

Bahan Kuliah IF4020 Kriptografi



# Fungsi Hash MD5

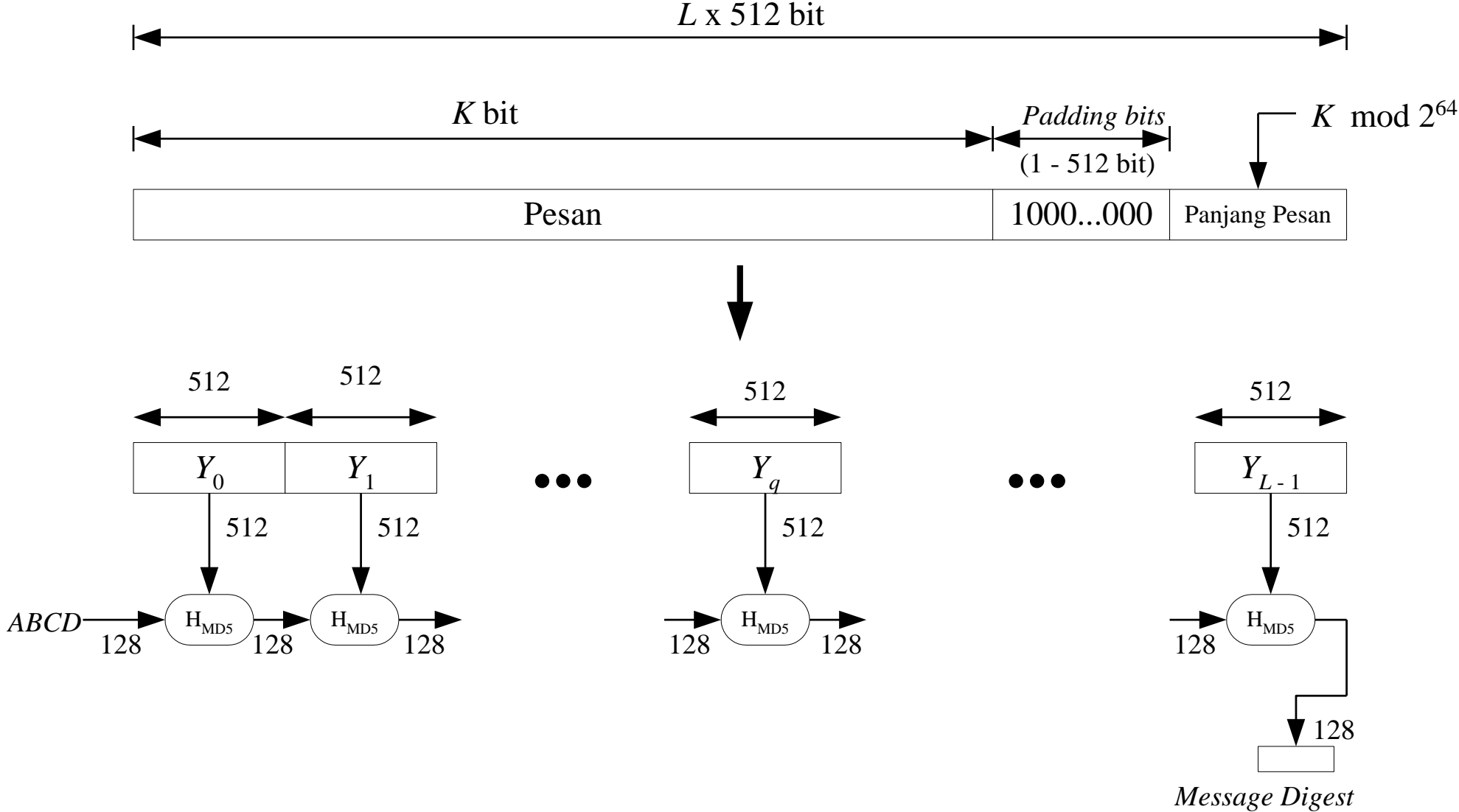
Oleh: Dr. Rinaldi Munir

**Program Studi Informatika  
Sekolah Teknik Elektro dan Informatika(STEI)  
ITB**

# Pendahuluan

- *MD5* adalah fungsi *hash* satu-arah yang dibuat oleh Ron Rivest.
- *MD5* merupakan perbaikan dari *MD4* setelah *MD4* ditemukan kolisinya.
- Algoritma *MD5* menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit.

