Algoritma Kriptografi Modern
(AES, RSA, MD5)

Muhammad Izzuddin Mahali, M.Cs.
Izzudin@uny.ac.id / m.Izzuddin.m@gmail.com
Program Studi Pendidikan Teknik Informatika
Jurusan Pendidikan Teknik Elektronika
Fakultas Teknik
Universitas Negeri Yogyakarta

Pendahuluan

- Beroperasi dalam mode bit (algoritma kriptografi klasik beroperasi dalam mode karakter)
- kunci, plainteks, cipherteks, diproses dalam rangkaian bit
- operasi bit xor paling banyak digunakan

Pendahuluan

- Tetap menggunakan gagasan pada algoritma klasik: substitusi dan transposisi, tetapi lebih rumit (sangat sulit dipecahkan)
- Perkembangan algoritma kriptografi modern didorong oleh penggunaan komputer digital untuk keamanan pesan.
- Komputer digital merepresentasikan data dalam biner.

Algoritma Enkripsi dengan rangkaian bit

- Pesan (dalam bentuk rangkaian bit) dipecah menajdi beberapa blok

- Contoh: Plainteks `100111010110`

  Bila dibagi menjadi blok 4-bit

  `1001  1101  0110`

  maka setiap blok menyatakan `0` sampai `15`:

  `9  13  6`

Algoritma Enkripsi dengan rangkaian bit

Bila plainteks dibagi menjadi blok 3-bit:

100 111 010 110

maka setiap blok menyatakan 0 sampai 7:

4 7 2 6

Jenis Algoritma Kriptografi

- Algoritma Simetri
  a. Blok Chiper : DES, IDEA, AES
  b. Stream Chiper : OTP, A5 dan RC4
- Algoritma Asimetri : RSA, DH, ECC, DSA
- Fungsi Hash : MD5, SHA1

AES (Advanced Encryption Standard)

ALGORITMA SIMETRIS : BLOK CHIPER

AES (Advanced Encryption Standard)

- DES dianggap sudah tidak aman.
- Perlu diusulkan standard algoritma baru sebagai pengganti DES.
- National Institute of Standards and Technology (NIST) mengusulkan kepada Pemerintah Federal AS untuk sebuah standard kriptografi kriptografi yang baru.
- NIST mengadakan lomba membuat standard algoritma kriptografi yang baru. Standard tersebut kelak diberi nama Advanced Encryption Standard (AES).

AES (Advanced Encryption Standard)

- Pada bulan Oktober 2000, NIST mengumumkan untuk memilih Rijndael (dibaca: Rhine-doll)
- Pada bulan November 2001, Rijndael ditetapkan sebagai AES
- Diharapkan Rijndael menjadi standard kriptografi yang dominan paling sedikit selama 10 tahun.

AES (Advanced Encryption Standard)

- Tidak seperti *DES* yang berorientasi bit, *Rijndael* beroperasi dalam orientasi *byte.*
- Setiap putaran mengunakan kunci internal yang berbeda (disebut *round key*).
- *Enciphering* melibatkan operasi substitusi dan permutasi.
- Karena *AES* menetapkan panjang kunci adalah 128, 192, dan 256, maka dikenal *AES*-128, *AES*-192, dan *AES*-256

