GBC083 – Segurança da Informação
Aula 5 - Algoritmos de funções hash

Colisões em funções hash

- $p(n, d)$ é probabilidade de achar colisão para n hashes em espaço de d possíveis.
- $\hat{p}(n, d) = 1 - p(n, d)$
- $\hat{p}(n, d) = (1 - 1/d)(1 - 2/d) \ldots (1 - (n-1)/d)$
- Série de Taylor: $e^x = 1 + x + x^2/2! + \ldots$
- Se $a = x$ e $a << 1$, então $e^{-a/d} \approx 1 - a/d$
- $\hat{p}(n, d) \approx 1 \times e^{-1/d} \times e^{-2/d} \times \ldots e^{-(n-1)/d}$
- $\hat{p}(n, d) \approx e^{-\frac{(n-1)n}{2d}}$
- Então $p(n, d) \approx 1 - e^{-\frac{(n-1)n}{2d}}$
- Para achar o menor n' tal que $p(n', d) \geq 1/2$, queremos um limitante superior em $\hat{p}(n', d)$ ou seja quando temos a desigualdade $e^{-\frac{(n'-1)n'}{2d}} < 1/2$.
- Resolvendo essa desigualdade, $\log(e^{-\frac{(n'-1)n'}{2d}}) < \log(1/2)$
- $-\frac{(n'-1)n'}{2d} < -\log(2)$
- $-(n'-1)n' < -\log(2)2d$
- $(n'-1)n' > \log(2)2d$
- $(n')^2 > \log(2)2d$
- $n' > \sqrt{\log(2)2d}$.

Colisões em funções hash

- Para $2n$ bits de saída de funções hash, atacante tem que realizar aproximadamente $\sqrt{(2^{2n})}$ gerações de valores hash com entradas distintas para ter probabilidade 0.5 de sucesso em obter um colisão.
- De modo geral, sejam:
  - t número de tentativas de entradas distintas
  - $\lambda =$ probabilidade de 1 colisão$= 1 - P$(nenhuma colisão)
- temos

$$t \approx 2^{(n+1)/2} \sqrt{\log(\frac{1}{1-\lambda})}$$

- Exemplo
  - Se $\lambda = 0.5$, para $n = 128$, $t = 2^{65}$, para $n = 512$, $t = 2^{257}$
  - Se $\lambda = 0.9$, para $n = 128$, $t = 2^{67}$, para $n = 512$, $t = 2^{258}$

Algoritmos de funções Hash

- Entrada de tamanho variável e saída de tamanho fixo
- Ideia básica
  - Segmentar entrada em blocos de tamanhos iguais
  - Aplicar função de compressão de maneira iterada
- Tipos
  - Funções hash dedicadas
  - Funções hash baseadas em cifras de blocos

Família MD4

- Ronald Rivest desenvolveu *message digest 4* (MD4) (hoje abandonada devido a colisões)
  - MD5, Secure Hash Algorithm (SHA-0), SHA-1, SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512), RIPEMD

