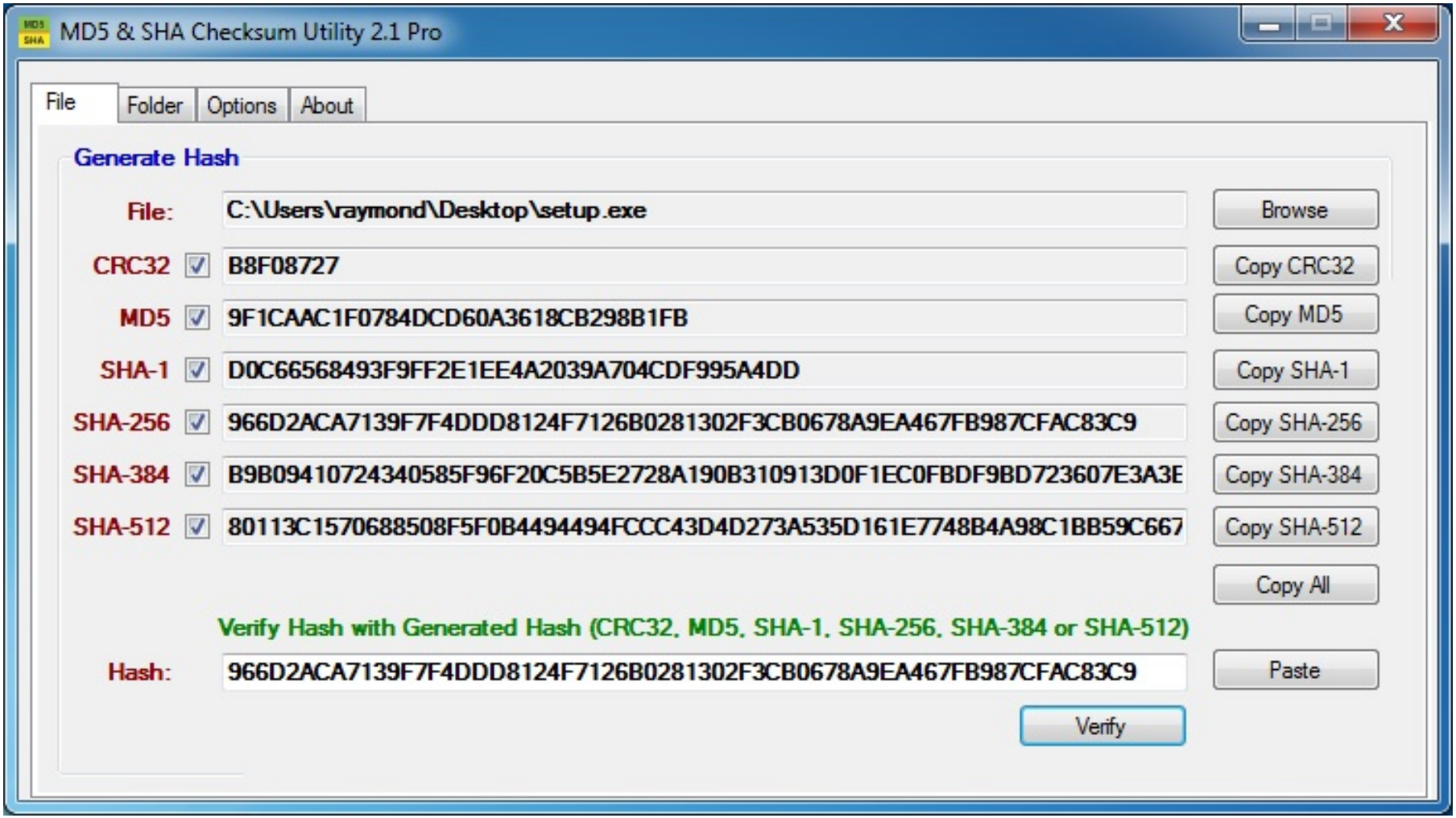
Introdução

Resistência à segunda pré-imagem (weak-collision): dado um resumo $h(x)$ de uma mensagem x, deve ser inviável encontrar uma mensagem $y \neq x$ tal que $h(x) = h(y)$.



Introdução

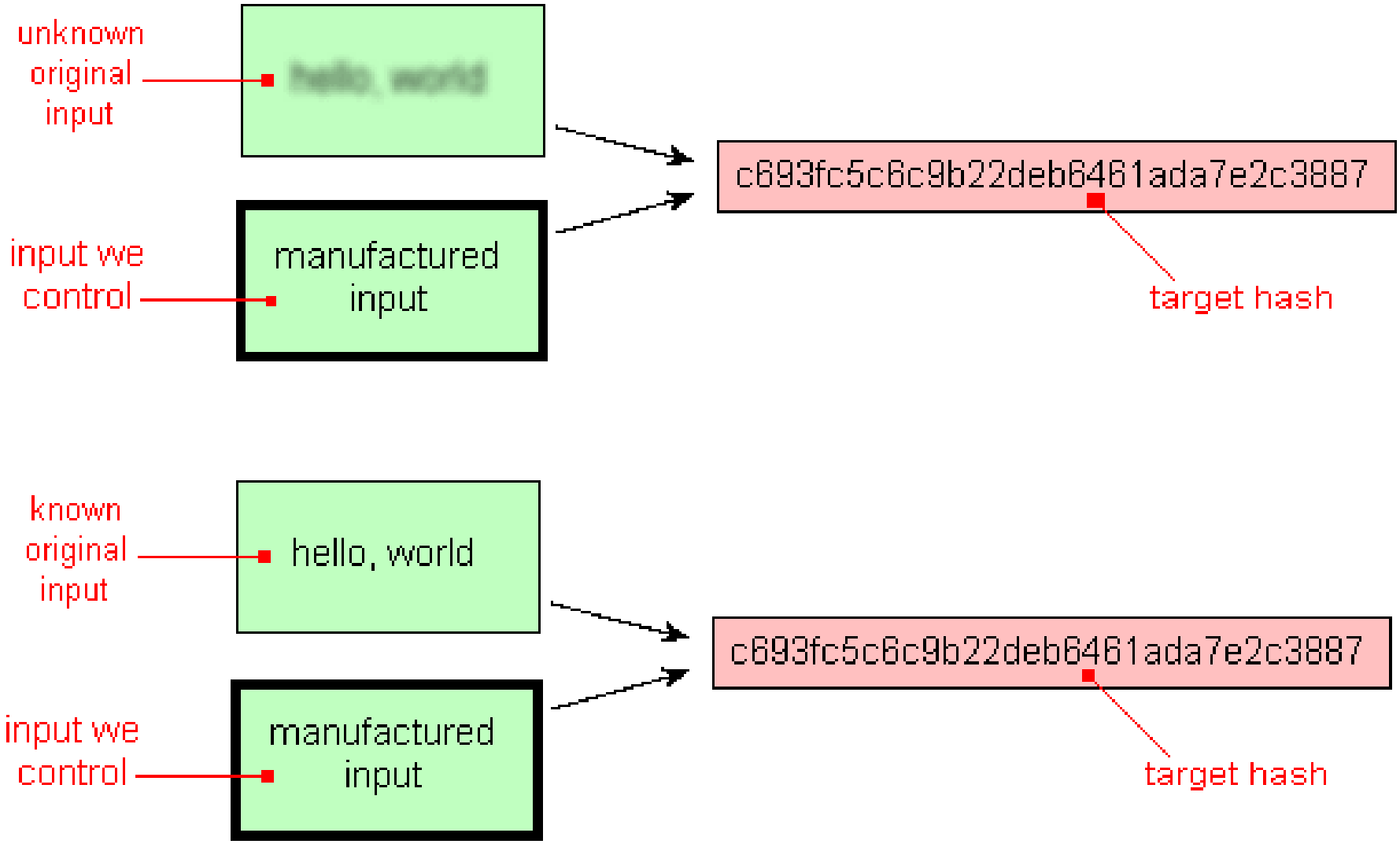
Cenário: resistência à segunda imagem



Ps. não é necessário resistência a primeira imagem aqui.

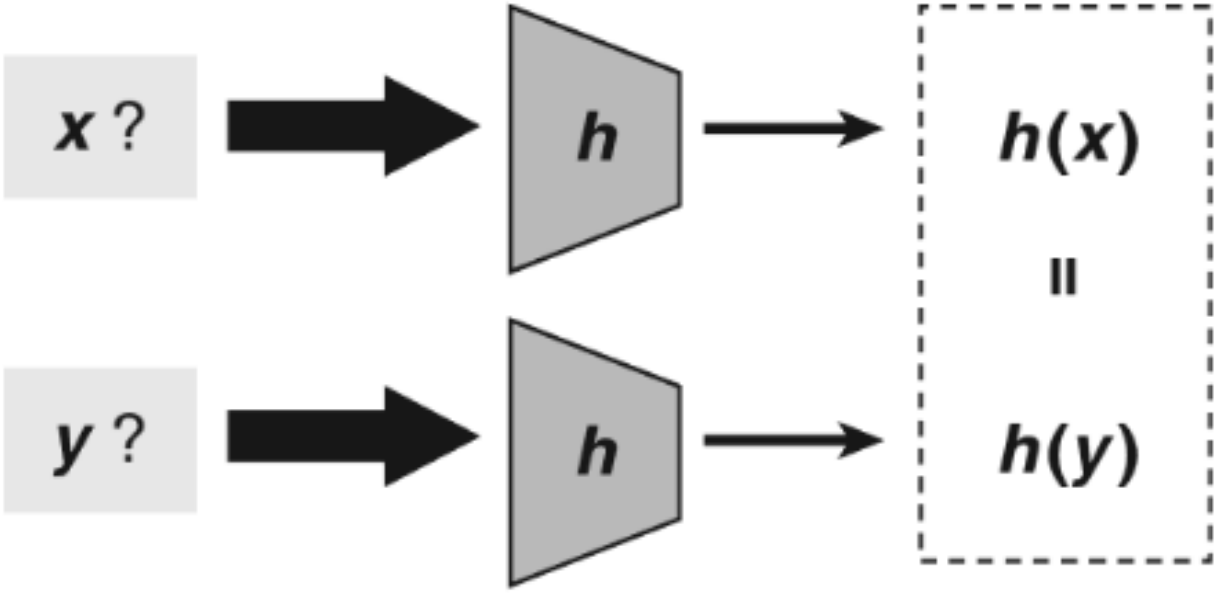
Introdução

Primeira **vs** Segunda imagem:



Introdução

Resistência à colisões (strong-collision): uma função hash é resistente a colisões se é computacionalmente inviável achar duas mensagens diferentes x e y com $h(x) = h(y)$.



Introdução

Resistência à colisões (strong-collision): qual seria um possível cenário de ataque aqui?

Mensagem original

(a) *Pay Fred Piper $200*
(b) *Pay F. Piper $200*
(c) *Pay F.C. Piper two hundred dollars*
(d) *Pay F.C. Piper two hundred dollars only*
(e) etc.

Mensagem de ataque

(a) *Pay Fred Piper $8000*
(b) *Pay F. Piper $8000*
(c) *Pay F.C. Piper eight thousand dollars*
(d) *Pay F.C. Piper eight thousand dollars only*
(e) etc.