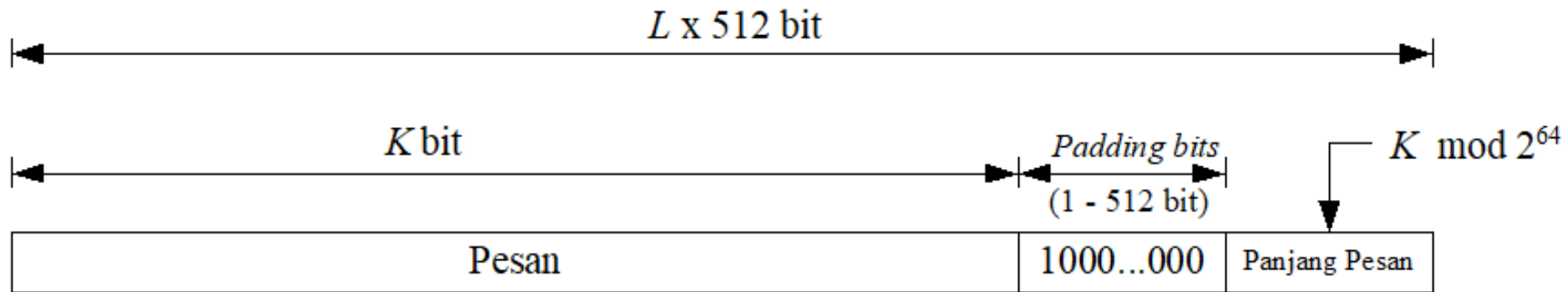
# Gambaran umum MD5



## **Langkah-langkah pembuatan *message digest* secara garis besar:**

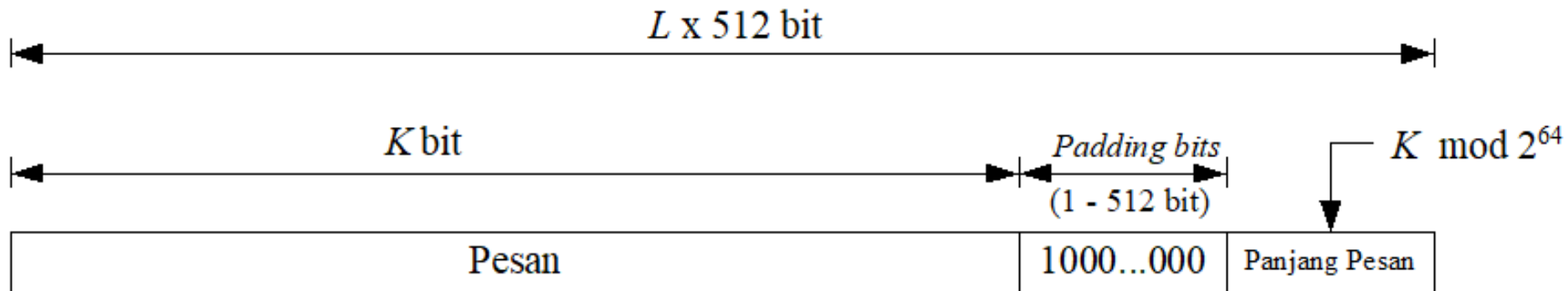
1. Penambahan bit-bit pengganjal (*padding bits*).
2. Penambahan nilai panjang pesan semula.
3. Inisialisasi penyangga (*buffer*) MD.
4. Pengolahan pesan dalam blok berukuran 512 bit.

# 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:  $K = 32000$  bit  $\rightarrow 32000 + 192$  bit = 32192  $\rightarrow 32192 \bmod 512 = 448$

## 2. Penambahan Nilai Panjang Pesan



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

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

- Jika  $K > 2^{64}$  maka yang diambil adalah  $K \bmod 2^{64}$ .
- Setelah ditambah dengan 64 bit, panjang pesan sekarang menjadi kelipatan 512 bit.

### 3. Inisialisasi Penyangga MD

- *MD5* membutuhkan 4 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah  $4 \times 32 = 128$  bit. Keempat penyangga ini menampung hasil antara dan hasil akhir.
- Keempat penyangga ini diberi nama *A*, *B*, *C*, dan *D*. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi HEX) sebagai berikut:

*A* = 01234567

*B* = 89ABCDEF

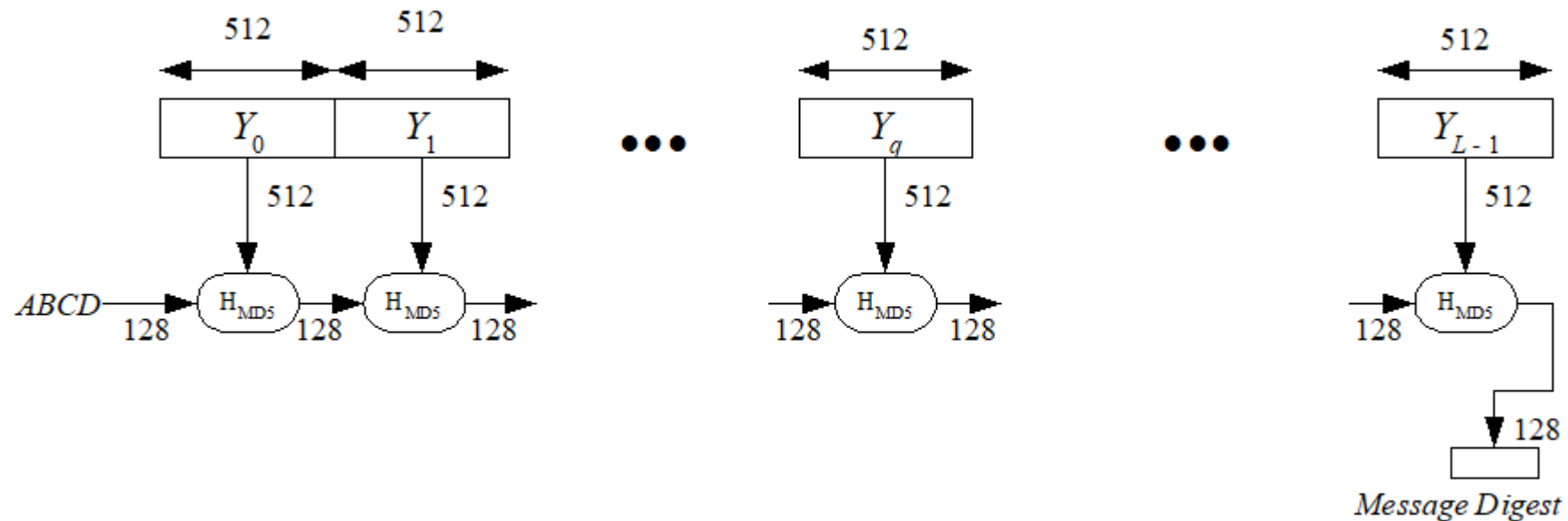
*C* = FEDCBA98

*D* = 76543210

- Untuk mempersingkat penyebutan, keempat penyangga itu disebut MD saja

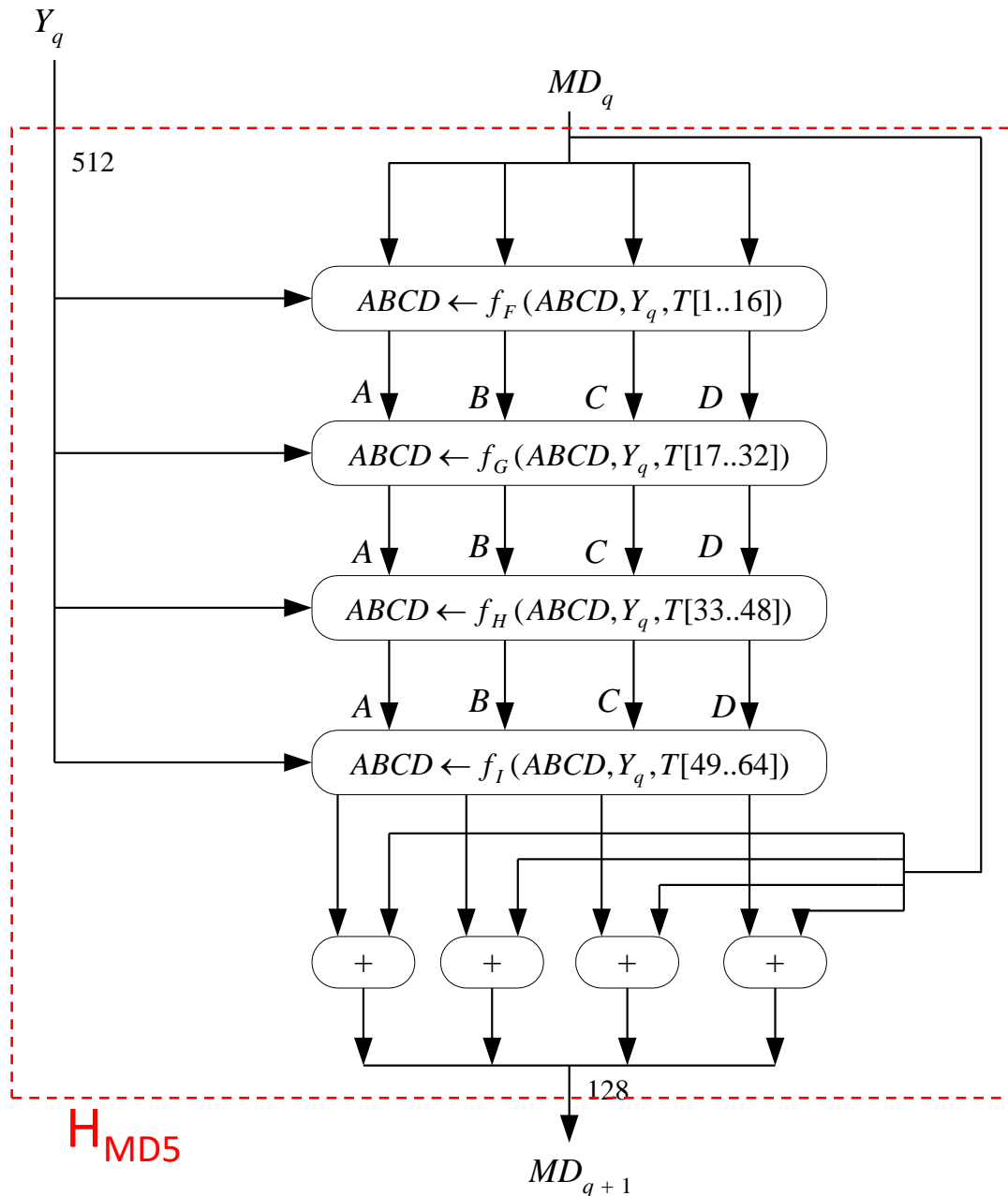
## 4. Pengolahan Pesan dalam Blok Berukuran 512 bit.