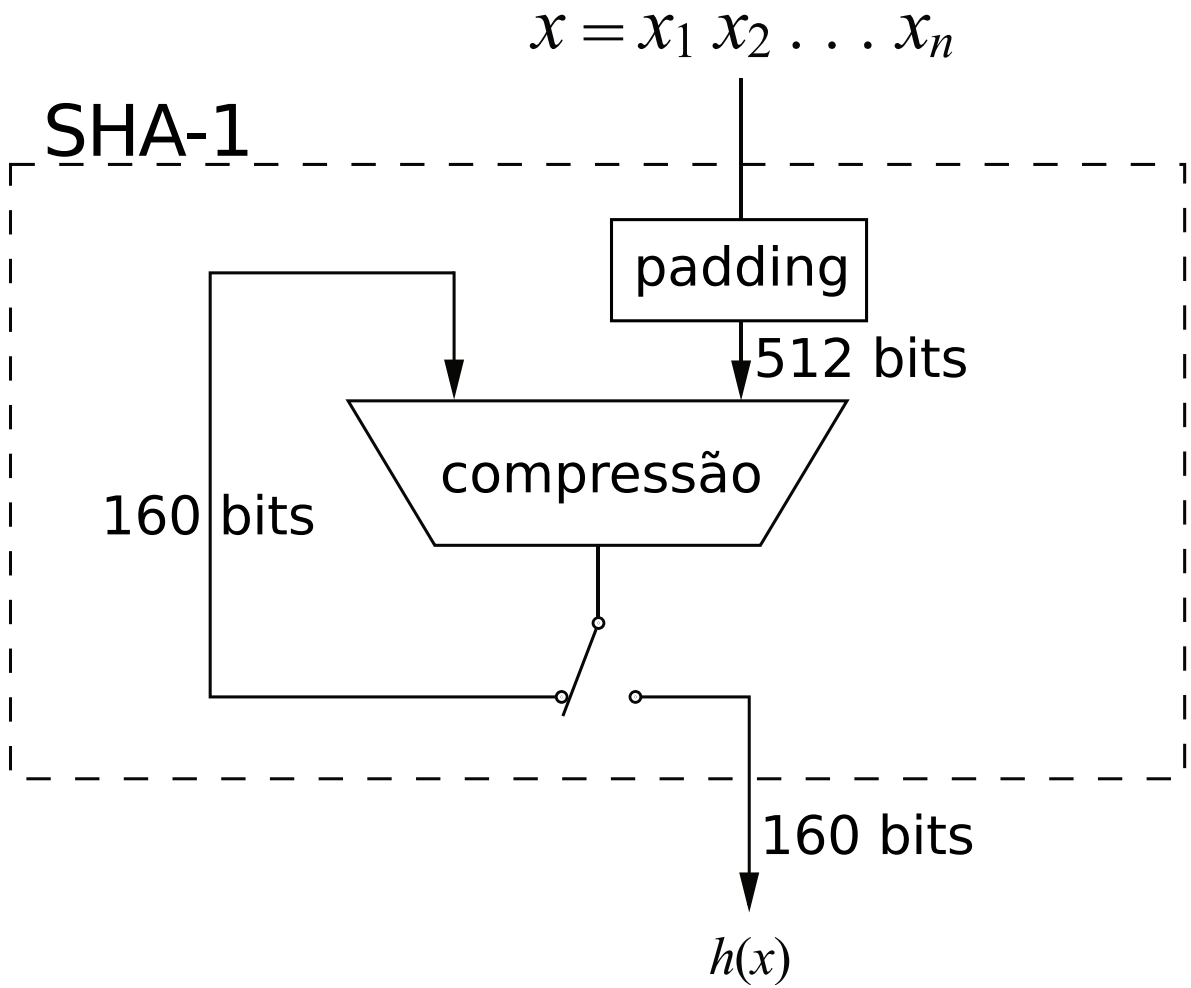
Table 11.2 The MD4 family of hash functions

Algorithm		Output [bit]	Input [bit]	No. of rounds	Collisions found
MD5		128	512	64	yes
SHA-1		160	512	80	not yet
SHA-2	**SHA-224**	224	512	64	no
	SHA-256	256	512	64	no
	SHA-384	384	1024	80	no
	SHA-512	512	1024	80	no