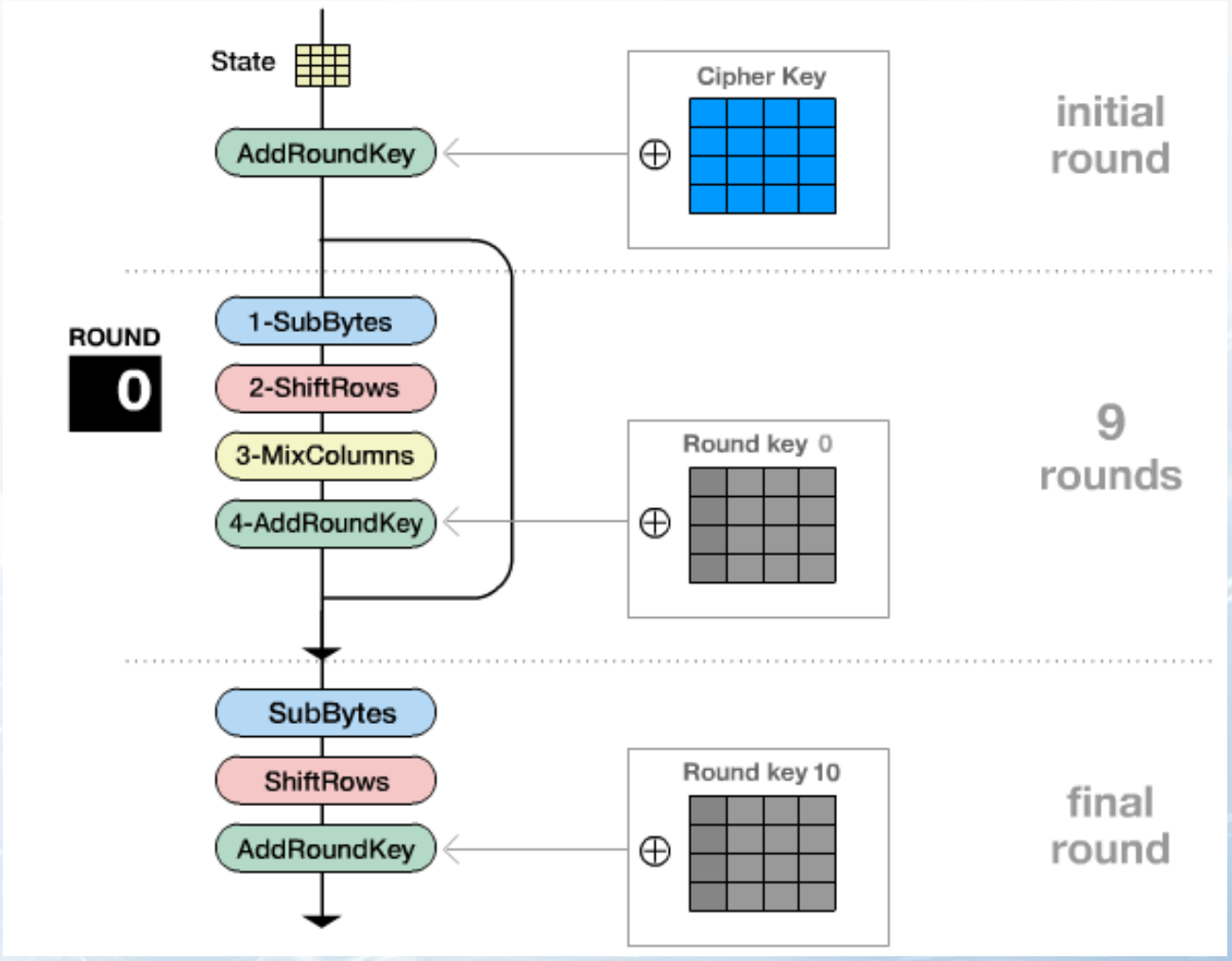
	Panjang Kunci (*Nk words*)	Ukuran Blok (*Nb words*)	Jumlah Putaran (*Nr*)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Catatan: 1 *word* = 32 bit

AES (Advanced Encryption Standard)

