# Algoritma MD5

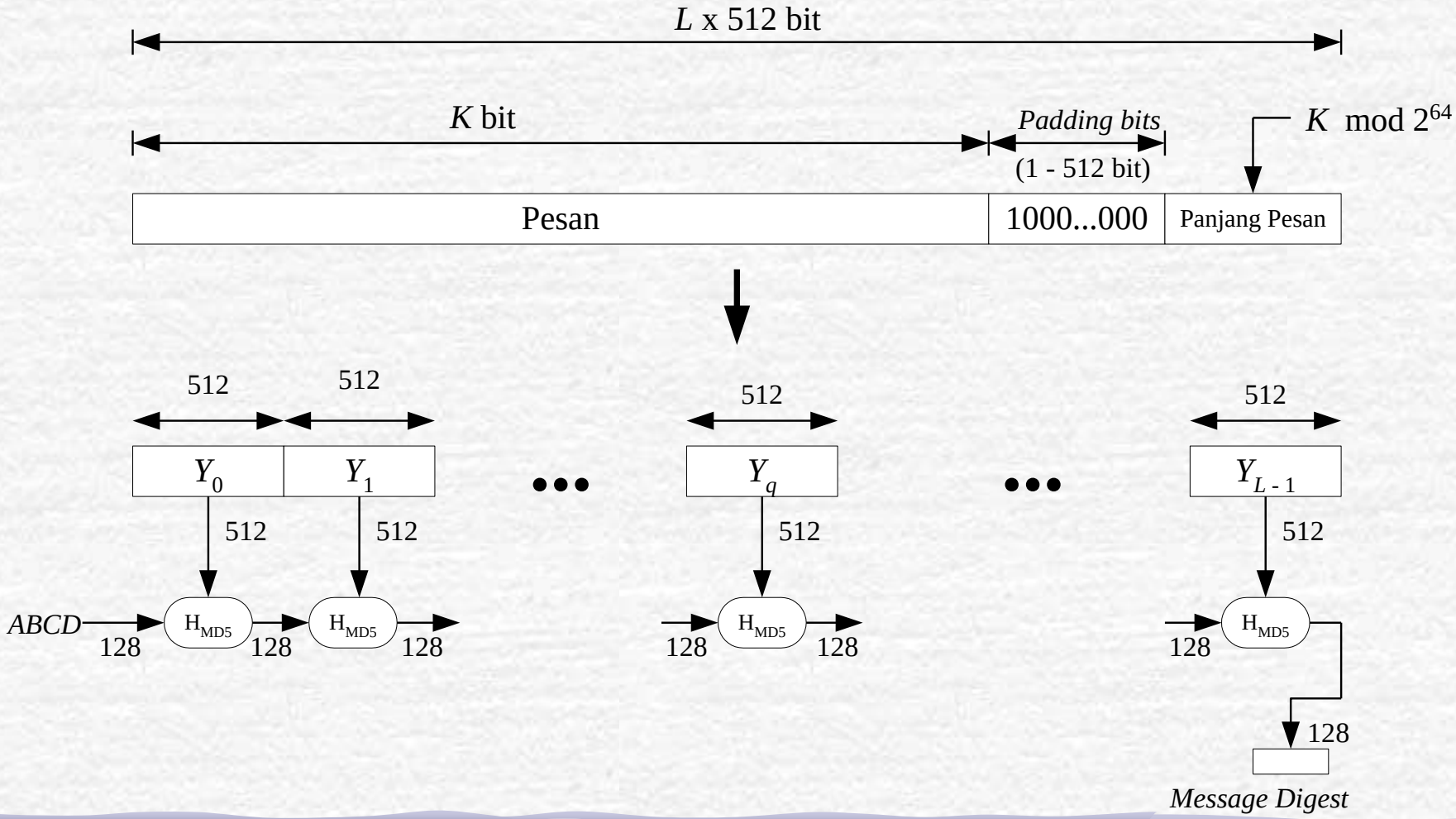
Bahan Kuliah

IF3058 Kriptografi

# 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



- **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 sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 (mod 512).