Introdução

Note que a propriedade strong-collision é mais difícil de assegurar que a propriedade weak-collision, pois o atacante tem dois graus de liberdade para obter uma colisão. Exemplo:

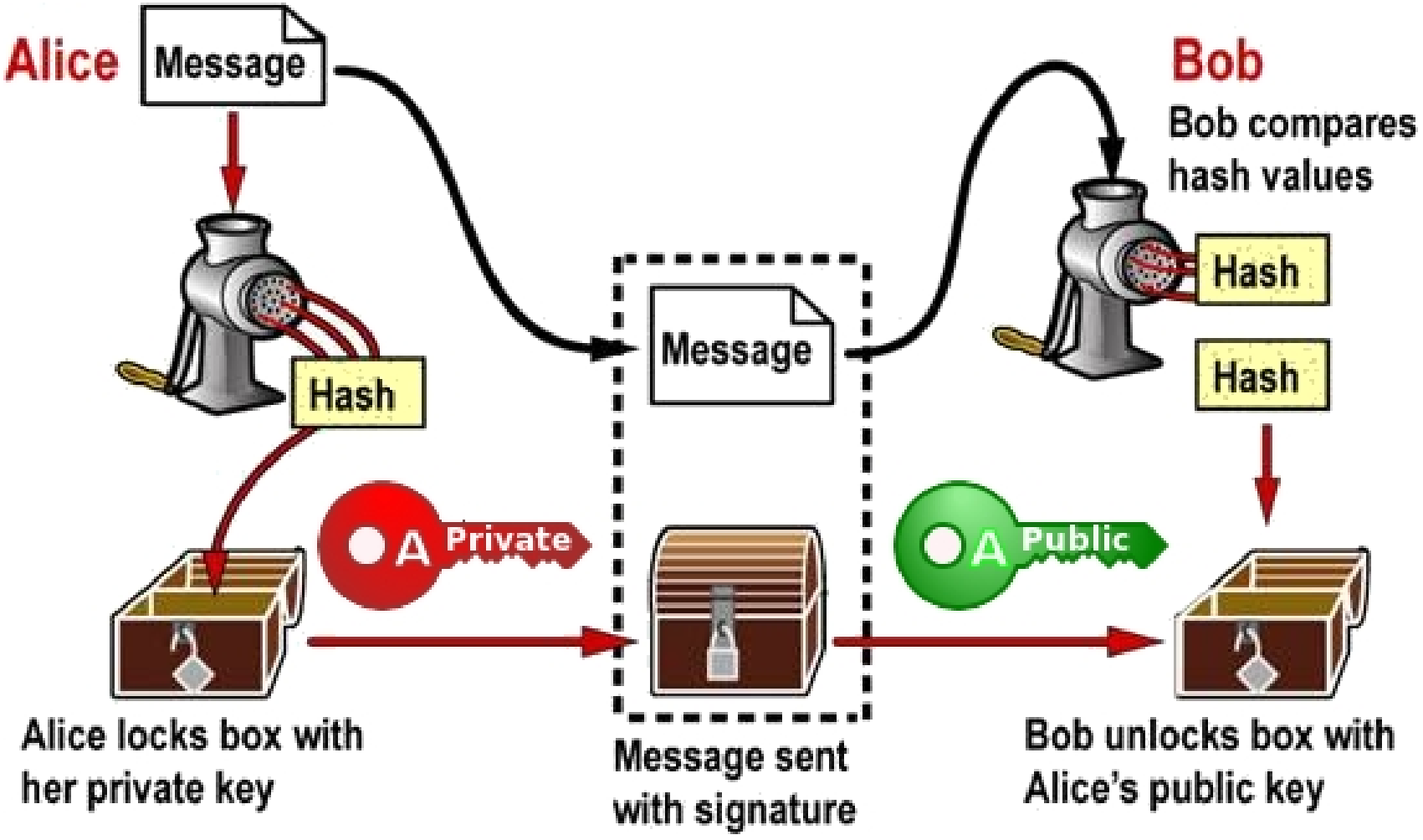
$x =$ Transfira R$ 100,00 para Oscar.

$y =$ Transfira R$ 100.000,00 para Oscar.

É mais fácil para Oscar se ele puder manipular (adicionar espaços, brancos, vírgulas) x **e** y para obter $h(x) = h(y)$.

Introdução

Qual seria um possível cenário de ataque aqui?



Introdução

Devido ao princípio da casa dos pombos, sabemos que colisões sempre existem. **Questão**: qual a probabilidade de uma colisão? Nossa primeira estimativa é que são necessários 2^n mensagens modificadas para uma colisão com um hash de n bits. No entanto, a quantidade de ataques necessários para uma colisão é bem menor, da ordem de $2^{n/2}$. Esse resultado surpreendente se deve ao paradoxo do aniversário.

Introdução

Paradoxo do aniversário: quantas pessoas são necessárias para que a probabilidade de que duas delas façam aniversário no mesmo dia (colisão) seja maior do que 50%? Prob. de colisão de aniversários para 2 pessoas:

$$p(\text{colisão}) = 1 - (\frac{365}{365} \times \frac{364}{365})$$

Introdução

Paradoxo do aniversário: quantas pessoas são necessárias para que a probabilidade de que duas delas façam aniversário no mesmo dia (colisão) seja maior do que 50%? Prob. de colisão de aniversários para 2 pessoas:

$$p(\text{colisão}) = 0.002$$

Introdução

Paradoxo do aniversário: quantas pessoas são necessárias para que a probabilidade de que duas delas façam aniversário no mesmo dia (colisão) seja maior do que 50%? Prob. de colisão de aniversários para 3 pessoas:

$$p(\text{colisão}) = 1 - (\frac{365}{365} \times \frac{364}{365} \times \frac{363}{365})$$

Introdução

Paradoxo do aniversário: quantas pessoas são necessárias para que a probabilidade de que duas delas façam aniversário no mesmo dia (colisão) seja maior do que 50%? Prob. de colisão de aniversários para 2 pessoas:

$$p(\text{colisão}) = 0.008$$

Introdução

Paradoxo do aniversário: quantas pessoas são necessárias para que a probabilidade de que duas delas façam aniversário no mesmo dia (colisão) seja maior do que 50%? Prob. de colisão de aniversários para n pessoas:

$$p(\text{colisão}) = 1 - \left(\frac{365}{365} \times \frac{364}{365} \times \cdots \times \frac{365 - (n-1)}{365}\right)$$

Introdução

Paradoxo do aniversário: quantas pessoas são necessárias para que a probabilidade de que duas delas façam aniversário no mesmo dia (colisão) seja maior do que 50%? Prob. de colisão de aniversários para 23 pessoas:

$$p(\text{colisão}) = .538$$

Introdução

Paradoxo do aniversário: quantas pessoas são necessárias para que a probabilidade de que duas delas façam aniversário no mesmo dia (colisão) seja maior do que 50%? Prob. de colisão de aniversários para 366 pessoas:

$$p(\text{colisão}) = 1$$

Sumário

Famílias de Hash

De forma geral, existem dois tipos de algoritmos para resumos criptográficos:

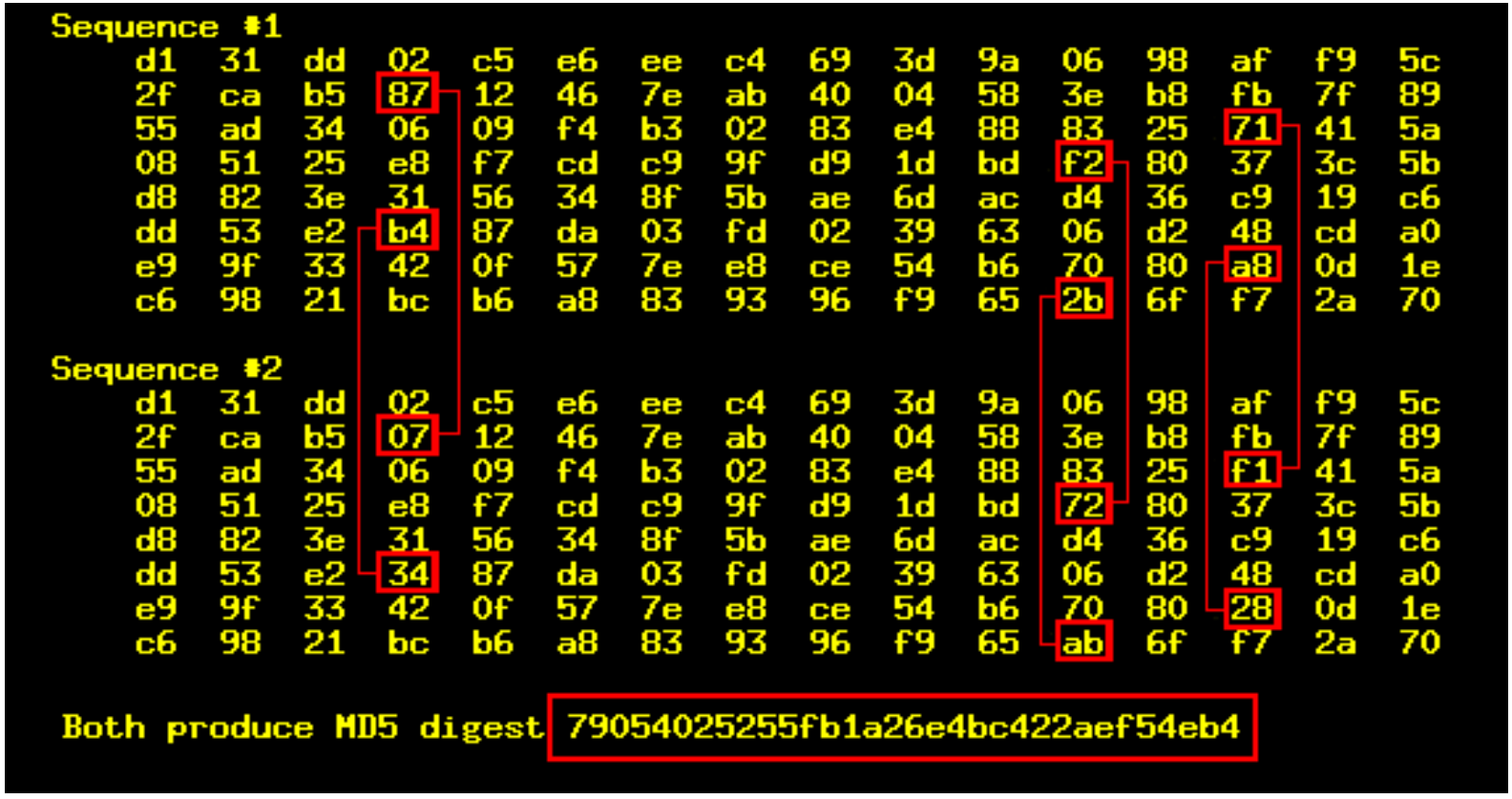
- funções hash dedicadas: que são especificamente projetadas para servir como funções hash;
- funções hash baseadas em cifras de bloco: cifras de bloco como o AES podem ser utilizadas para construir funções de hash.