- Pesan dibagi menjadi  $L$  buah blok yang masing-masing panjangnya 512 bit ( $Y_0$  sampai  $Y_{L-1}$ ).



- Setiap blok 512-bit diproses bersama dengan penyangga  $MD$  menjadi keluaran 128-bit, dan ini disebut proses  $H_{MD5}$ .



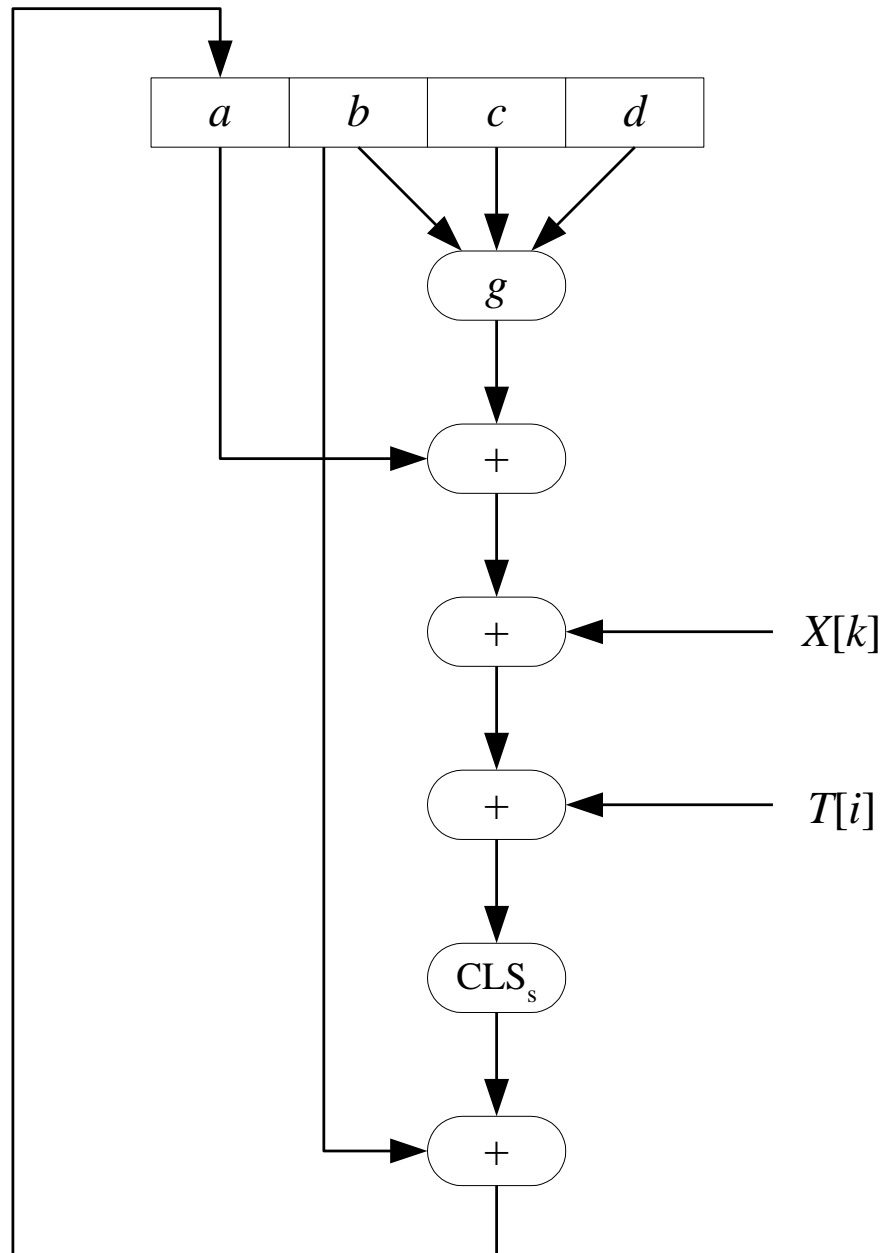


Pada Gambar tersebut,  $Y_q$  menyatakan blok 512-bit ke- $q$

$MD_q$  adalah nilai *message digest* 128-bit dari proses  $H_{MD5}$  ke- $q$ . Pada awal proses,  $MD_q$  berisi nilai inisialisasi penyangga  $MD$ .

Proses  $H_{MD5}$  terdiri dari 4 buah putaran:

- Setiap putaran melakukan operasi dasar  $MD5$  sebanyak 16 kali
- Operasi dasar melibatkan fungsi  $f_F$ ,  $f_G$ ,  $f_H$ , dan  $f_I$
- Setiap operasi dasar memakai sebuah elemen  $T$
- Jadi setiap putaran memakai 16 elemen Tabel  $T$ .



Operasi dasar *MD5* pada gambar di samping dapat ditulis dengan sebuah persamaan:

$$a \leftarrow b + CLS_s(a + g(b, c, d) + X[k] + T[i])$$

$a, b, c, d$  = empat buah peubah penyangga 32-bit  $A, B, C, D$

$g$  = salah satu fungsi  $F, G, H, I$

$CLS_s$  = *circular left shift* sebanyak  $s$  bit

$X[k]$  = kelompok 32-bit ke- $k$  dari blok 512 bit *message* ke- $q$ .  
Nilai  $k = 0$  sampai 15.