- Jika panjang pesan 448 bit, maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512.

- Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

# 2. Penambahan Nilai Panjang Pesan

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

- Jika panjang pesan > $2^{64}$ maka yang diambil adalah panjangnya dalam modulo $2^{64}$. Dengan kata lain, jika panjang pesan semula adalah *K* bit, maka 64 bit yang ditambahkan menyatakan *K* modulo $2^{64}$.

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

# 3. Inisialisai 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

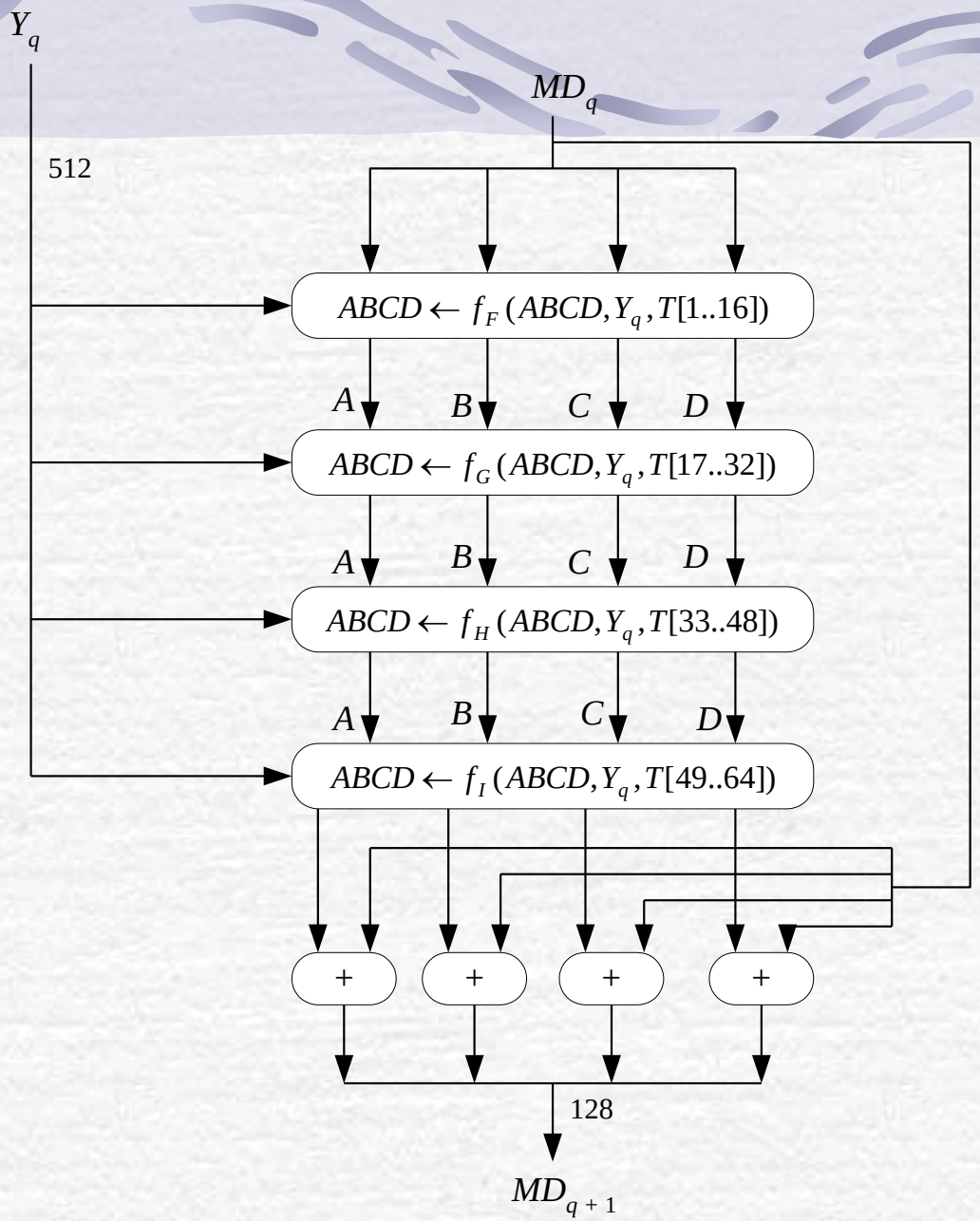
  *D* = 76543210

# 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 $H_{MD5}$.

- Gambaran proses $H_{MD5}$ diperlihatkan pada Gambar berikut:


$Y_q$
$MD_q$
512
$ABCD \leftarrow f_F(ABCD, Y_q, T[1..16])$
A B C D
$ABCD \leftarrow f_G(ABCD, Y_q, T[17..32])$
A B C D
$ABCD \leftarrow f_H(ABCD, Y_q, T[33..48])$
A B C D
$ABCD \leftarrow f_I(ABCD, Y_q, T[49..64])$
+ + + +
128
$MD_{q+1}$

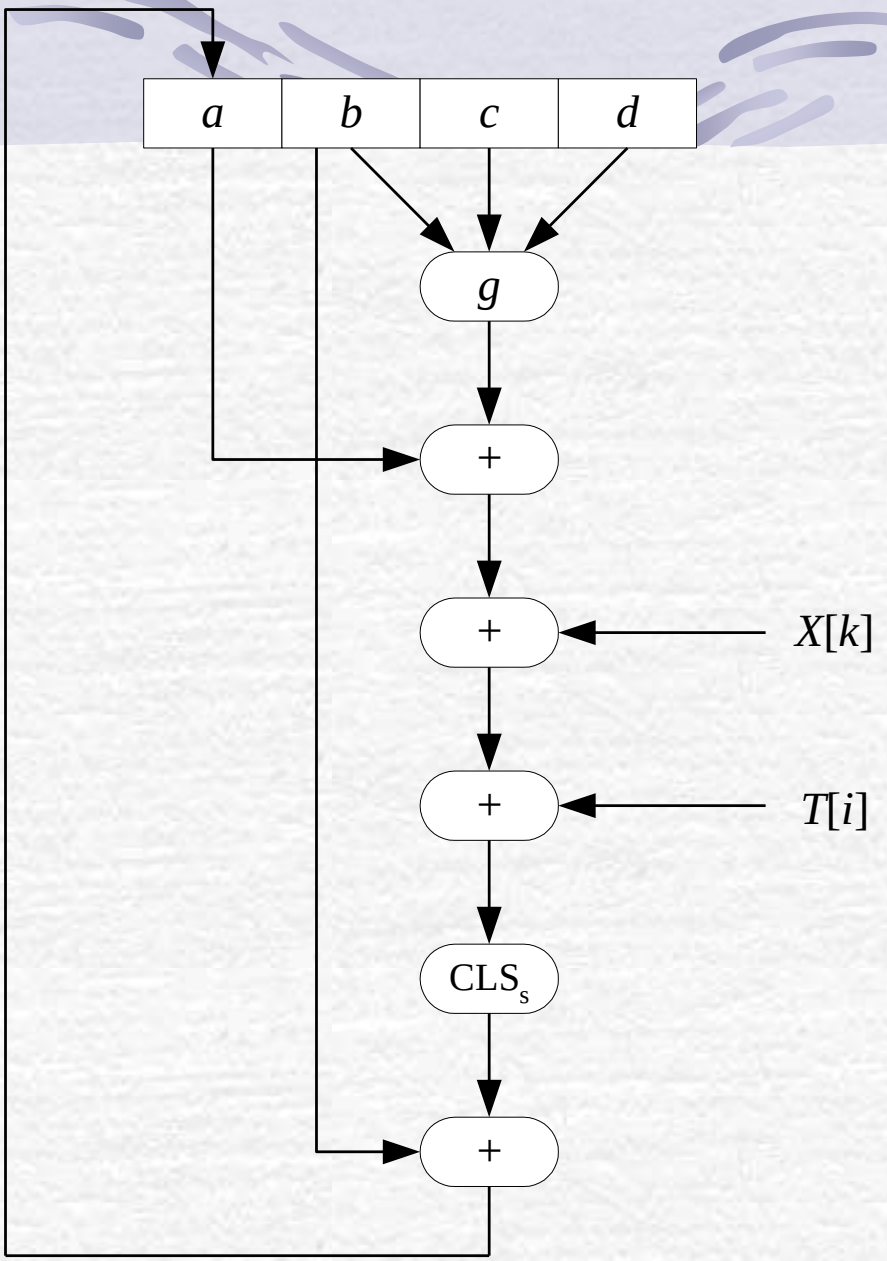
- Pada Gambar tersebut, $Y_q$ menyatakan blok 512-bit ke-$q$