Famílias de Hash

Muitas funções hash dedicadas são baseadas no que é conhecido como **família MD4** desenvolvidas por Ronald Rivest. Hashes MD4 são especialmente projetadas para alto desempenho em software (uso de variáveis de 32 bits e de operações como **xor**, **and**, **or** e **not**). Em 1991 foi criado o MD5, como melhoramento do MD4.

Famílias de Hash

O MD5 foi amplamente utilizado para segurança de protocolos da internet, checksum de arquivos e armazenamento de senhas. No entanto surgiram diversos ataques ao MD5:



Famílias de Hash

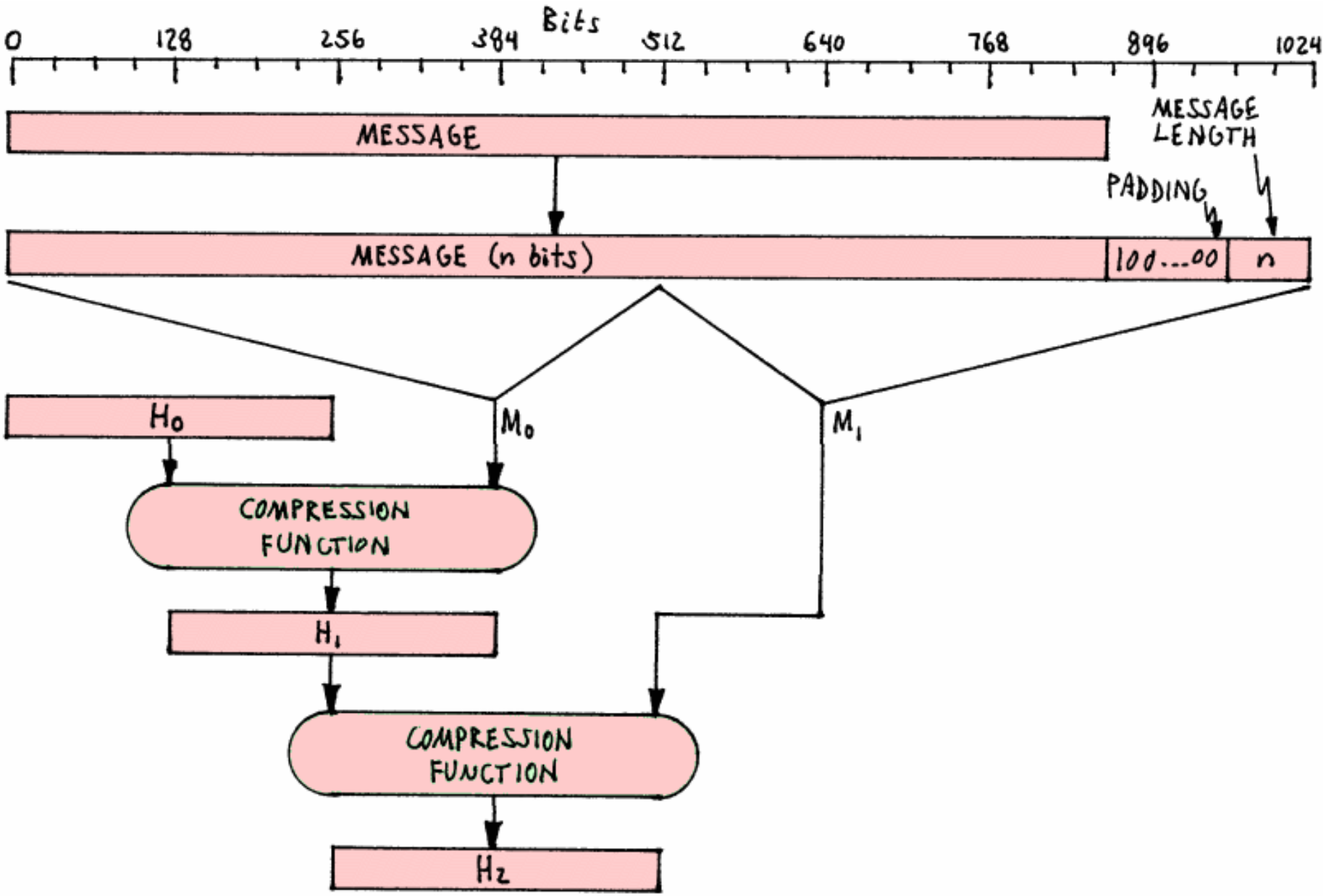
Em 1993 devido a diversos ataques ao MD5, a NSA desenvolveu a ideia do que é conhecido hoje como **SHA** (**secure hash algorithm**). Existem 4 algoritmos SHA: SHA-0, SHA-1, SHA-2 e SHA-3. O algoritmo SHA-2 que veremos em detalhes é implementada em algumas aplicações de segurança e protocolos amplamente usados, incluindo TLS e SSL, PGP, SSH, S/MIME, e IPsec.

Famílias de Hash

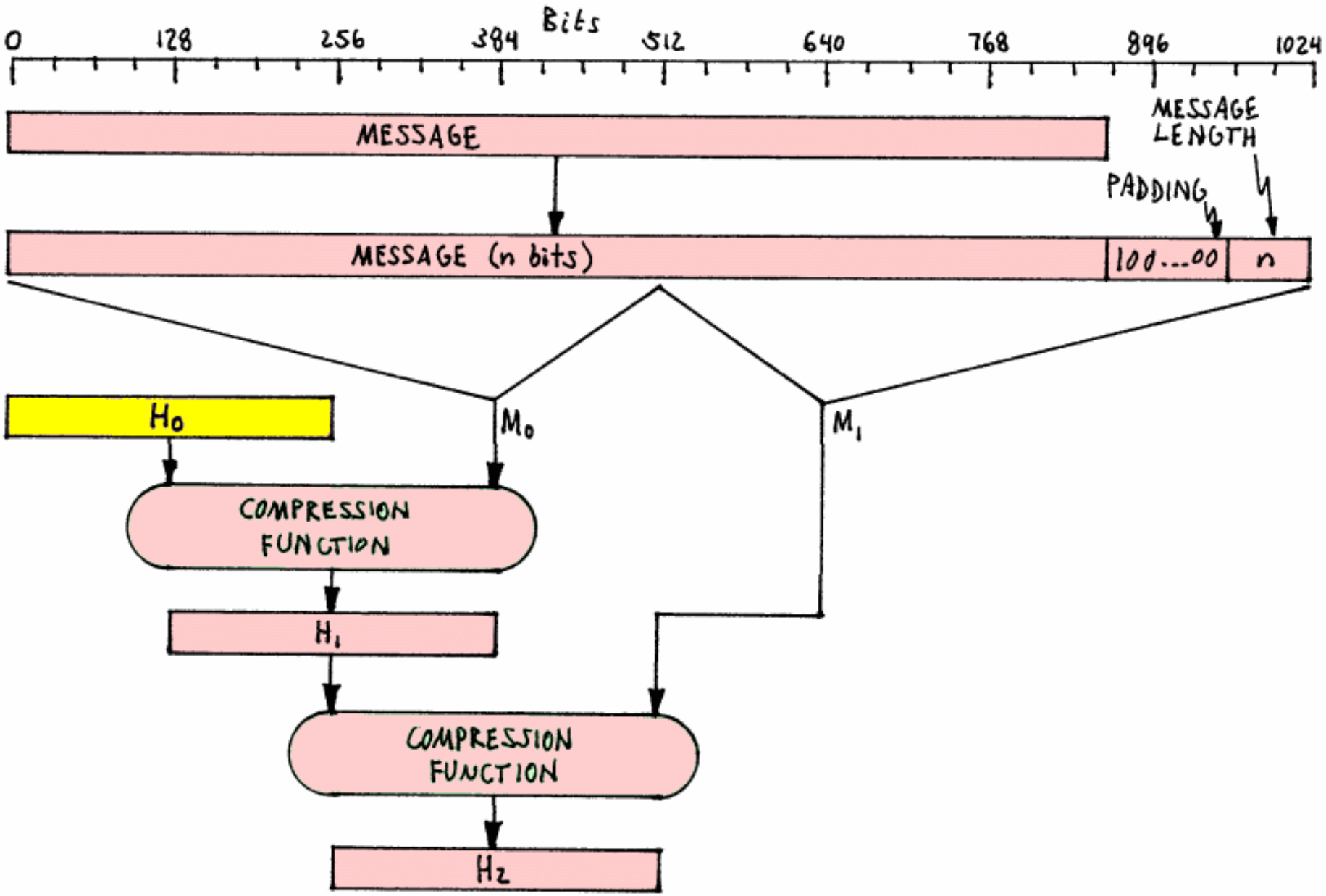
Algorithm and variant		Output size (bits)	Block size (bits)	Max message size (bits)	Collisions found
MD5 (as reference)		128	512	$2^{64} - 1$	Yes
SHA-0		160	512	$2^{64} - 1$	Yes
SHA-1		160	512	$2^{64} - 1$	Theoretical attack (2^{60})
SHA-2	*SHA-224* *SHA-256*	224 256	512	$2^{64} - 1$	None
	SHA-384 *SHA-512* *SHA-512/224* *SHA-512/256*	384 512 224 256	1024	$2^{128} - 1$	None
SHA-3		224/256 /384/512			None

Sumário

SHA-256


0
128
256
384
Bits
512
640
768
896
1024
MESSAGE
MESSAGE LENGTH
PADDING
MESSAGE (n bits)
100...00
n
H0
M0
M1
COMPRESSION FUNCTION
H1
COMPRESSION FUNCTION
H2

SHA-256


0
128
256
384
Bits
512
640
768
896
1024
MESSAGE
MESSAGE LENGTH
PADDING
MESSAGE (n bits)
100...00
n
H0
M0
M1
COMPRESSION FUNCTION
H1
COMPRESSION FUNCTION
H2

SHA-256

A função H_0 serve como inicialização, e retorna os seguintes valores fixos:

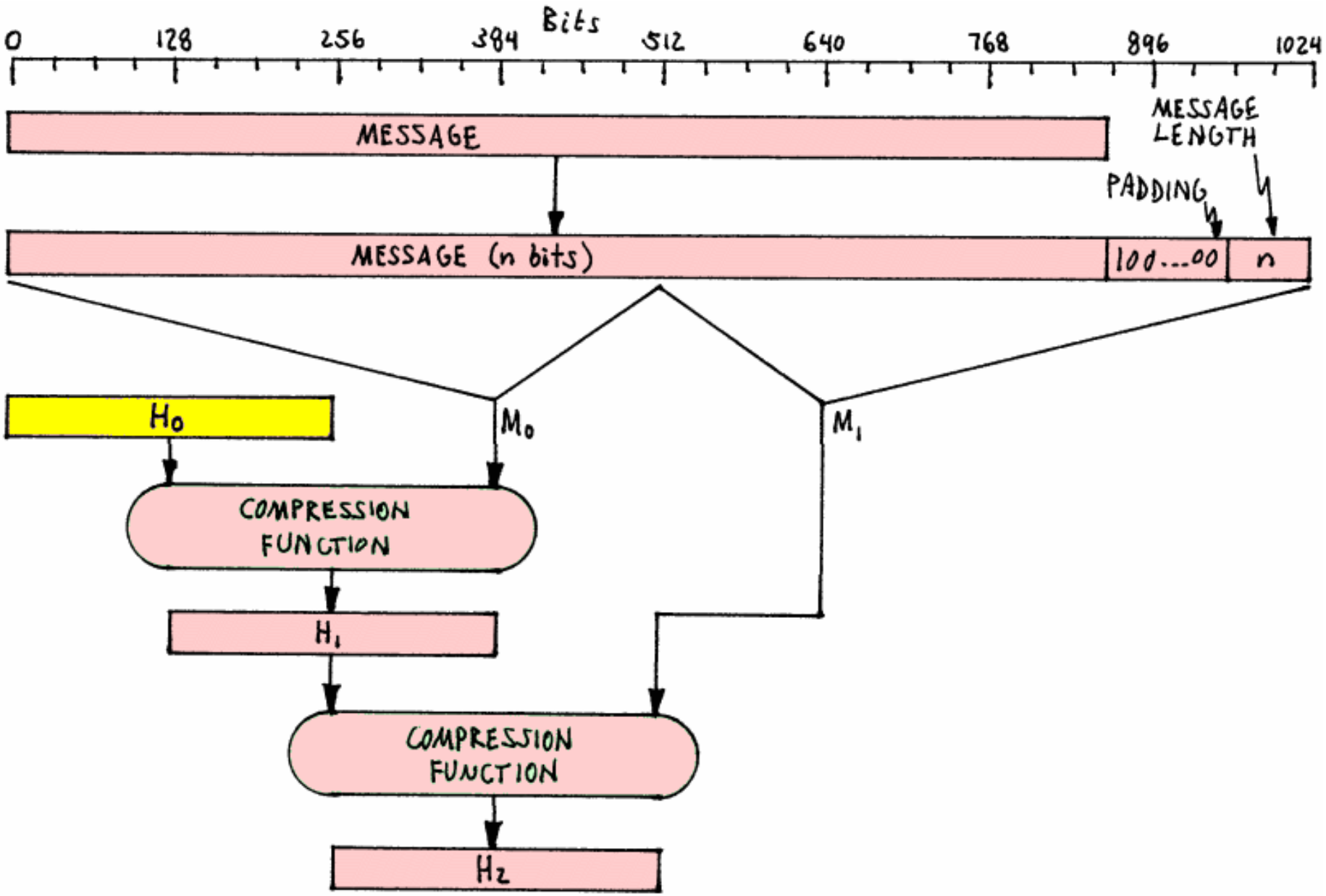
0x6a09e667	0xbb67ae85	0x3c6ef372	0xa54ff53a
0x510e527f	0x9b05688c	0x1f83d9ab	0x5be0cd19

ou seja, $8 \times 8 \times 4 = 256$ bits de informação (4 hexadecimais por número). Esses números correspondem aos 32 bits iniciais da parte fracionária dos 8 primeiros números primos (2, ..., 19). Exemplo: $\lfloor \mathrm{frac}(\sqrt{19}) \times 2^{64} \rfloor$.

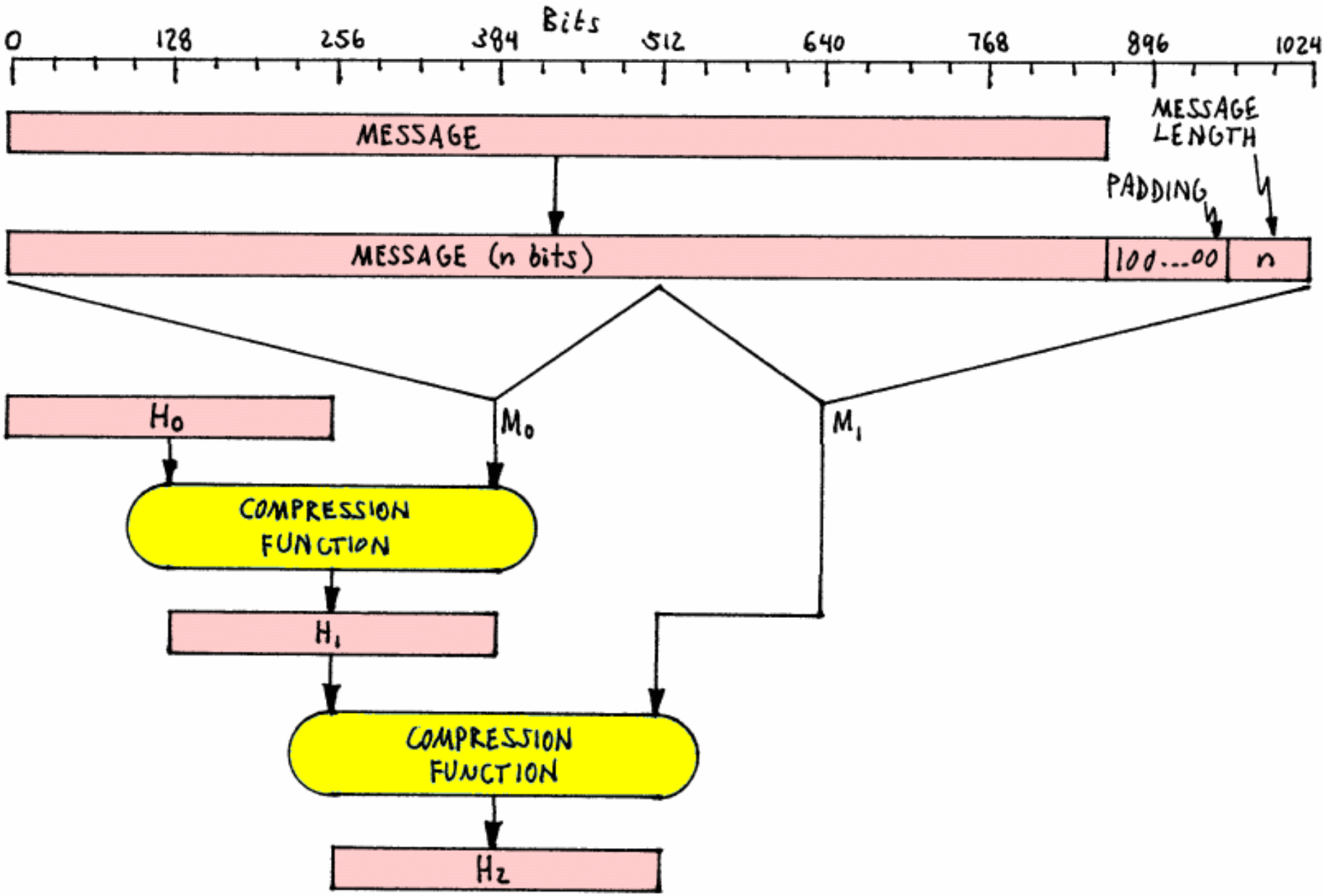
SHA-256

A razão pela qual o algoritmo SHA usa números primos é que sua raiz quadrada é sempre irracional, o que contribui para um número de aparência aleatória (o número 4 daria 0 o que é menos interessante). Embora esses números para inicialização não precisem ser nada em particular, os projetistas do SHA queriam que as pessoas soubessem que os números não foram escolhidos para introduzir furtivamente um backdoor no algoritmo (combinação de valores cuidadosamente escolhidos para permitir quebras da função de hash) o que seria muito difícil de confirmar ou refutar, pois exigiria meses, talvez anos de criptoanálise.

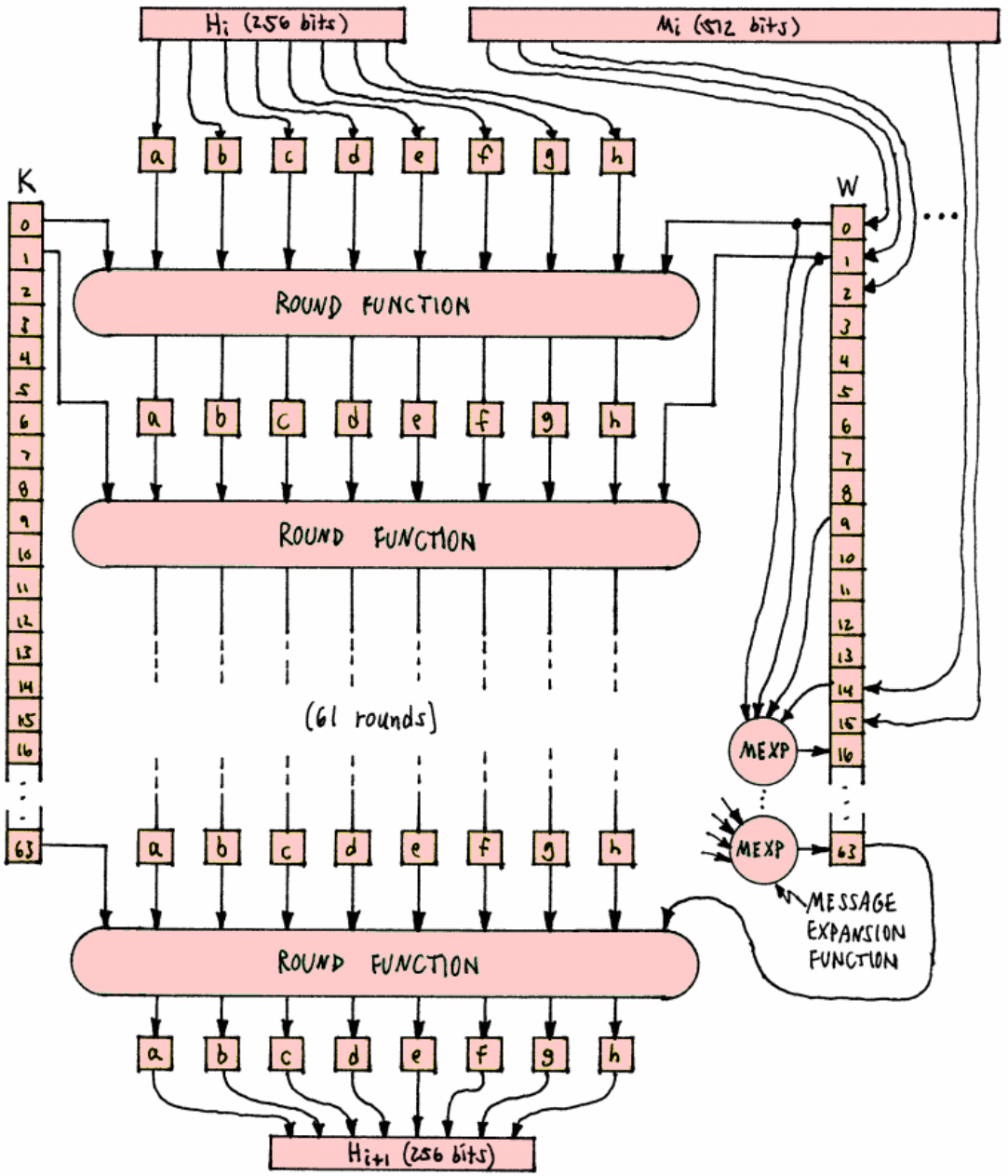
SHA-256


0
128
256
384
Bits
512
640
768
896
1024
MESSAGE
MESSAGE LENGTH
PADDING
MESSAGE (n bits)
100...00
n
H0
M0
M1
COMPRESSION FUNCTION
H1
COMPRESSION FUNCTION
H2