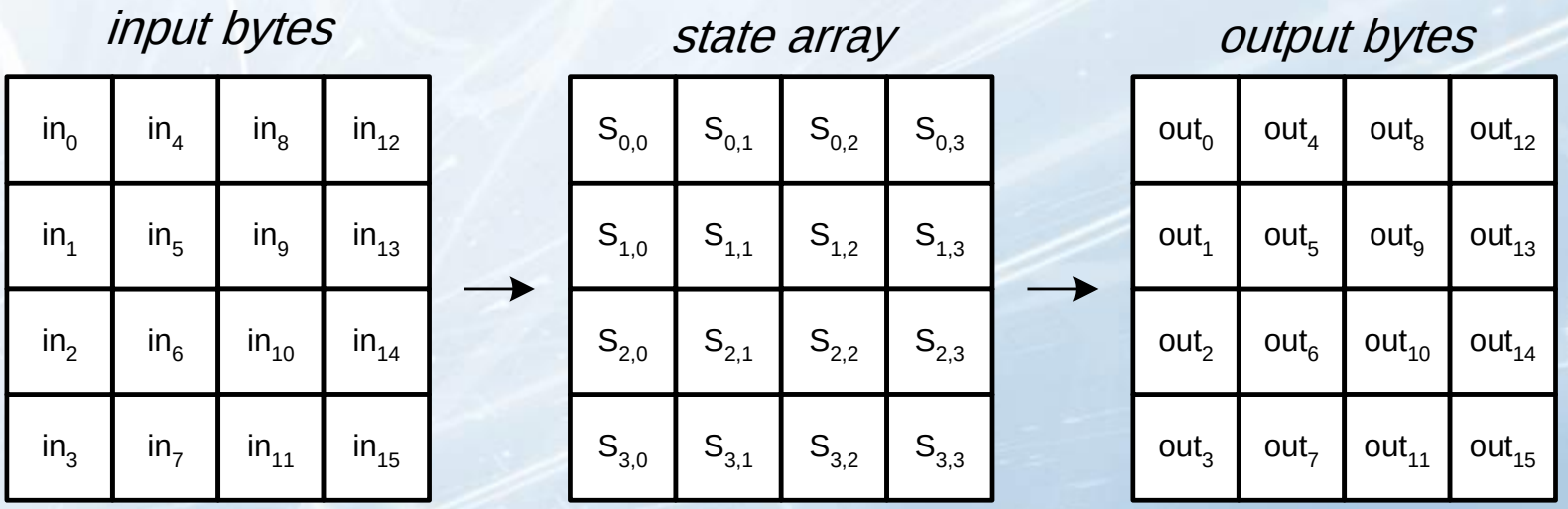
- Garis besar Algoritma *Rijndael* yang beroperasi pada blok 128-bit dengan kunci 128-bit adalah sebagai berikut (di luar proses pembangkitan *round key*):
  - *AddRoundKey*: melakukan *XOR* antara *state* awal (plainteks) dengan *cipher key*. Tahap ini disebut juga *initial round*.
  - Putaran sebanyak *Nr* – 1 kali. Proses yang dilakukan pada setiap putaran adalah:
    - *SubBytes*: substitusi *byte* dengan menggunakan tabel substitusi (*S-box*).
    - *ShiftRows*: pergeseran baris-baris *array state* secara *wrapping*.
    - *MixColumns*: mengacak data di masing-masing kolom *array state.*
    - *AddRoundKey*: melakukan XOR antara *state* sekarang *round key*.
  - *Final round*: proses untuk putaran terakhir:
    - *SubBytes*
    - *ShiftRows*
    - *AddRoundKey*

AES (Advanced Encryption Standard)



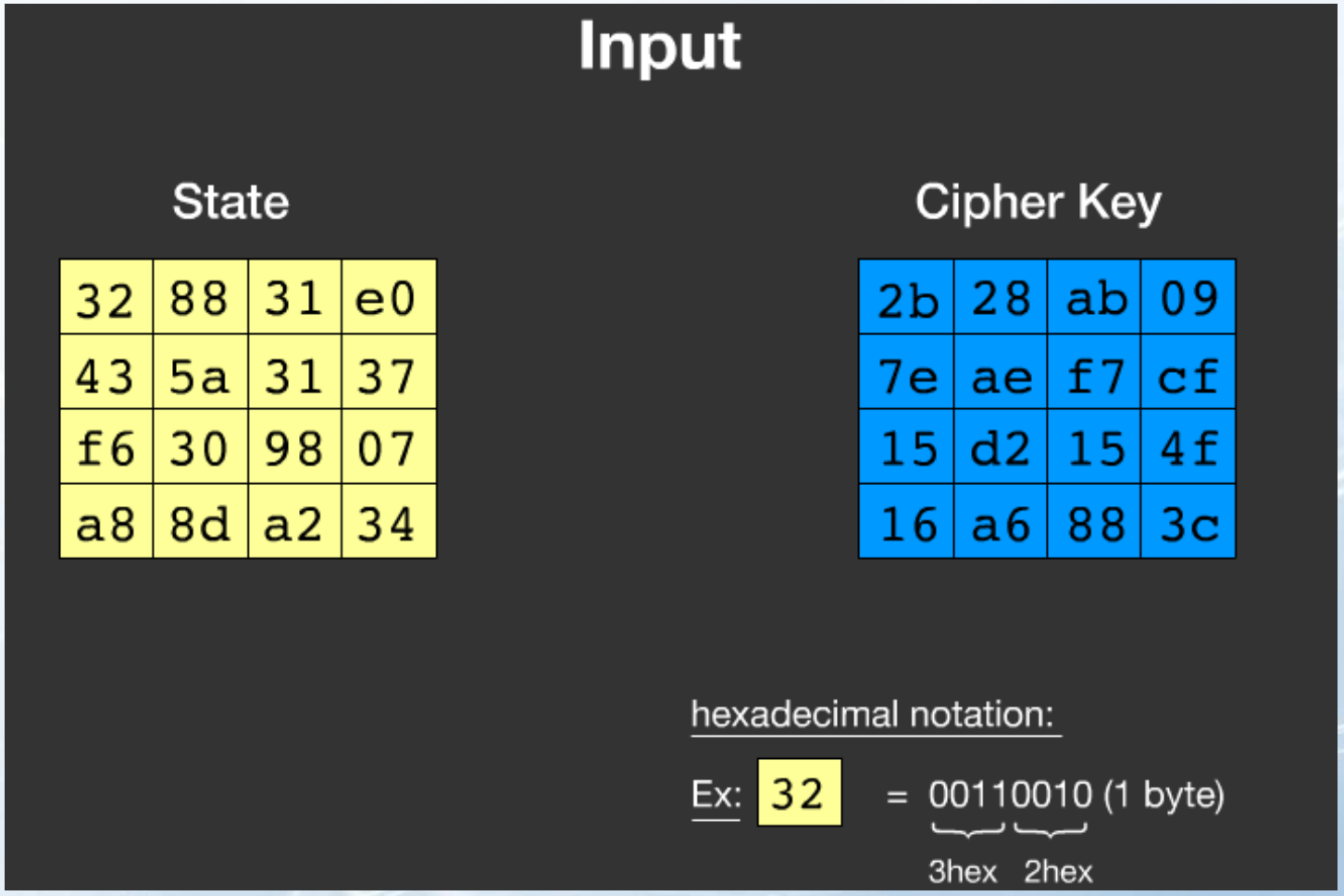
AES (Advanced Encryption Standard)

