



Bahan Kuliah II4031 Kriptografi dan Koding

# *Secure Hash Algorithm (SHA)*

Oleh: Rinaldi Munir

**Program Studi Sistem dan Teknologi Informasi  
Sekolah Teknik Elektro dan Informatika(STEI) - ITB  
2023**

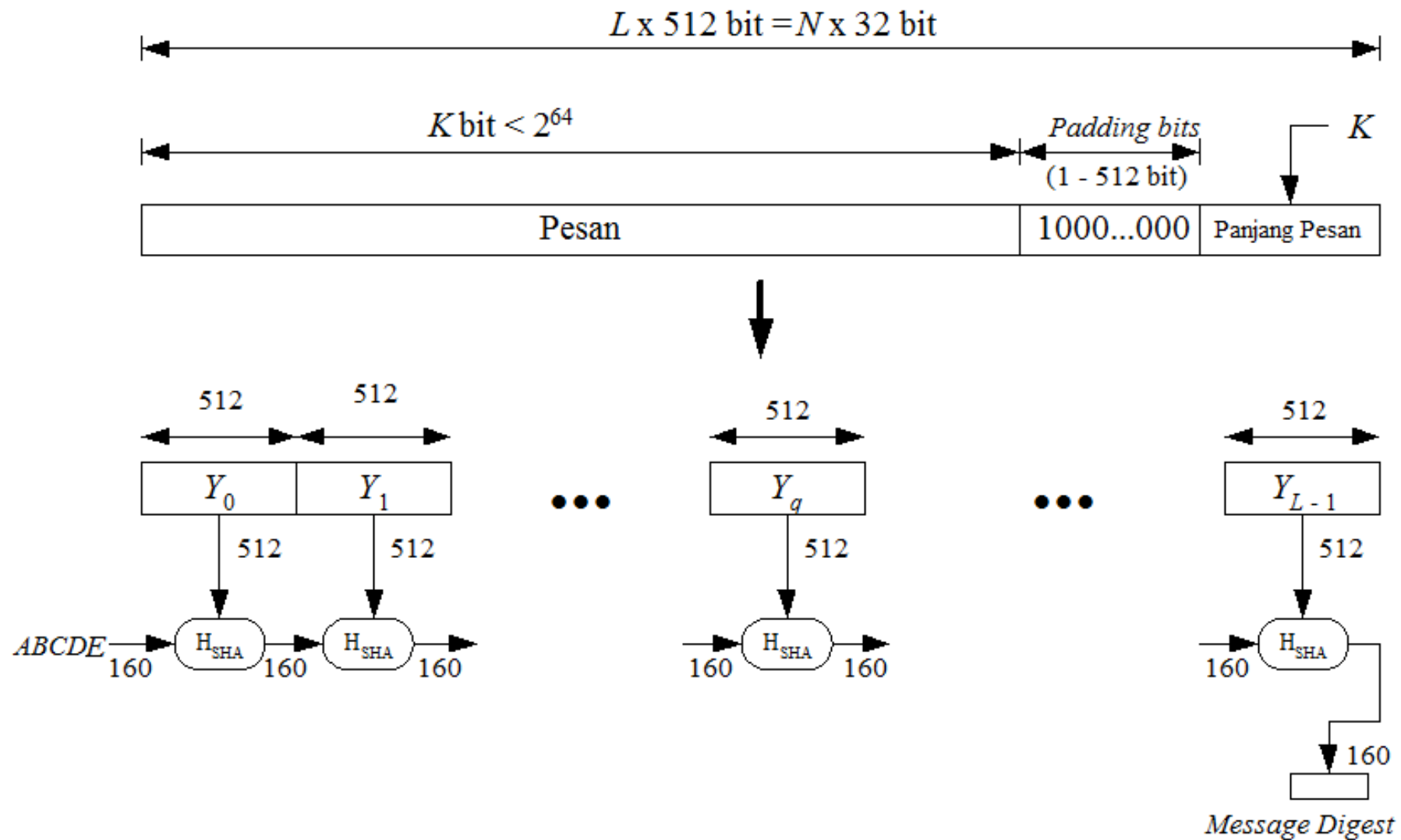
# *Secure Hash Algorithm (SHA)*

- *SHA* adalah fungsi *hash* satu-arah yang dibuat oleh *NIST* dan digunakan bersama *DSS (Digital Signature Standard)*.
- Oleh *NSA*, *SHA* dinyatakan sebagai standard fungsi *hash* satu-arah.
- *SHA* didasarkan pada *MD4* yang dibuat oleh Ronald L. Rivest dari *MIT*.
- Algoritma *SHA* menerima masukan berukuran maksimum  $2^{64}$  bit (2.147.483.648 *gigabyte*) dan menghasilkan *message digest* yang panjangnya 160 bit.
- *Message digest* dari *SHA* lebih panjang dari *message digest* yang dihasilkan oleh *MD5* (128 bit).

- Sebutan *SHA* mengacu pada keluarga fungsi *hash* satu-arah SHA.
- Enam varian *SHA*:
  - SHA-0
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- SHA-0 sering diacu sebagai *SHA* saja
- Yang akan dibahas: SHA-1
- Detil algoritma SHA-1 dapat dibaca di dalam dokumen RFC 3174 (<http://www.faqs.org/rfcs/rfc3174.html>)

		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Word size (bits)	Rounds	Operations	<u>Collisions</u> found?
<b>SHA-0</b>									Yes
<b>SHA-1</b>		160	160	512	$2^{64} - 1$	32	80	+,and,or, xor,rot	Yes
<b>SHA-2</b>	<i>SHA-256/224</i>	256/224	256	512	$2^{64} - 1$	32	64	+,and,or, xor,shr,rot	No
	<i>SHA-512/384</i>	512/384	512	1024	$2^{128} - 1$	64	80		