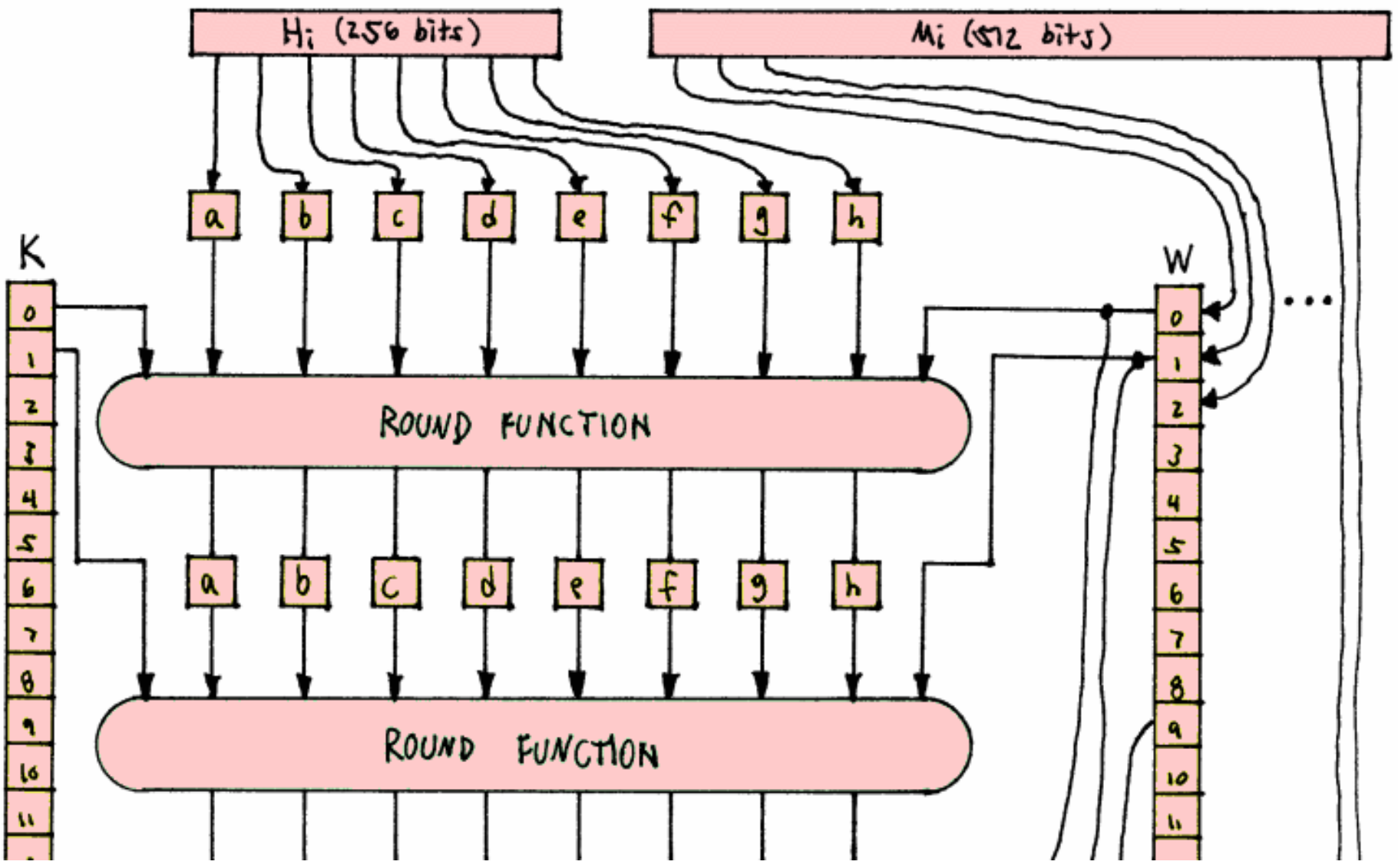
SHA-256


0
128
256
384
Bits
512
640
768
896
1024
MESSAGE
MESSAGE LENGTH
PADDING
MESSAGE (n bits)
100...00
n
H0
M0
M1
COMPRESSION FUNCTION
H1
COMPRESSION FUNCTION
H2