$T[i]$  = elemen Tabel  $T$  ke- $i$  (32 bit)

$+$  = operasi penjumlahan dalam modulo  $2^{32}$

**Tabel 1.** Fungsi-fungsi dasar MD5

<b>Nama</b>	<b>Notasi</b>	<b><math>g(b, c, d)</math></b>
$f_F$	$F(b, c, d)$	$(b \wedge c) \vee (\sim b \wedge d)$
$f_G$	$G(b, c, d)$	$(b \wedge d) \vee (c \wedge \sim d)$
$f_H$	$H(b, c, d)$	$b \oplus c \oplus d$
$f_I$	$I(b, c, d)$	$c \oplus (b \wedge \sim d)$

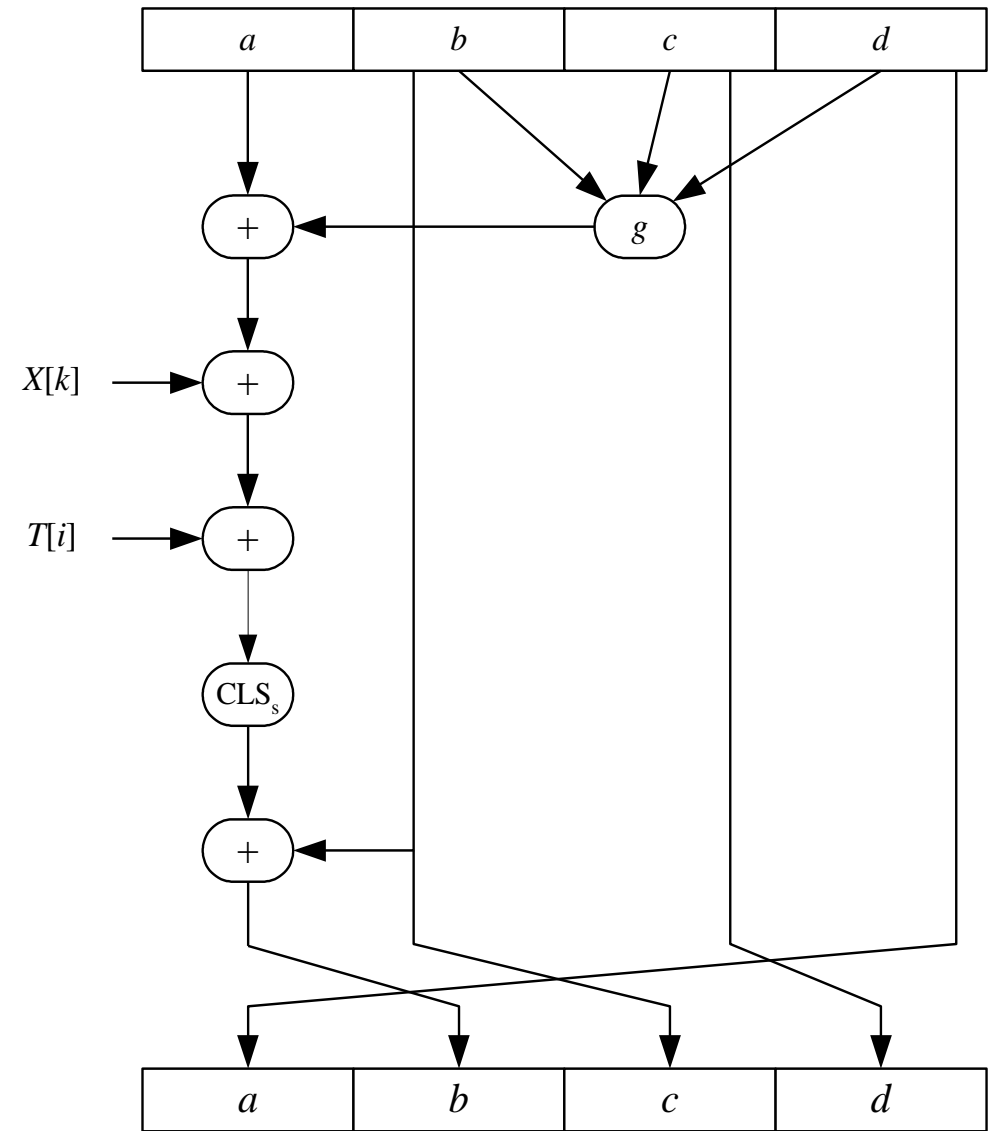
Catatan: operator logika AND, OR, NOT, XOR masing-masing dilambangkan dengan  $\wedge$ ,  $\vee$ ,  $\sim$ ,  $\oplus$

- Karena ada 16 kali operasi dasar, maka setiap kali selesai satu operasi dasar, penyangga-penyangga itu digeser ke kanan secara sirkuler dengan cara pertukaran sebagai berikut:

```

temp ← d
d ← c
c ← b
b ← a
a ← temp