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

- Proses $H_{\text{MD5}}$ terdiri dari 4 buah putaran, dan masing-masing putaran melakukan operasi dasar *MD5* sebanyak 16 kali dan setiap operasi dasar memakai sebuah elemen *T*. Jadi setiap putaran memakai 16 elemen Tabel *T*.

- Fungsi-fungsi $f_F$, $f_G$, $f_H$, dan $f_I$ masing-masing berisi 16 kali operasi dasar terhadap masukan, setiap operasi dasar menggunakan elemen Tabel *T*.

- Operasi dasar *MD5* diperlihatkan pada Gambar berikut:


a
b
c
d
g
+
+
X[k]
+
T[i]
CLS_s
+

Operasi dasar *MD5* yang diperlihatkan pada Gambar di atas dapat ditulis dengan sebuah persamaan sebagai berikut:

$$a \leftarrow b + \text{CLS}_s(a + g(b, c, d) + X[k] + T[i])$$

yang dalam hal ini,

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

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

$\text{CLS}_s$ = *circular left shift* sebanyak *s* bit

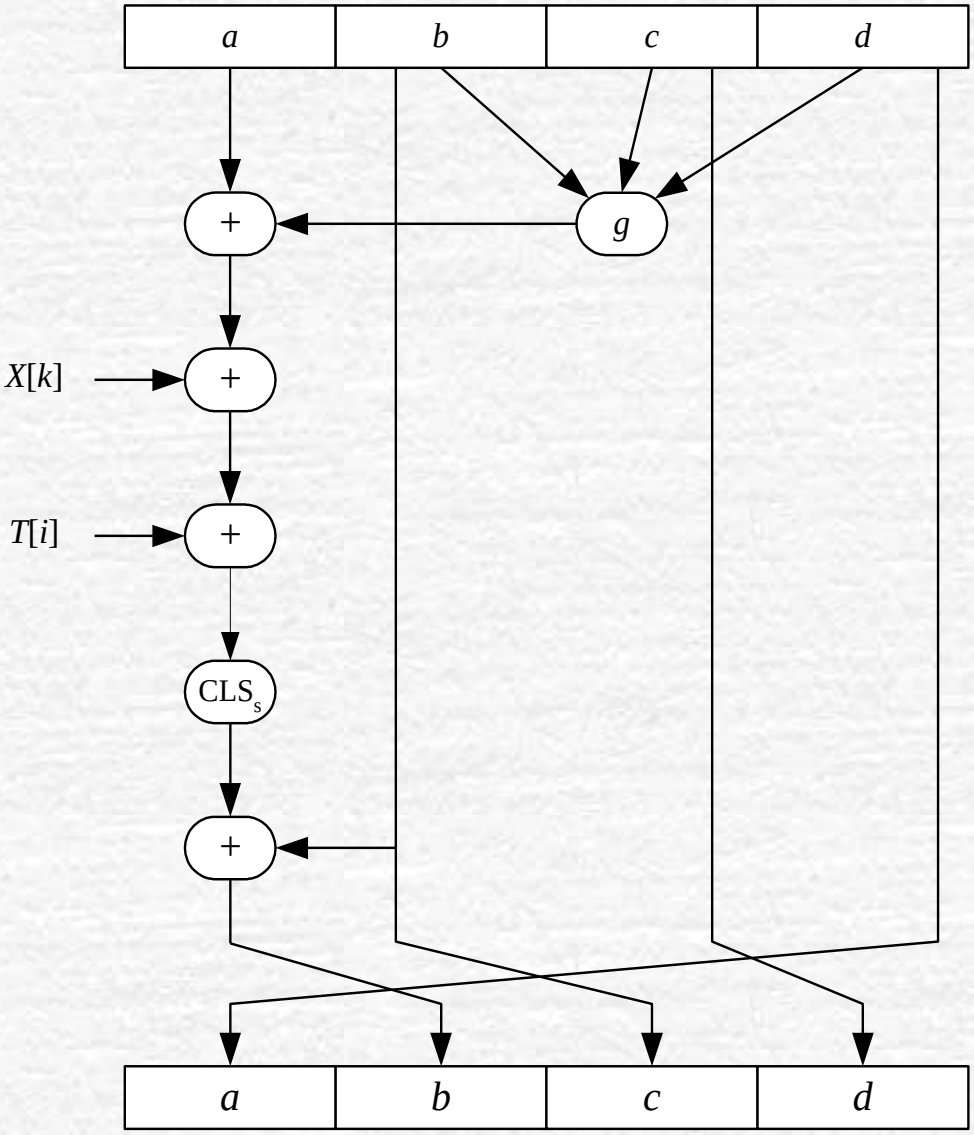
*X*[*k*] = kelompok 32-bit ke-*k* dari blok 512 bit *message* ke-*q*. Nilai *k* = 0 sampai 15.

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

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

- 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
```


a
b
c
d
+
g
X[k]
+
T[i]
+
CLS_s
+
a
b
c
d

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

| **Nama** | **Notasi** | ***g*(*b*, *c*, *d*)** |
|---|---|---|
| $f_F$ | $F(b, c, d)$ | $(b \wedge c) \vee (\sim b \wedge d)$ |
| $f_G$ | $G(b, c, d)$ | $(b \wedge d) \vee (c \wedge \sim d)$ |
| $f_H$ | $H(b, c, d)$ | $b \oplus c \oplus d$ |