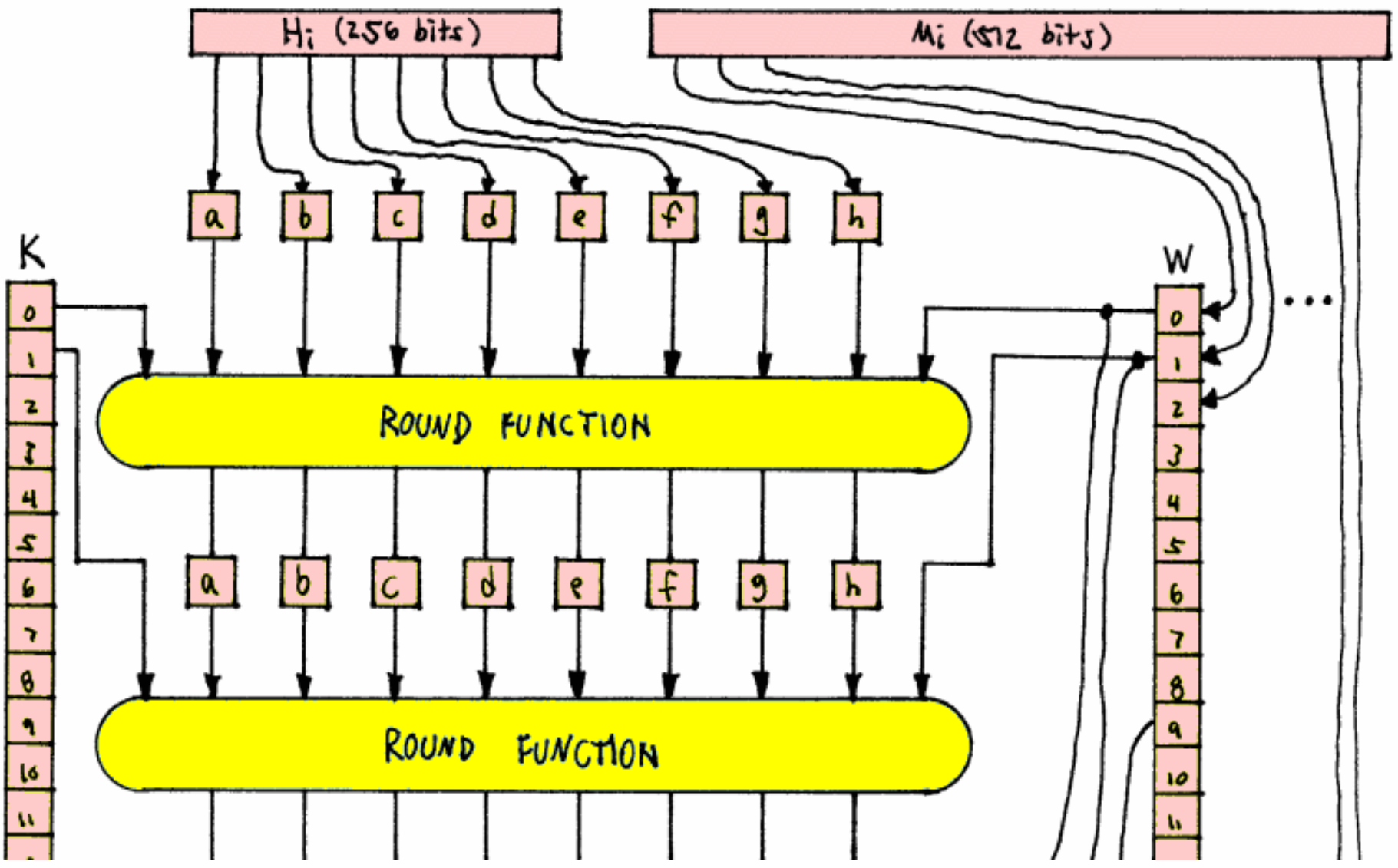
SHA-256 - Compression function



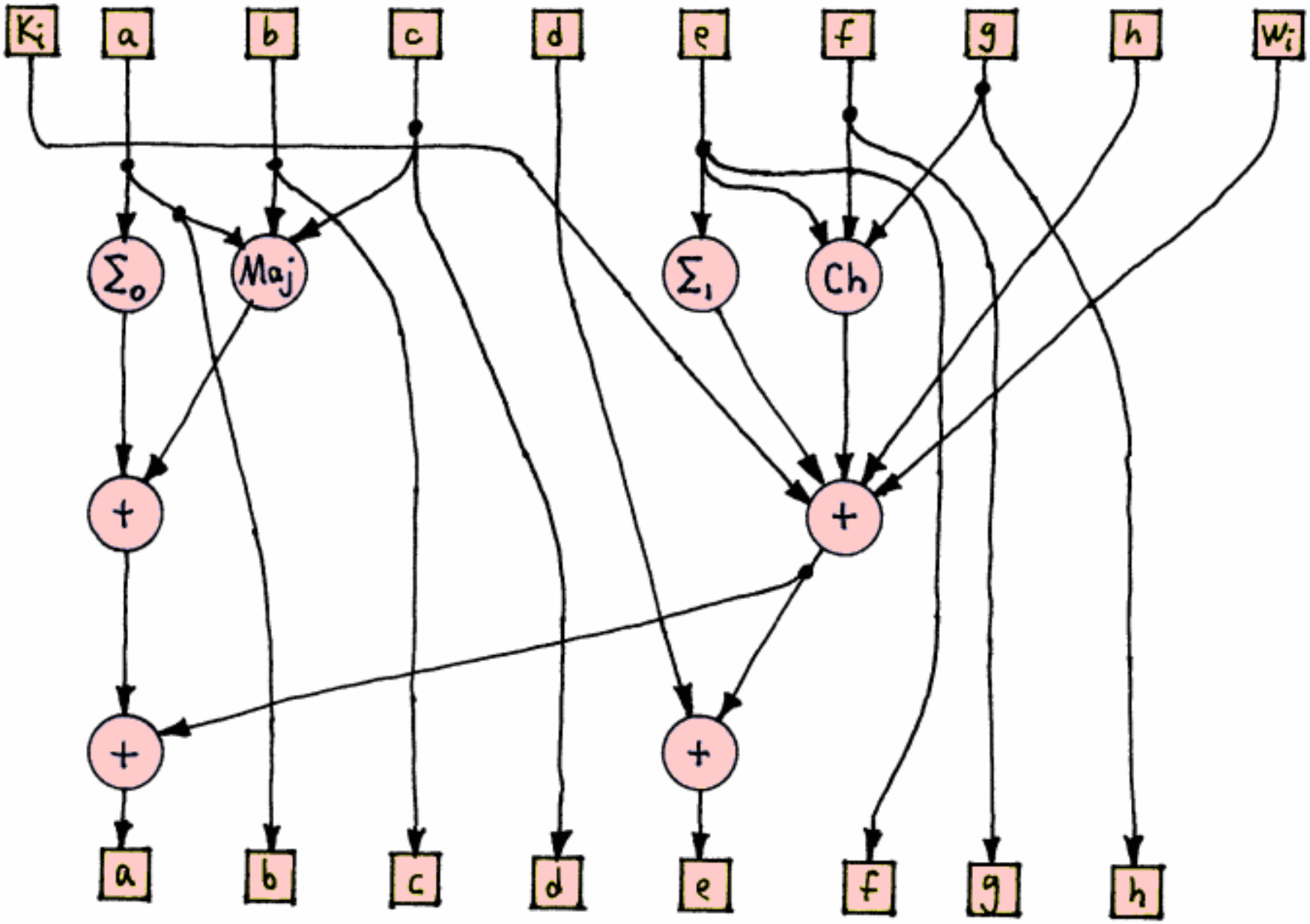
SHA-256 - Compression function



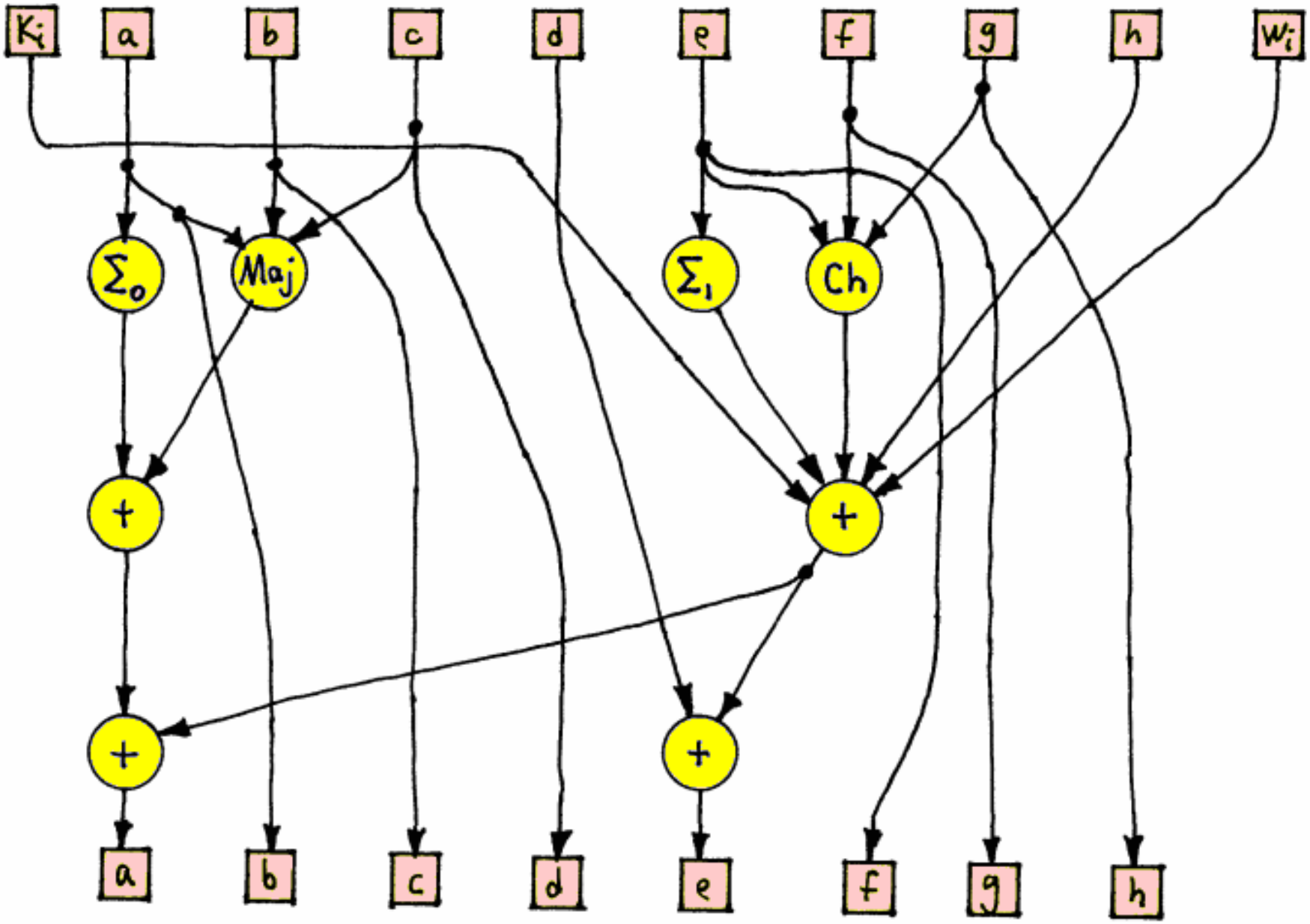
SHA-256 - Compression function



SHA-256 - Round function



SHA-256 - Round function



Introdução

Funções utilizadas no round function:

$$Ch(x, y, z) = (x \cap y) \oplus (\bar{x} \cap z)$$

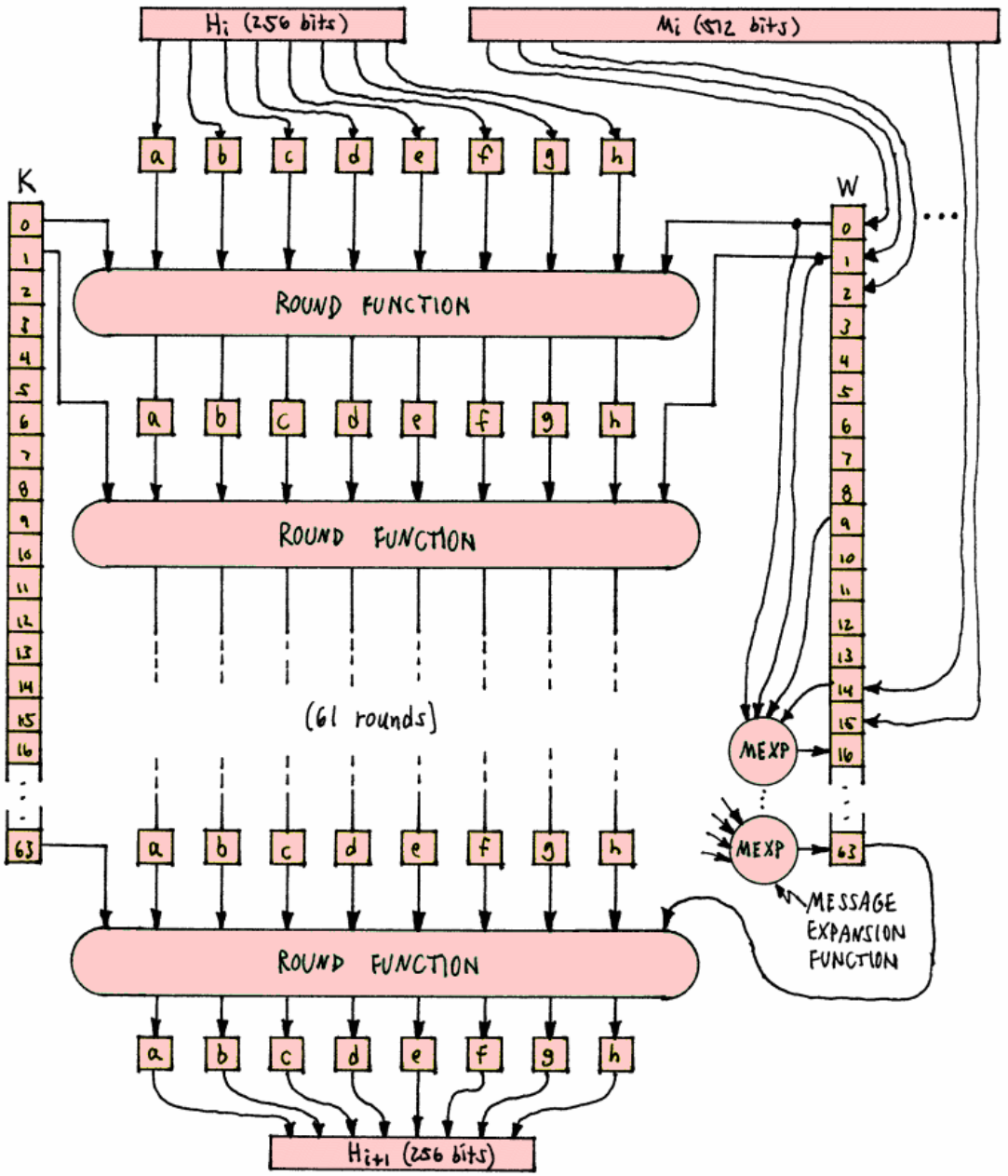
$$Maj(x, y, z) = (x \cap y) \oplus (x \cap z) \oplus (y \cap z)$$

$$\sum_{0}(x) = (x \gg 2) \oplus (x \gg 13) \oplus (x \gg 22)$$

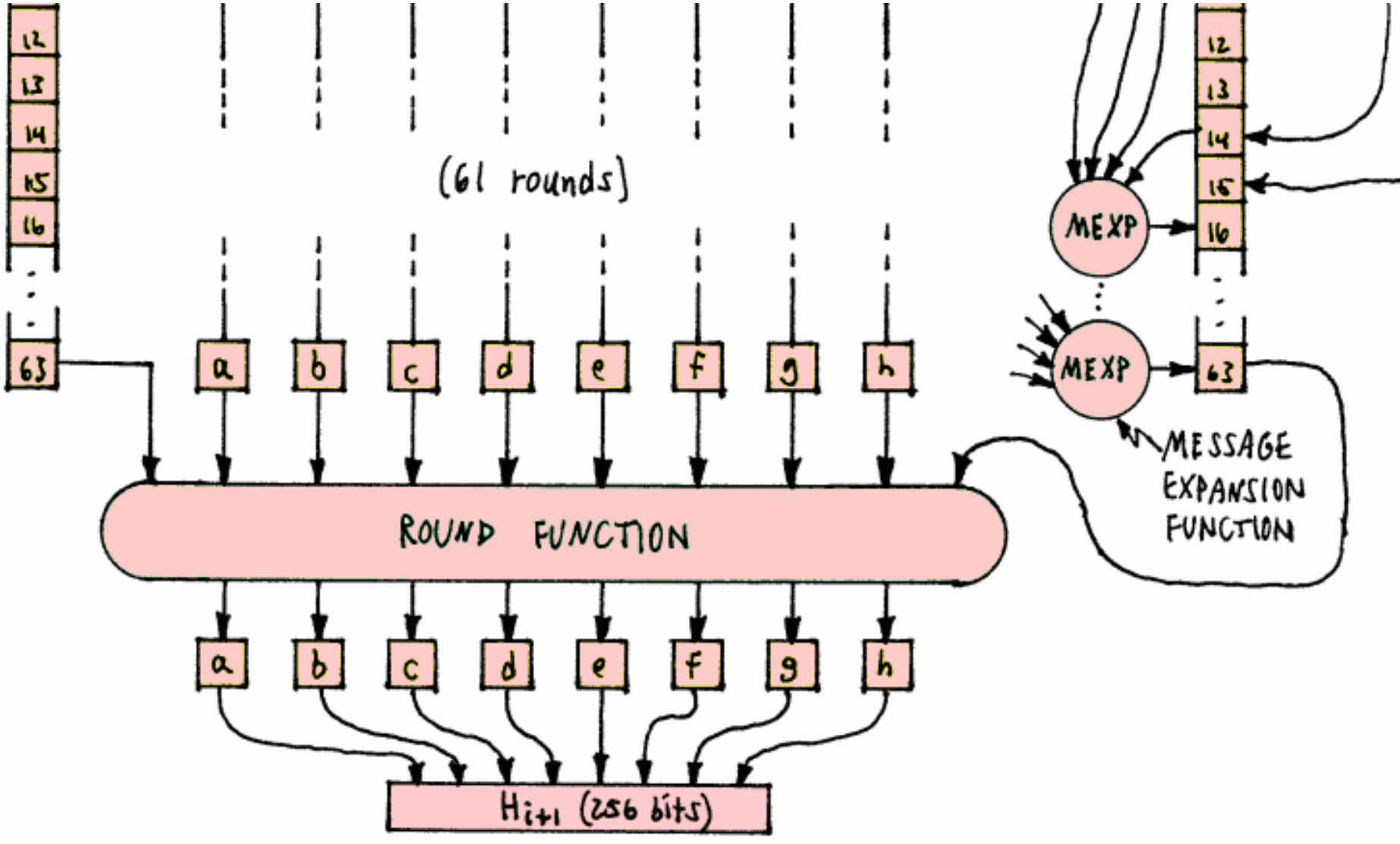
$$\sum_{1}(x) = (x \gg 6) \oplus (x \gg 11) \oplus (x \gg 25)$$