```



**Tabel 2.** Nilai  $T[i]$

---

T[1] = D76AA478	T[17] = F61E2562	T[33] = FFFA3942	T[49] = F4292244
T[2] = E8C7B756	T[18] = C040B340	T[34] = 8771F681	T[50] = 432AFF97
T[3] = 242070DB	T[19] = 265E5A51	T[35] = 69D96122	T[51] = AB9423A7
T[4] = C1BDCEEE	T[20] = E9B6C7AA	T[36] = FDE5380C	T[52] = FC93A039
T[5] = F57C0FAF	T[21] = D62F105D	T[37] = A4BEEA44	T[53] = 655B59C3
T[6] = 4787C62A	T[22] = 02441453	T[38] = 4BDECFA9	T[54] = 8F0CCC92
T[7] = A8304613	T[23] = D8A1E681	T[39] = F6BB4B60	T[55] = FFEFF47D
T[8] = FD469501	T[24] = E7D3FBCB	T[40] = BEBFBC70	T[56] = 85845DD1
T[9] = 698098D8	T[25] = 21E1CDE6	T[41] = 289B7EC6	T[57] = 6FA87E4F
T[10] = 8B44F7AF	T[26] = C33707D6	T[42] = EAA127FA	T[58] = FE2CE6E0
T[11] = FFFF5BB1	T[27] = F4D50D87	T[43] = D4EF3085	T[59] = A3014314
T[12] = 895CD7BE	T[28] = 455A14ED	T[44] = 04881D05	T[60] = 4E0811A1
T[13] = 6B901122	T[29] = A9E3E905	T[45] = D9D4D039	T[61] = F7537E82
T[14] = FD987193	T[30] = FCEFA3F8	T[46] = E6DB99E5	T[62] = BD3AF235
T[15] = A679438E	T[31] = 676F02D9	T[47] = 1FA27CF8	T[63] = 2AD7D2BB
T[16] = 49B40821	T[32] = 8D2A4C8A	T[48] = C4AC5665	T[64] = EB86D391

---

- Misalkan notasi

$[abcd \ k \ s \ i]$

menyatakan operasi

$$a \leftarrow b + CLS_s(a + g(b, c, d) + X[k] + T[i])$$

maka operasi dasar pada masing-masing putaran dapat ditabulasikan sebagai berikut:

Putaran 1: 16 kali operasi dasar dengan  $g(b, c, d) = F(b, c, d)$  dapat dilihat pada Tabel 3.

**Tabel 3.** Rincian operasi pada fungsi  $F(b, c, d)$

No.	[abcd	k	s	i]
1	[ABCD	0	7	1]
2	[DABC	1	12	2]
3	[CDAB	2	17	3]
4	[BCDA	3	22	4]
5	[ABCD	4	7	5]
6	[DABC	5	12	6]
7	[CDAB	6	17	7]
8	[BCDA	7	22	8]
9	[ABCD	8	7	9]
10	[DABC	9	12	10]
11	[CDAB	10	17	11]
12	[BCDA	11	22	12]
13	[ABCD	12	7	13]
14	[DABC	13	12	14]
15	[CDAB	14	17	15]
16	[BCDA	15	22	16]

$$a \leftarrow b + CLS_s(a + g(b, c, d) + X[k] + T[i])$$

Putaran 2: 16 kali operasi dasar dengan  $g(b, c, d) = G(b, c, d)$  dapat dilihat pada Tabel 4.

**Tabel 4.** Rincian operasi pada fungsi  $G(b, c, d)$

No .	[ $abcd$ $k$ $s$ $i$ ]
1	[ABCD 1 5 17]
2	[DABC 6 9 18]
3	[CDAB 11 14 19]
4	[BCDA 0 20 20]
5	[ABCD 5 5 21]
6	[DABC 10 9 22]
7	[CDAB 15 14 23]
8	[BCDA 4 20 24]
9	[ABCD 9 5 25]
10	[DABC 14 9 26]
11	[CDAB 3 14 27]
12	[BCDA 8 20 28]
13	[ABCD 13 5 29]
14	[DABC 2 9 30]
15	[CDAB 7 14 31]
16	[BCDA 12 20 32]

$$a \leftarrow b + CLS_s(a + g(b, c, d) + X[k] + T[i])$$



Putaran 3: 16 kali operasi dasar dengan  $g(b, c, d) = H(b, c, d)$  dapat dilihat pada Tabel 5.

**Tabel 5.** Rincian operasi pada fungsi  $H(b, c, d)$

No .	[ <i>abcd</i> <i>k</i> <i>s</i> <i>i</i> ]
1	[ABCD 5 4 33]
2	[DABC 8 11 34]
3	[CDAB 11 16 35]
4	[BCDA 14 23 36]
5	[ABCD 1 4 37]
6	[DABC 4 11 38]
7	[CDAB 7 16 39]
8	[BCDA 10 23 40]
9	[ABCD 13 4 41]
10	[DABC 0 11 42]
11	[CDAB 3 16 43]
12	[BCDA 6 23 44]
13	[ABCD 9 4 45]
14	[DABC 12 11 46]
15	[CDAB 15 16 47]
16	[BCDA 2 23 48]