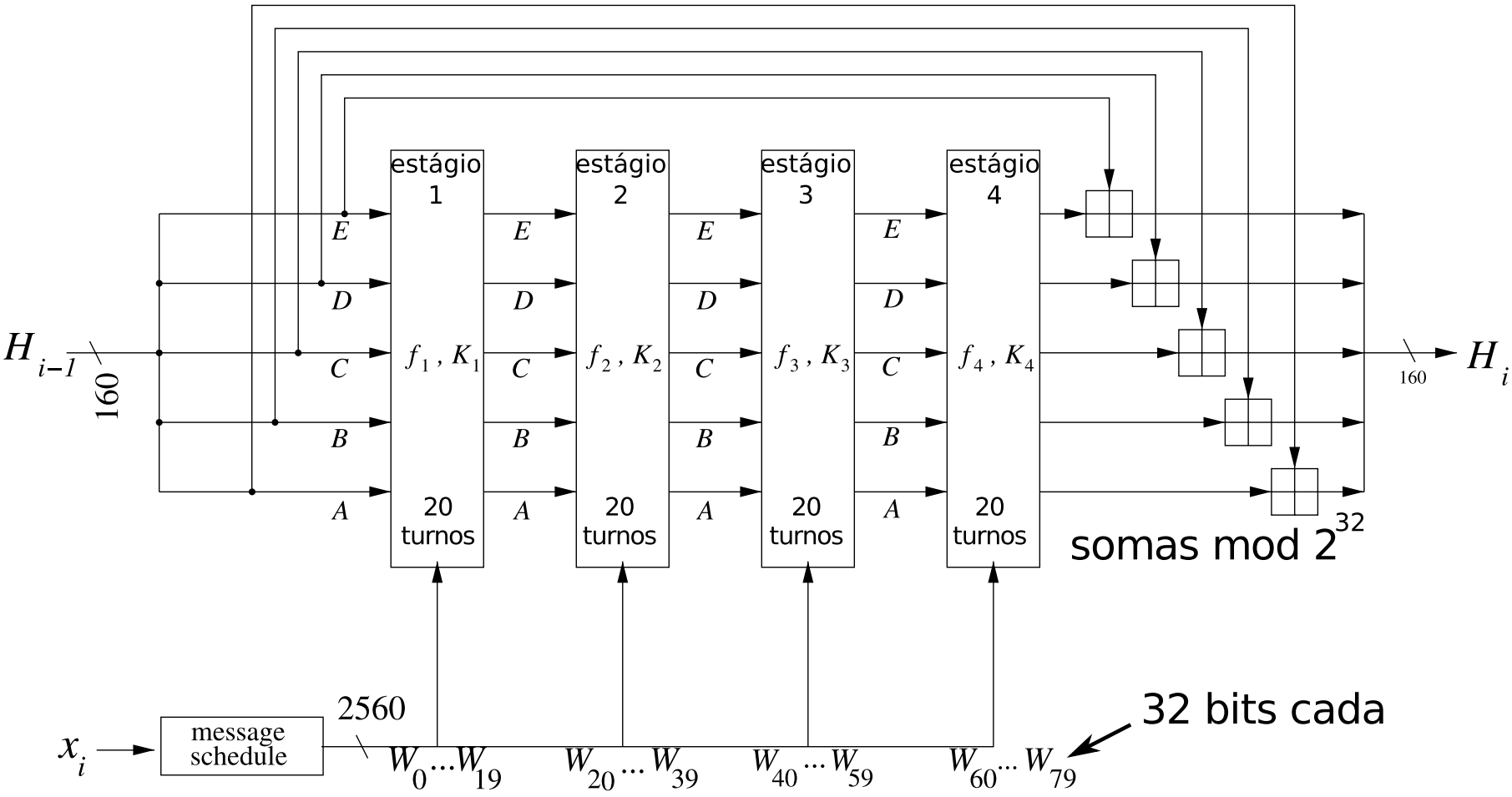
Secure Hash Algorithm (SHA-1)

- Função de compressão atua como uma cifra de blocos tipo Feistel
- 80 turnos de Merkle-Damgård
- 512 bits de processamento
- 160 bits de saída