$x + y$: adição inteira módulo 2^{32}

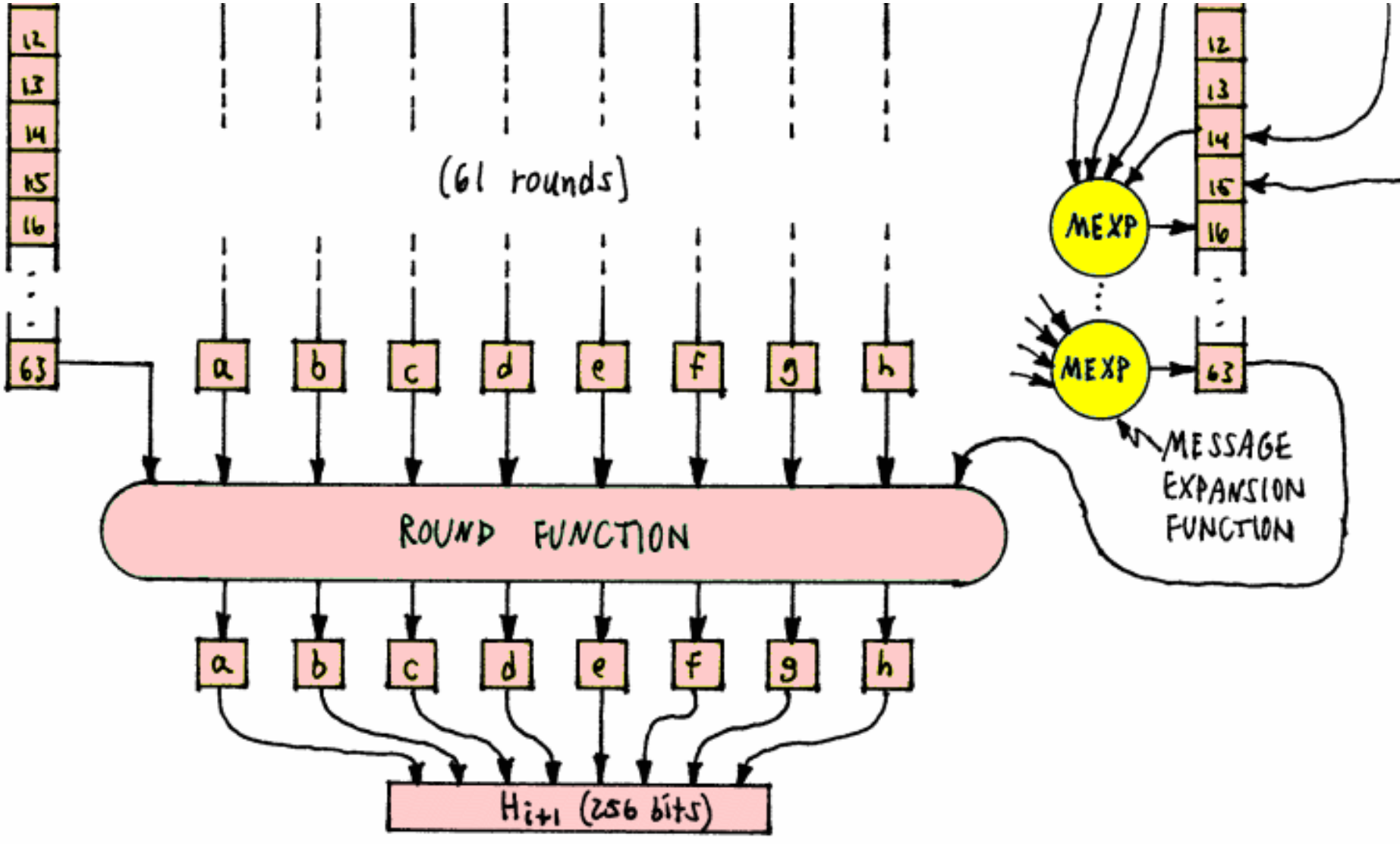
SHA-256 - Compression function



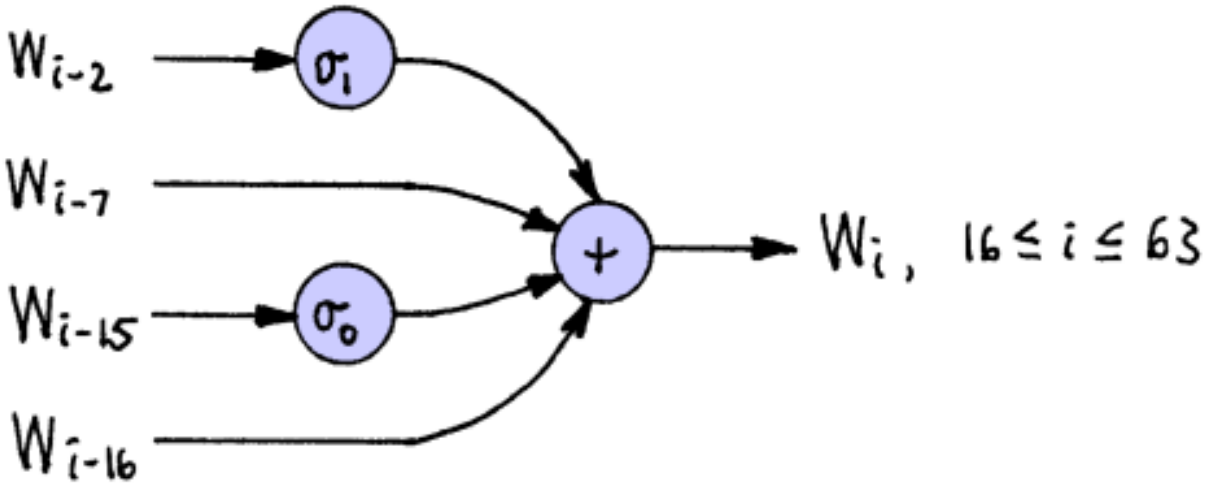
SHA-256 - Compression function



SHA-256 - Compression function

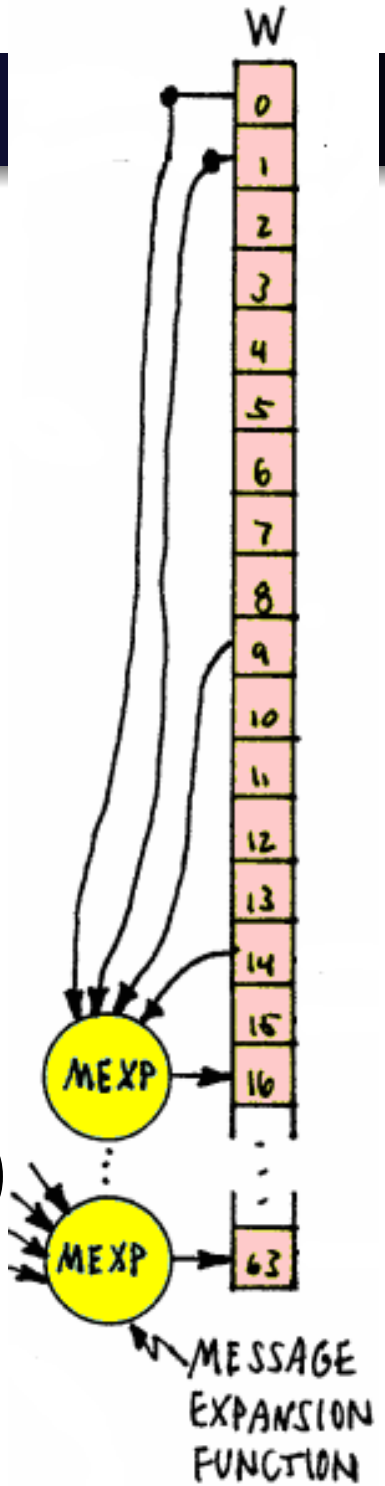


SHA-256 - Expansion function

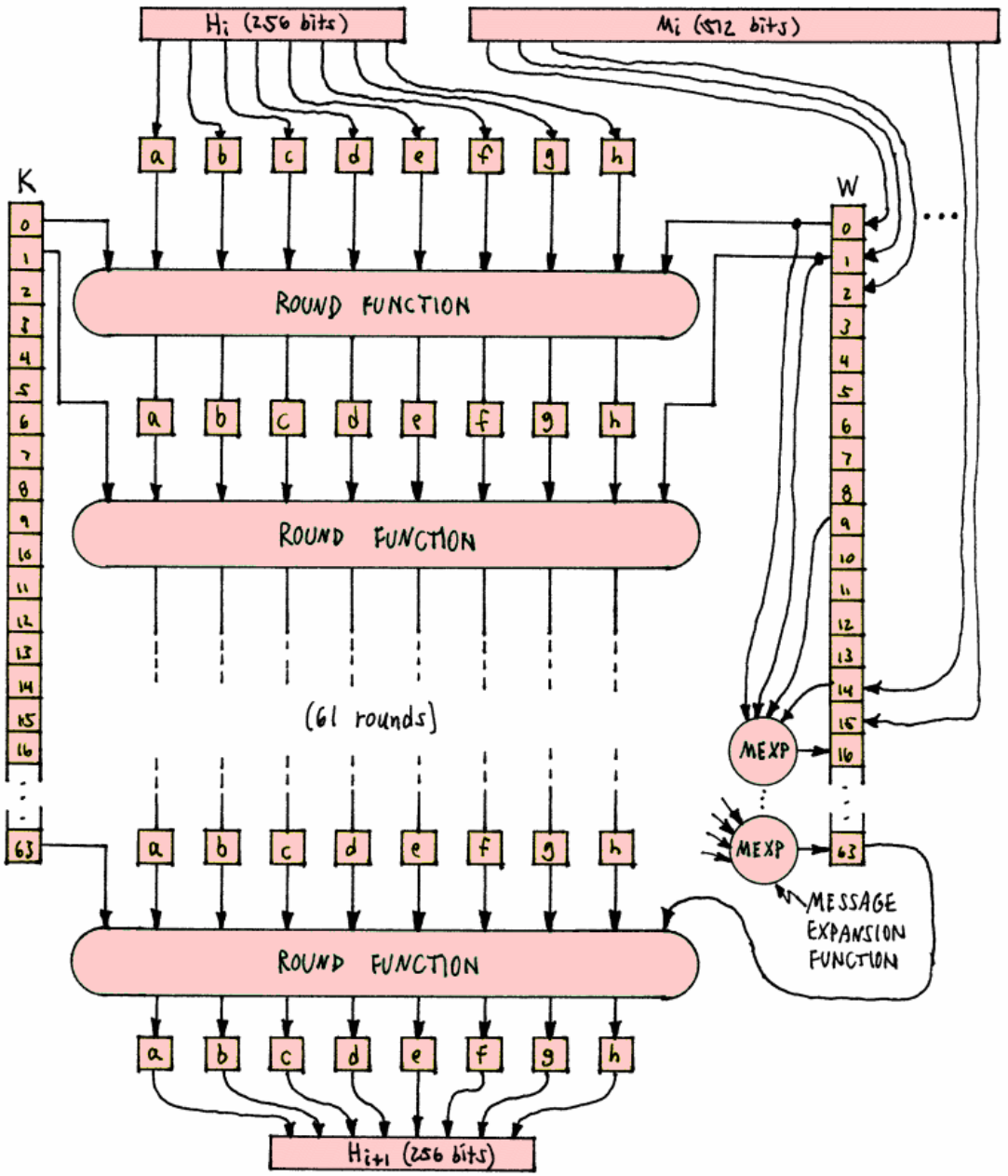


$$\sigma_0(x) = (x \gg 7) \oplus (x \gg 18) \oplus (x \gg 3)$$

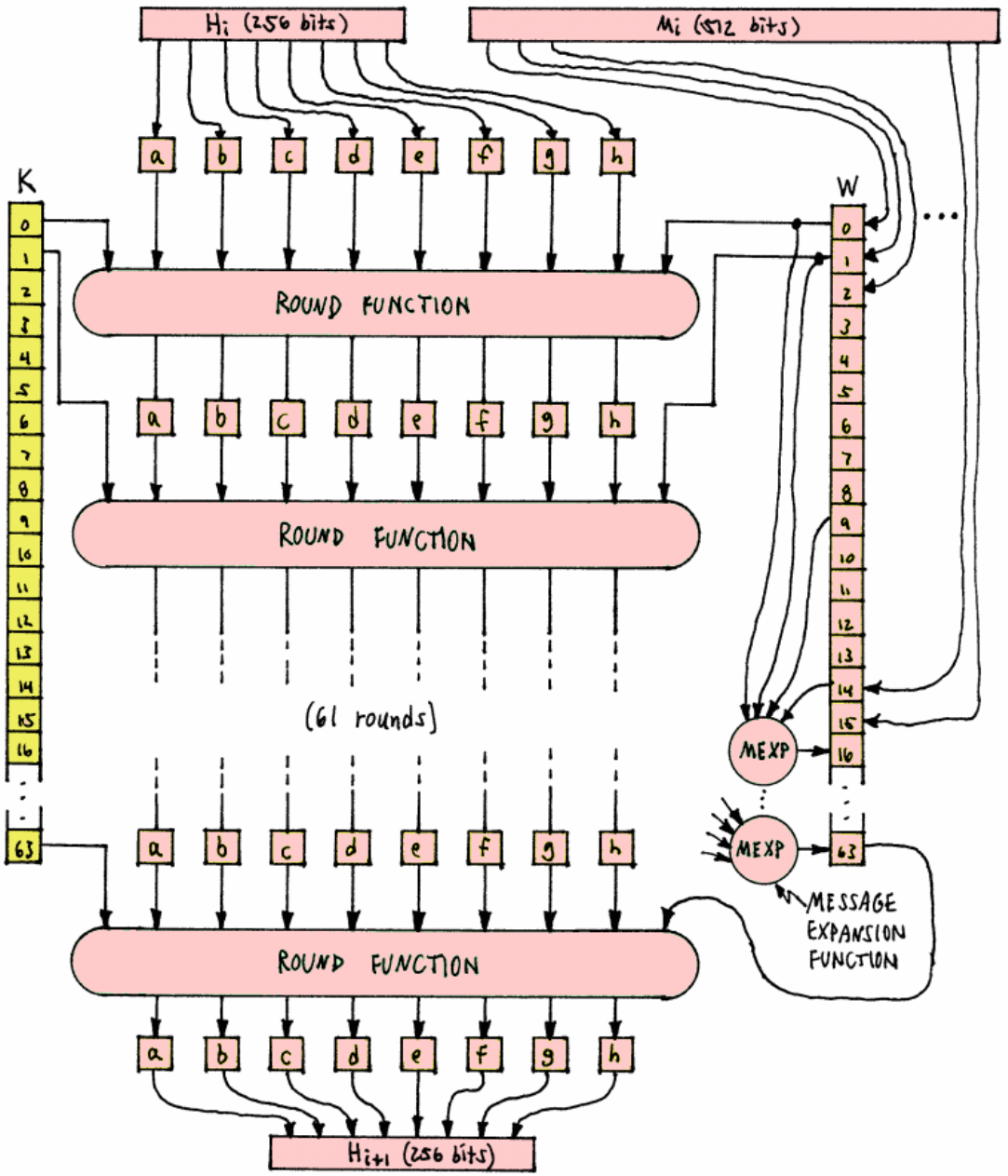
$$\sigma_1(x) = (x \gg 17) \oplus (x \gg 19) \oplus (x \gg 10)$$



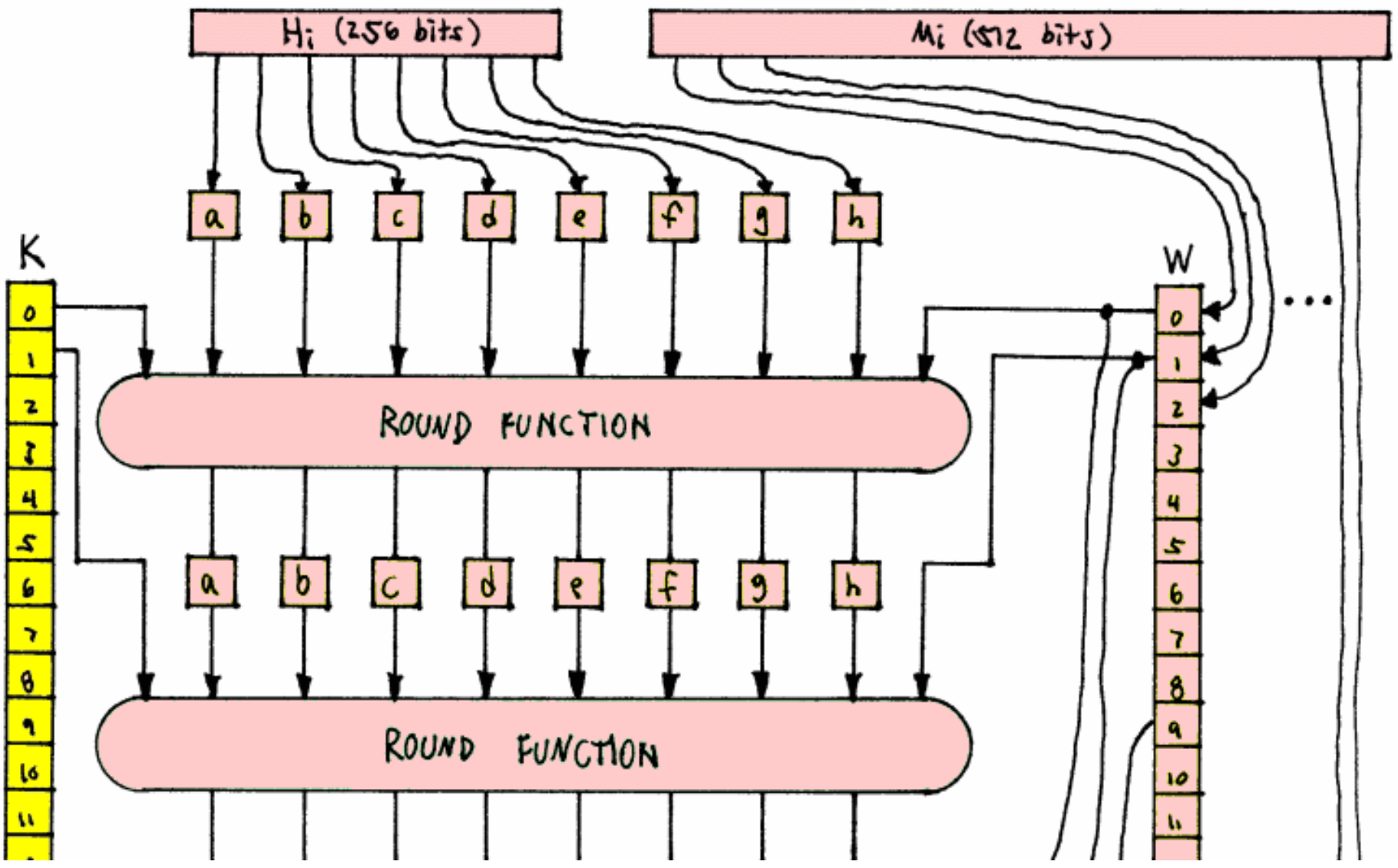
SHA-256 - Compression function



SHA-256 - Compression function



SHA-256 - Compression function



SHA-256 - Valores K

Constantes referentes a parte fracionária de 64 números primos (2, . . . , 311).