$$a \leftarrow b + CLS_s(a + g(b, c, d) + X[k] + T[i])$$

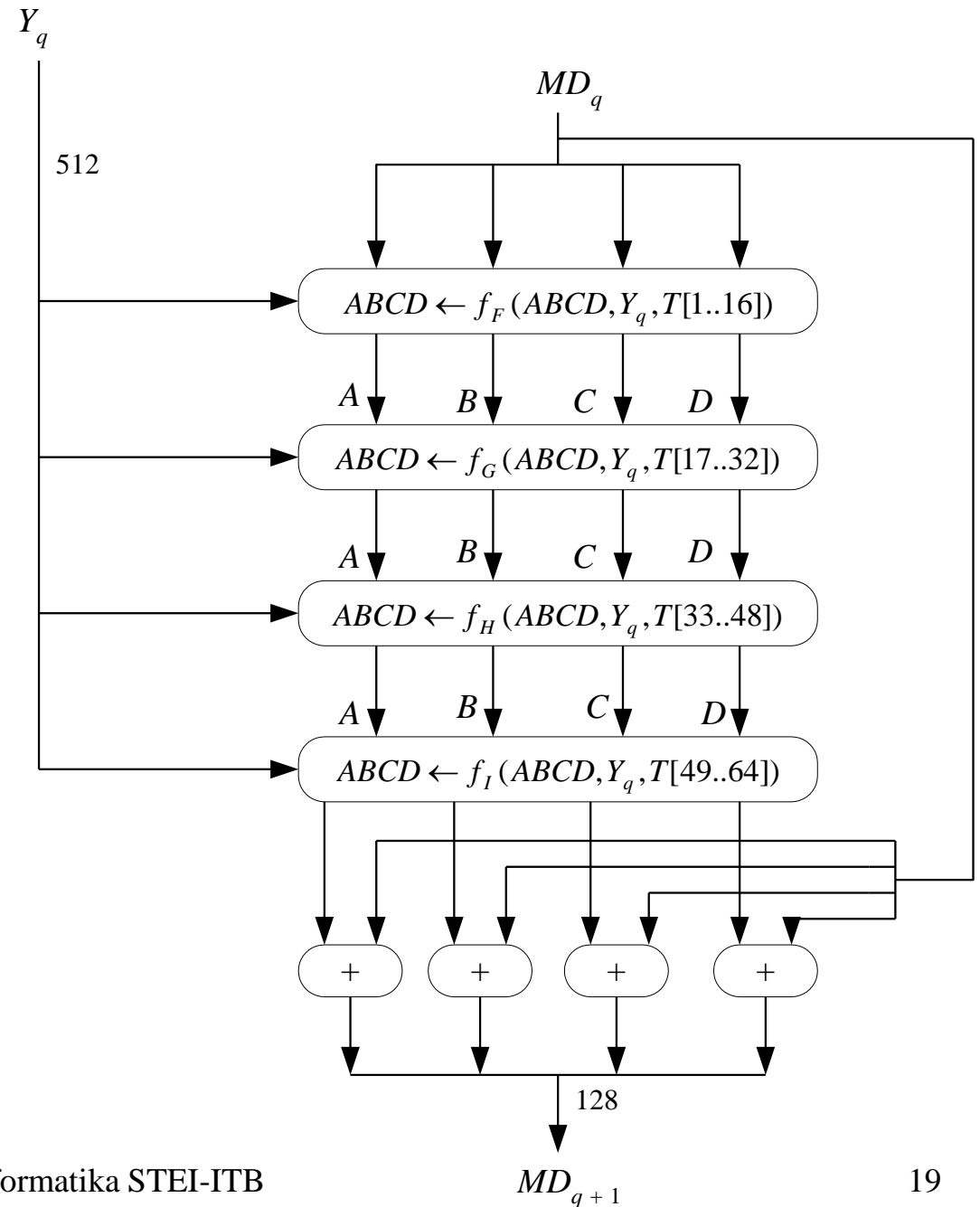
Putaran 4: 16 kali operasi dasar dengan  $g(b, c, d) = I(b, c, d)$  dapat dilihat pada Tabel 6.

**Tabel 6.** Rincian operasi pada fungsi  $I(b, c, d)$

No .	[ $abcd$ $k$ $s$ $i$ ]
1	[ABCD 0 6 49]
2	[DABC 7 10 50]
3	[CDAB 14 15 51]
4	[BCDA 5 21 52]
5	[ABCD 12 6 53]
6	[DABC 3 10 54]
7	[CDAB 10 15 55]
8	[BCDA 1 21 56]
9	[ABCD 8 6 57]
10	[DABC 15 10 58]
11	[CDAB 6 15 59]
12	[BCDA 13 21 60]
13	[ABCD 4 6 61]
14	[DABC 11 10 62]
15	[CDAB 2 15 63]
16	[BCDA 9 21 64]

$$a \leftarrow b + CLS_s(a + g(b, c, d) + X[k] + T[i])$$

- Setelah putaran ke-4,  $a$ ,  $b$ ,  $c$ , dan  $d$  ditambahkan ke  $A$ ,  $B$ ,  $C$ , dan  $D$ .
- Selanjutnya algoritma memproses untuk blok data berikutnya ( $Y_{q+1}$ ).
- Luaran akhir dari algoritma  $MD5$  adalah hasil penyambungan bit-bit di  $A$ ,  $B$ ,  $C$ , dan  $D$ .