Construção de Merkle-Damgård - SHA-1



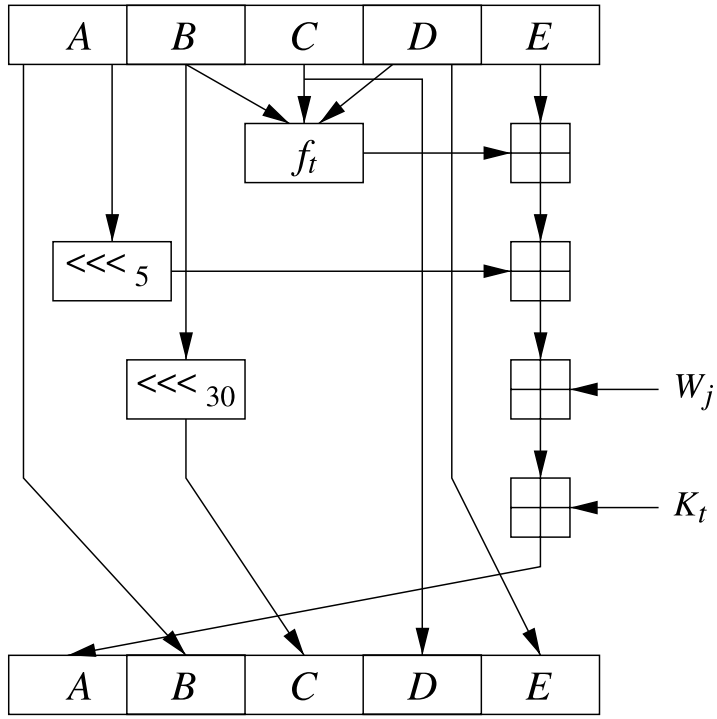
Detalhes SHA-1



- A,B,C,D,E tem 32 bits
- K_1, K_2, K_3, K_4 são constantes de estágio

Turno

$$A,B,C,D,E = (E + f_t(B,C,D) + (A)_{\lll 5} + W_j + K_t), A, (B)_{\lll 30}, C, D$$



- Parecido a uma rede de Feistel desbalanceada
- f_t é a função do turno

Funções e constantes de turnos do SHA

Estágio t	Turnos	Constante K_t	Função f_t
1	$0 \ldots 19$	$5A827999$	$f_1 = (B\&C)\|(!B\&D)$
2	$20 \ldots 39$	$6ED9EBA1$	$f_2 = B \oplus C \oplus D$
3	$40 \ldots 59$	$8F1BBCDC$	$f_3 = (B\&C)\|(B\&D)\|(C\&D)$
4	$60 \ldots 79$	$CA62C1D6$	$f_4 = B \oplus C \oplus D$