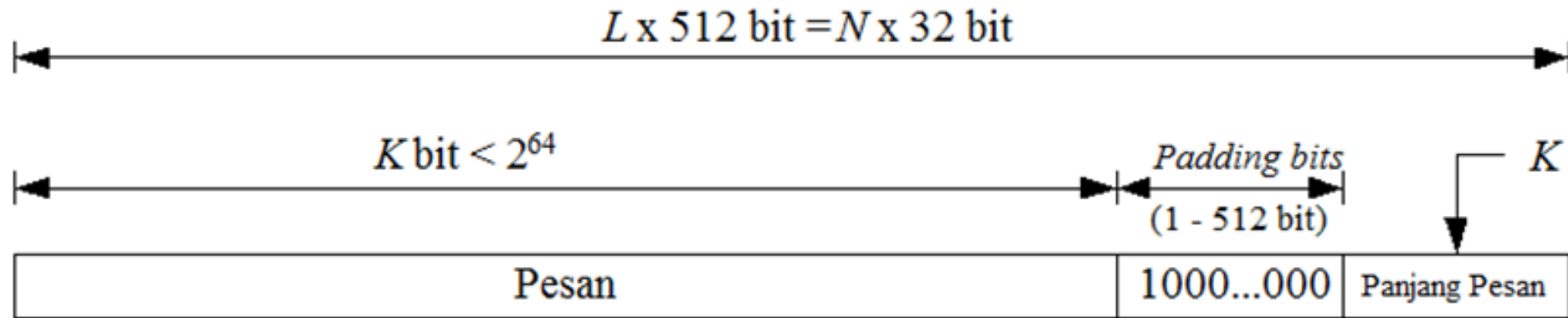
# Gambaran Umum SHA-1



## **Langkah-langkah pembuatan *message digest* dengan SHA-1:**

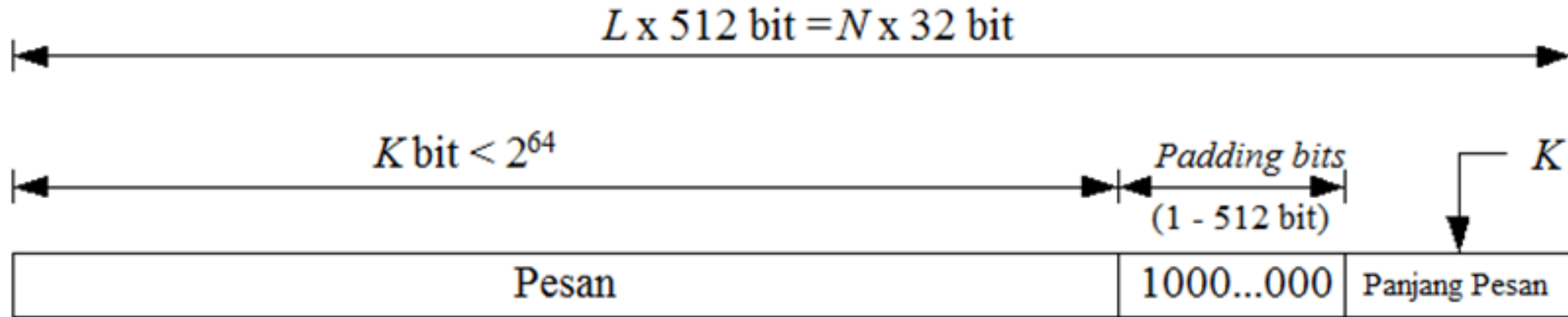
1. Penambahan bit-bit pengganjal (*padding bits*).
2. Penambahan nilai panjang pesan semula.
3. Inisialisasi penyangga (*buffer*) MD.
4. Pengolahan pesan dalam blok berukuran 512 bit.

# 1. Penambahan Bit-bit Pengganjal



- Pesan ditambah dengan sejumlah bit pengganjal (*padding bits*) sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 (mod 512).
- Panjang bit-bit pengganjal adalah antara 1 sampai 512.
- Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.
- Contoh:  $K = 32000 \text{ bit} \rightarrow 32000 + 192 \text{ bit} = 32192 \rightarrow 32192 \text{ mod } 512 = 448$

## 2. Penambahan Nilai Panjang Pesan



- Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula.

Contoh:  $K = 32000 = 1111101000000000_2 = 000\dots1111101000000000$

- Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.



### 3. Inisialisasi Penyangga MD

- *SHA* membutuhkan 5 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit.
- Total panjang penyangga adalah  $5 \times 32 = 160$  bit.
- Kelima penyangga *MD* ini diberi nama *A*, *B*, *C*, *D*, dan *E*. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi HEX) sebagai berikut:

$A = 67452301$

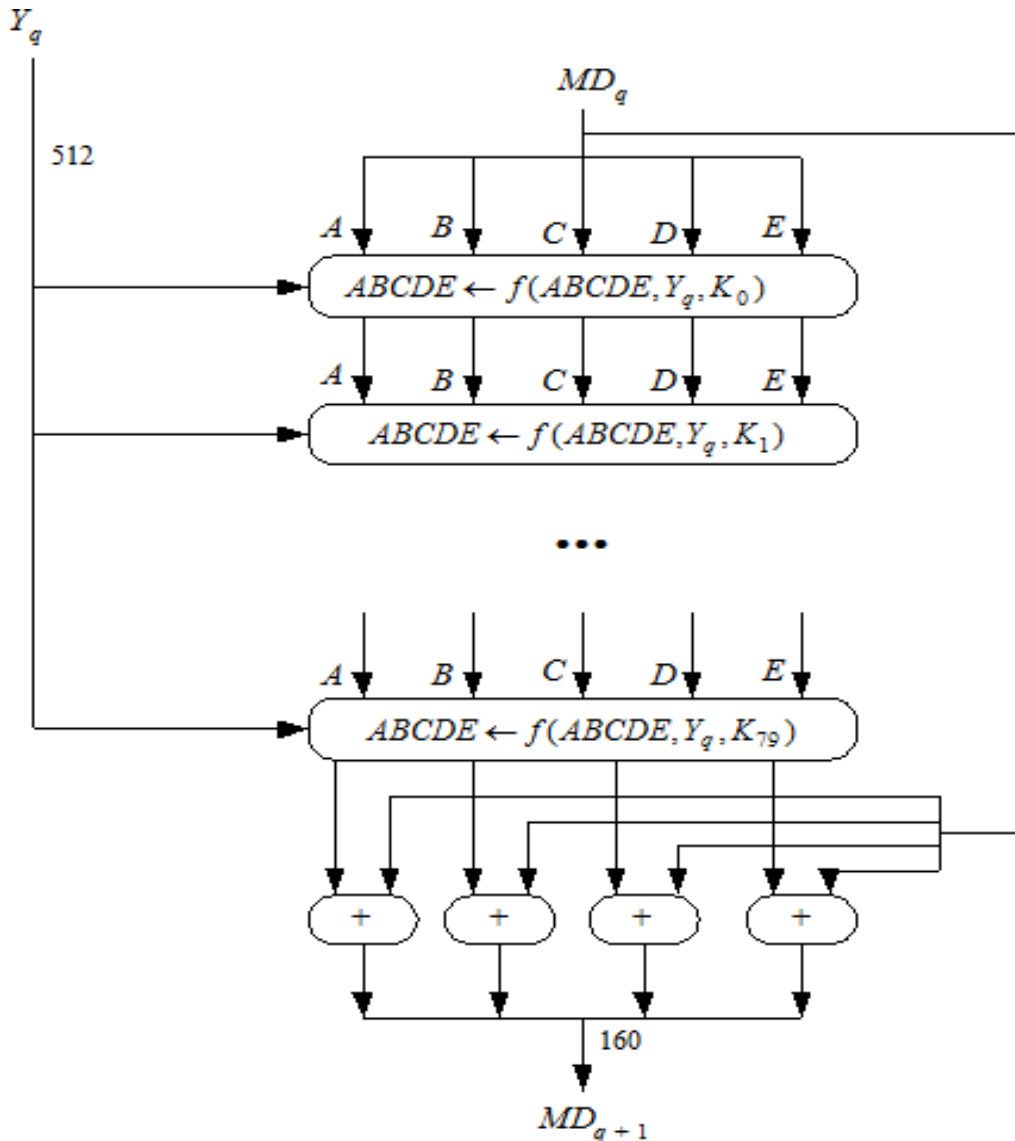
$B = \text{EFCDAB89}$

$C = 98BADCFE$

$D = 10325476$

$E = \text{C3D2E1F0}$

## 4. Pengolahan Pesan dalam Blok Berukuran 512 bit.



- Proses  $H_{\text{SHA}}$  terdiri dari 80 buah putaran ( $MD5$  hanya 4 putaran)
- Masing-masing putaran menggunakan bilangan penambah  $K_t$ , yaitu:

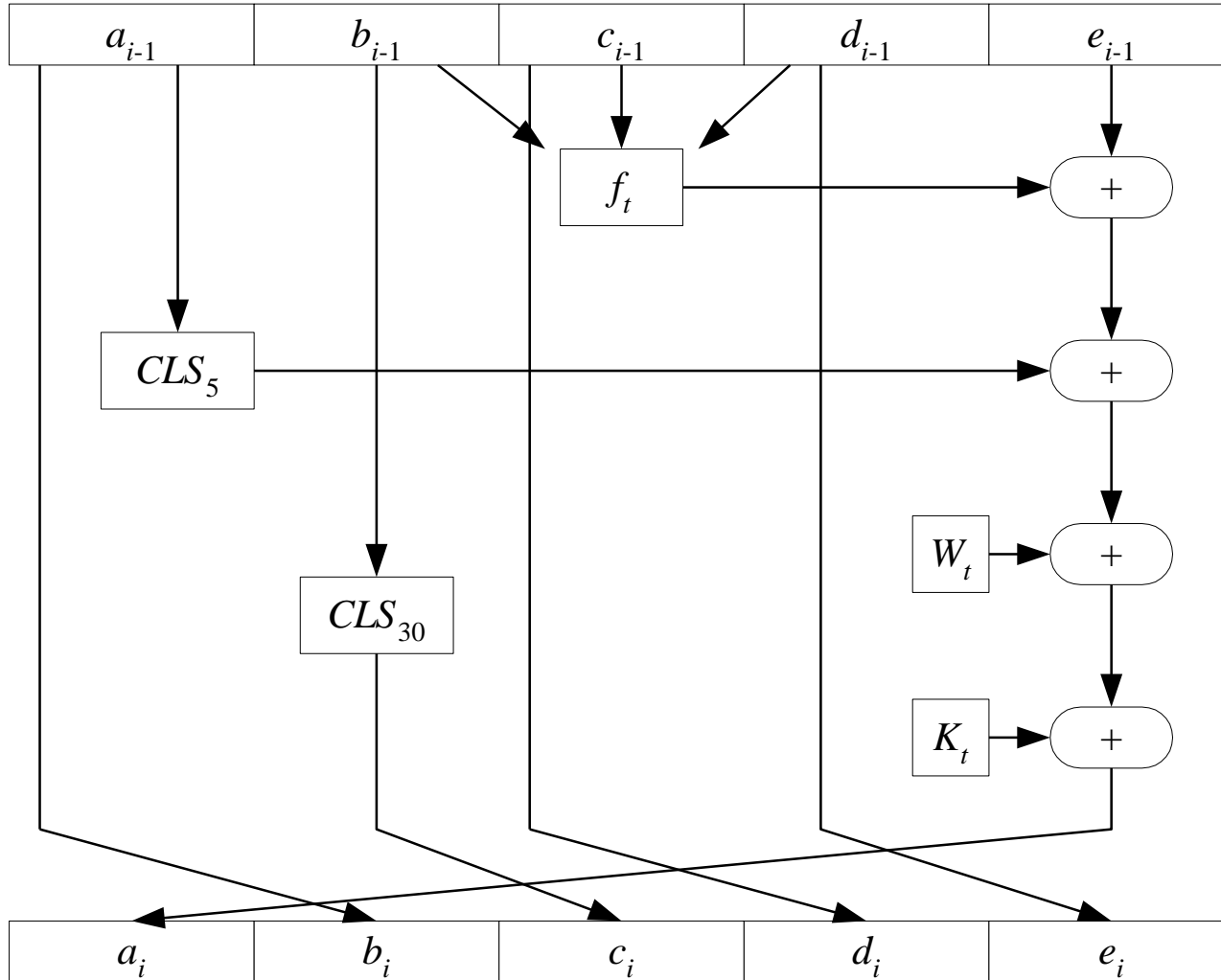
Putaran  $0 \leq t \leq 19$      $K_t = 5A827999$

Putaran  $20 \leq t \leq 39$      $K_t = 6ED9EBA1$

Putaran  $40 \leq t \leq 59$      $K_t = 8F1BBCDC$

Putaran  $60 \leq t \leq 79$      $K_t = CA62C1D6$

# Operasi dasar pada setiap putaran:



**Tabel 1.** Fungsi logika  $f_i$  pada setiap putaran

Putaran	$f_i(b, c, d)$
0 .. 19	$(b \wedge c) \vee (\sim b \wedge d)$
20 .. 39	$b \oplus c \oplus d$
40 .. 59	$(b \wedge c) \vee (b \wedge d) \vee (c \wedge d)$
60 .. 79	$b \oplus c \oplus d$

$$e \leftarrow d$$

$$d \leftarrow c$$

$$c \leftarrow \text{CLS}_{30}(b)$$

$$b \leftarrow a$$

$$a \leftarrow (\text{CLS}_5(a) + f_t(b, c, d) + e + W_t + K_t)$$

Nilai  $W_1$  sampai  $W_{16}$  berasal dari 16 *word* pada blok yang sedang diproses (1 *word* = 32 bit), sedangkan nilai  $W_t$  berikutnya didapatkan dari persamaan

$$W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}$$

Contoh *message digest* beberapa pesan yang dihasilkan oleh SHA-1:

sha1("halo")= 33b1eac210971fb02a3b90afce9dbff758be794d

sha1("halo, apa kabar teman? ") =  
03d17abd2962dbed1037d1230f012037cd25fe91

sha1("The quick brown fox jumps over the lazy dog") =  
2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