- Contoh *message digest* beberapa pesan yang dihasilkan oleh MD5:

md5("halo") = 57f842286171094855e51fc3a541c1e2

md5("halo, apa kabar teman?") =  
d29029a1dcf256466290d773f5dbfc0f

md5("The quick brown fox jumps over the lazy dog") =  
9e107d9d372bb6826bd81d3542a419d6

**Contoh.** Contoh message digest dari sebuah file, bandung.txt, sebagai berikut:

Pada bulan Oktober 2004 ini, suhu udara kota Bandung terasa lebih panas dari hari-hari biasanya. Menurut laporan Dinas Meteorologi Kota Bandung, suhu tertinggi kota Bandung adalah 33 derajat Celcius pada Hari Rabu, 17 Oktober yang lalu. Suhu tersebut sudah menyamai suhu kota Jakarta pada hari-hari biasa. Menurut Kepala Dinas Meteorologi, peningkatan suhu tersebut terjadi karena posisi bumi sekarang ini lebih dekat ke matahari daripada hari-hari biasa.

Sebutan Bandung sebagai kota sejuk dan dingin mungkin tidak lama lagi akan tinggal kenangan. Disamping karena faktor alam, jumlah penduduk yang padat, polusi dari pabrik di sekita Bandung, asap knalpot kendaraan, ikut menambah kenaikan suhu udara kota.

*Message digest* dari arsip bandung.txt yang dihasilkan oleh algoritma *MD5* adalah 128-bit:

```
0010 1111 1000 0010 1100 0000 1100 1000 1000 0100 0101 0001 0010 0001
1011 0001 1011 1001 0101 0011 1101 0101 0111 1101 0100 1100 0101 1001
0001 1110 0110 0011
```

atau, dalam notasi HEX adalah:

```
2F82D0C845121B953D57E4C3C5E91E63
```

# Kriptanalisis MD5

Review kembali sifat-sifat fungsi *hash*  $H$ :

- a) **collision resistance** : sangat sukar menemukan dua input  $a$  dan  $b$  sedemikian sehingga  $H(a) = H(b)$
  
- b) **preimage resistance**: untuk sembarang output  $y$ , sukar menemukan input  $a$  sedemikian sehingga  $H(a) = y$
  
- c) **second preimage resistance** – untuk input  $a$  dan output  $y = H(a)$ , sukar menemukan input kedua  $b$  sedemikian sehingga  $H(b) = y$

# Kriptanalisis MD5

- Secara teori, menemukan kolisi pada fungsi *hash* sangatlah sukar dilakukan.
- Pada awalnya penemu algoritma *MD5* menganggap usaha tersebut hampir tidak mungkin dilakukan karena membutuhkan waktu yang sangat lama.
- Tetapi, pada tahun 1996, Dobbertin melaporkan penemuan kolisi pada algoritma *MD5* meskipun kecacatan ini bukan kelemahan yang fatal.

- Pada tanggal 1 Maret 2005, Arjen Lenstra, Xiaoyun Wang, dan Benne de Weger mendemostraskan pembentukan dua buah pesan berbeda (berupa sertifikat X.509, yang akan dijelaskan di dalam bab *Infrastruktur Kunci Publik*) tetapi mempunyai nilai *hash* yang sama (lihat publikasinya di dalam <http://eprint.iacr.org/2005/067>).
- Beberapa hari kemudian, Vlastimil Klima (<http://eprint.iacr.org/2005/075>) memperbaiki algoritma Lenstra dkk yang dapat menghasilkan kolisi *MD5* hanya dalam waktu beberapa jam dengan menggunakan komputer *PC*.

(Sumber: Wikipedia)



Contoh dua pesan yang hanya berbeda 6 bit tetapi memiliki nilai *hash* sama (Sumber: *Eric Rescorla (2004-08-17). "A real MD5 collision"*):

M1 =

```
d131dd02c5e6eec4693d9a0698aff95c2fcab58712467eab400458
3eb8fb7f8955ad340609f4b30283e488832571415a085125e8f7cd
c99fd91dbdf280373c5bd8823e3156348f5bae6dacd436c919c6dd
53e2b487da03fd02396306d248cda0e99f33420f577ee8ce54b670
80a80d1ec69821bcb6a8839396f9652b6ff72a70
```

M2 =

```
d131dd02c5e6eec4693d9a0698aff95c2fcab50712467eab400458
3eb8fb7f8955ad340609f4b30283e4888325f1415a085125e8f7cd
c99fd91dbd7280373c5bd8823e3156348f5bae6dacd436c919c6dd
53e23487da03fd02396306d248cda0e99f33420f577ee8ce54b670
80280d1ec69821bcb6a8839396f965ab6ff72a70
```

Kedua pesan di atas memiliki nilai *hash* **79054025255fb1a26e4bc422aef54eb4**.



**SELAMAT BELAJAR**