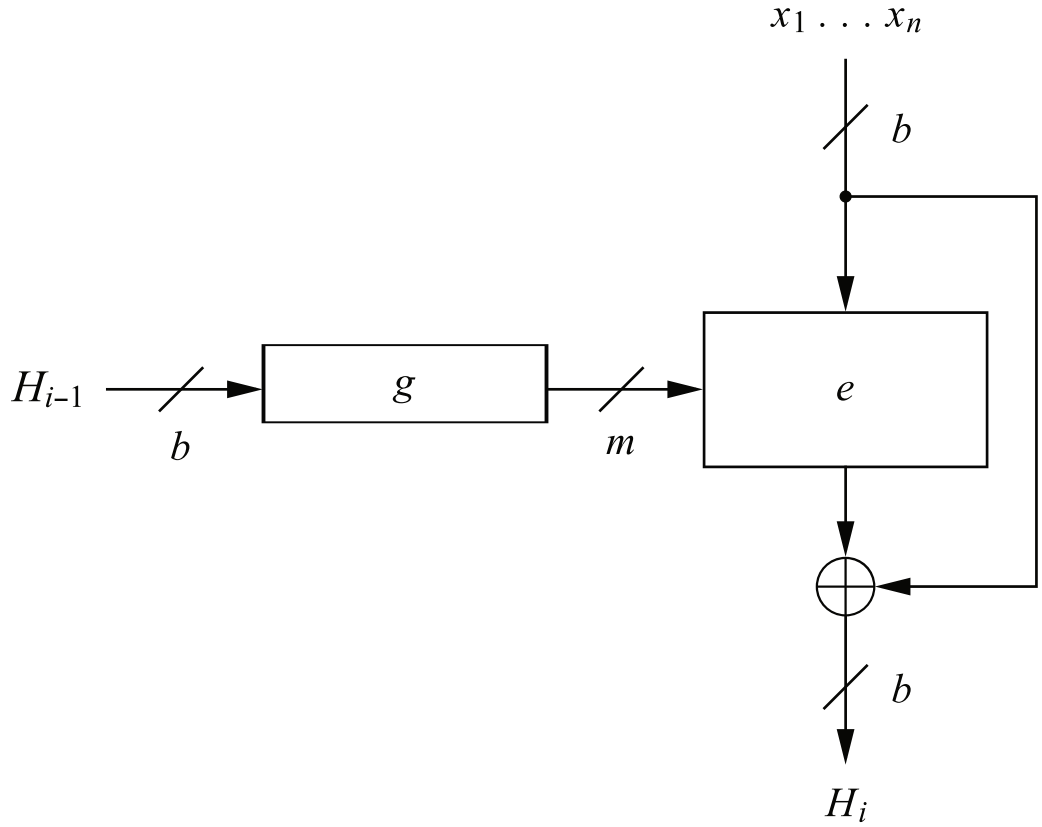
Message schedule

- Preparar entrada de 512 bits: $x_i = (x_i^{(0)} x_i^{(1)} \dots x_i^{(15)})$ em palavras w
- Para $0 \leq j \leq 15$, copiar $w_j = x_i^{(j)}$
- Para $16 \leq j \leq 79$, fazer $w_j = w_{j-16} \oplus w_{j-14} \oplus w_{j-8} \oplus w_{j-3}$

Inicialização

- Aplicar padding para obter mensagem de tamanho múltiplo de 512
  - Passo 1: Concatenar um único bit com valor 1
  - Passo 2: Concatenar k zeros para mensagem de tamanho l: $k = 448 - (l + 1) \mod 512$
  - Passo 3: Concatenar 64 bits contendo o valor l
- Para cada bloco, usar computação de Hash prévia H_{i-1} e seus estados A, B, C, D, E
- Necessário inicializar para H_0:
  - $A = H_0^0 = 67452301$
  - $B = H_0^1 = EFCDAB89$
  - $C = H_0^2 = 98BADCFE$
  - $D = H_0^3 = 10325476$
  - $E = H_0^4 = C3D2E1F0$

Funções hash a partir de cifras de blocos



- Função hash de Matya-Meyer-Oseas

$$H_i = e_{g(H_{i-1})}(x_i) \oplus x_i$$

- $e_{g(H_{i-1})}$ é cifra chaveada com $g(H_{i-1})$
- $g(\cdot)$ mapeia o tamanho da saída da função hash para o tamanho do bloco da cifra

Função de compressão Davies-Meyer (caso específico de Matya-Meyer-Oseas)

- Dada uma cifra de blocos tal que $k \times \{0,1\}^n \rightarrow \{0,1\}^n$
- Função de Davies-Meyer

$$h(H, m) = H_i = E(m, H_{i-1}) \oplus H_{i-1}$$

, onde

  - m atua como a chave
  - H é o hash da parte anterior da mensagem

- Se E for uma cifra ideal (coleção de $|K|$ permutações), achar uma colisão custa $O(2^{n/2})$ avaliações

Outras hashes

- SHA-256
  - Cifra de blocos: SHACAL-2
  - Tipo Davies-Meyer: $h(H, m) = E(m, H) \oplus H$
  - Bloco de mensagem (chave) 512 bits
  - Tamanho saída de hash de 256 bits
- SHA-3 / Keccak
  - Criado em competição pública de 2008 a 2012 – Bertoni, Daemen (AES), Peeters, Assche
  - Projeto diferente de outros SHA - "construção esponja"
  - Usar 224, 256, 384, 512 bits
- Whirlpool
  - Vincent Rijmen (AES) e Paulo Barreto (USP)
  - Miyauchi-Preneel: $h(H, m) = E(m, H) \oplus H \oplus m$
  - 512 bits, inpirado no AES
  - Competição NESSIE (Europa)
  - Usado por TrueCrypt

Funções de compressão comprovadas

- Existem funções de compressão que tem segurança comprovada
- Exemplo:
  - Seja p um número primo de 2000 bits e números aleatórios $1 \leq u, v \leq p$
  - Para $m, h \in \{0, \ldots, p-1\}$, definir função de compressão como

$$h(H, m) = u^H \times v^m \mod p$$

  - Encontrar colisão para essa função é um problema difícil (equivale a resolver problema de log-discreto $\mod p$)
- De modo geral, essas funções são lentas