# Kriptanalisis *SHA-1*

- Pada tahun 2005, Rijmen dan Oswald mempublikasikan serangan pada versi *SHA-1* yang direduksi (hanya menggunakan 53 putaran dari 80 putaran) dan menemukan kolisi dengan kompleksitas sekitar  $2^{80}$  operasi (lihat di <http://eprint.iacr.org/2005/010>).
- Pada bulan Februari 2005, Xiayoun Wang, Yiqun Lisa Yin, dan Hongbo Yo mempublikasikan serangan yang dapat menemukan kolisi pada versi penuh *SHA-1*, yang membutuhkan sekitar  $2^{69}$  operasi (lihat beritanya di: [http://www.schneier.com/blog/archives/2005/02/sha\\_1broken.html](http://www.schneier.com/blog/archives/2005/02/sha_1broken.html))

- Rizki Wicaksono, seorang *hacker* alumni Informatika ITB Angkatan 1999 mendemonstrasikan cara membentuk dua file berbeda yang memiliki nilai *hash* SHA-1 sama (silakan baca tulisannya pada pranala:

<http://www.ilmuhacking.com/cryptography/membuat-sha1-collision-file/>



Rizki Wicaksono

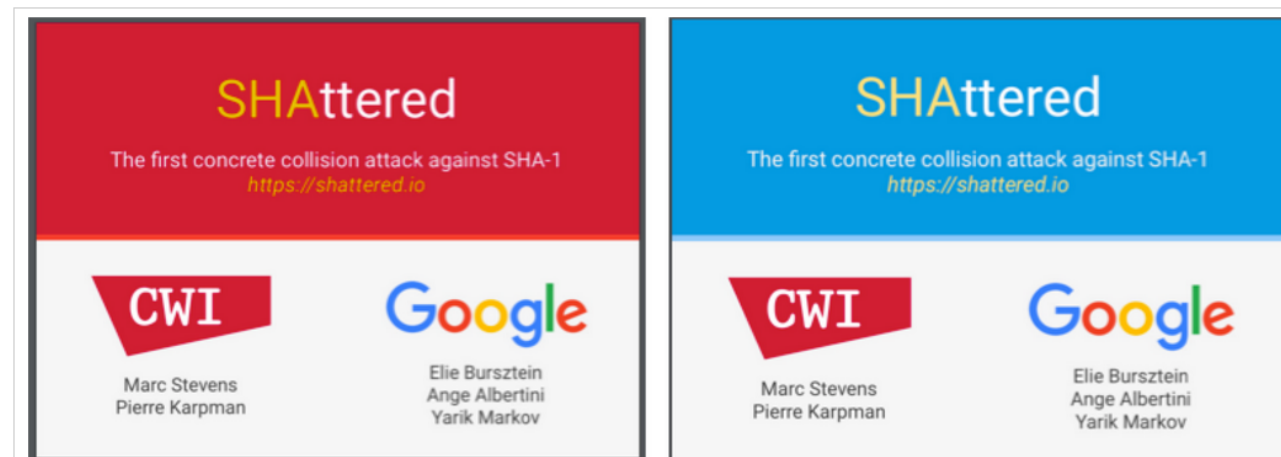
Published

March 21, 2017

in Cryptography

## Membuat file dengan SHA1 collision

Sejumlah ilmuwan dengan didukung kekuatan komputasi Google berhasil melakukan serangan **SHA1 collision terhadap dokumen PDF**. Mereka menciptakan dua file PDF dengan nilai SHA1 hash yang identik walaupun isi filenya berbeda.



Rinaldi Munir/II4031 Kriptografi dan Koding  
shattered-1.pdf dan shattered-2.pdf memiliki SHA1 yang sama

**SELAMAT BELAJAR**