0x428a2f98	0x71374491	0xb5c0fbcf	0xe9b5dba5
0xd807aa98	0x12835b01	0x243185be	0x550c7dc3
0xe49b69c1	0xefbe4786	0x0fc19dc6	0x240ca1cc
0x983e5152	0xa831c66d	0xb00327c8	0xbf597fc7
0x27b70a85	0x2e1b2138	0x4d2c6dfc	0x53380d13
0xa2bfe8a1	0xa81a664b	0xc24b8b70	0xc76c51a3
0x19a4c116	0x1e376c08	0x2748774c	0x34b0bcb5
0x748f82ee	0x78a5636f	0x84c87814	0x8cc70208
0x3956c25b	0x59f111f1	0x923f82a4	0xab1c5ed5
0x72be5d74	0x80deb1fe	0x9bdc06a7	0xc19bf174
0x2de92c6f	0x4a7484aa	0x5cb0a9dc	0x76f988da
0xc6e00bf3	0xd5a79147	0x06ca6351	0x14292967
0x650a7354	0x766a0abb	0x81c2c92e	0x92722c85
0xd192e819	0xd6990624	0xf40e3585	0x106aa070
0x391c0cb3	0x4ed8aa4a	0x5b9cca4f	0x682e6ff3
0x90befffa	0xa4506ceb	0xbef9a3f7	0xc67178f2

SHA-256 - Assinatura

Aplicação: assinatura digital