| $f_I$ | $I(b, c, d)$ | $c \oplus (b \wedge \sim d)$ |

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

**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

$$[abcd \quad k \quad s \quad i]$$

menyatakan operasi

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

yang dalam hal ini $<<<s$ melambangkan operasi *circular left shift* 32-bit, 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] |

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] |

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. | [*abcd* *k* *s* *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] |

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] |

- Setelah putaran keempat, *a*, *b*, *c*, dan *d* ditambahkan ke *A*, *B*, *C*, dan *D*, dan selanjutnya algoritma memproses untuk blok data berikutnya ($Y_{q+1}$).

- Keluaran akhir dari algoritma *MD5* adalah hasil penyambungan bit-bit di *A*, *B*, *C*, dan *D*.

**Contoh 13.** Misalkan *M* adalah isi sebuah arsip teks `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 terseut 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

- Dengan panjang *message digest* 128 bit, maka secara *brute force* dibutuhkan percobaan sebanyak $2^{128}$ kali untuk menemukan dua buah pesan atau lebih yang mempunyai *message digest* yang sama.

- 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 sertifikat X.509 (akan dijelaskan di dalam bab Infrastruktur Kunci Publik) dengan kunci publik yang berbeda tetapi mempunyai nilai *hash* yang sama (lihat publikasinya di dalam hhttp://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* [WIK06].