- Selama kalkulasi plainteks menjadi cipherteks, status sekarang dari data disimpan di dalam *array of bytes* dua dimensi, `state`, yang berukuran `NROWS` × `NCOLS`.
- Untuk blok data 128-bit, ukuran `state` adalah 4 × 4.
- Elemen *array* `state` diacu sebagai $S[r,c]$, $0 \le r < 4$ dan $0 \le c < Nb$ (Nb adalah panjang blok dibagi 32.
- Pada *AES*-128, $Nb = 128/32 = 4$)



AES (Advanced Encryption Standard)

- Contoh: (elemen state dan kunci dalam notasi HEX)



RSA

ALGORITMA ASIMETRIS

RSA

- Ditemukan oleh tiga orang yaitu **R**on Rivest, Adi **S**hamir, dan Leonard **A**dleman yang kemudian disingkat menjadi RSA.
- Termasuk algritma asimetri karena mempunyai dua kunci, yaitu kunci publik dan kunci privat.
- Algoritma kunci-publik yang paling terkenal dan paling banyak aplikasinya.
- Ditemukan oleh tiga peneliti dari MIT (Massachussets Institute of Technology), yaitu Ron Rivest, Adi Shamir, dan Len Adleman, pada tahun 1976.
- Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima.

RSA

Pembangkitan pasangan kunci

1. Pilih dua bilangan prima, a dan b (rahasia)
2. Hitung $n = a\,b$. Besaran n tidak perlu dirahasiakan.
3. Hitung $\phi(n) = (a-1)(b-1)$.
4. Pilih sebuah bilangan bulat untuk kunci publik, sebut namanya e, yang relatif prima terhadap $\phi(n)$.
5. Hitung kunci dekripsi, d, melalui $ed \equiv 1$ (**mod** m) atau $d \equiv e^{-1} \bmod (\phi(n))$

Hasil dari algoritma di atas:

- Kunci publik adalah pasangan (e, n)
- Kunci privat adalah pasangan (d, n)

Catatan: n tidak bersifat rahasia, namun ia diperlukan pada perhitungan enkripsi/dekripsi

RSA

Kunci Publik

- Misalkan $a = 47$ dan $b = 71$ (keduanya prima), maka dapat dihitung:

  $n = a \times b = 3337$

  $\phi(n) = (a-1)\times(b-1) = 46 \times 70 = 3220.$
- Pilih kunci publik $e = 79$ (yang relatif prima dengan 3220 karena pembagi bersama terbesarnya adalah 1).
- Hapus a dan b dan kunci publiknya adalah n=3337 dan e=79

Kunci Privat

- Selanjutnya akan dihitung kunci privat d dengan kekongruenan:

$$e \times d \equiv 1 \pmod{m} \Rightarrow \quad d = \frac{1 + (k \times 3220)}{79}$$

Dengan mencoba nilai-nilai $k = 1, 2, 3, \ldots$, diperoleh nilai d yang bulat adalah 1019. Ini adalah kunci privat (untuk dekripsi).

RSA

- Misalkan plainteks M = HARI INI
  atau dalam ASCII: 726582733273873

  Pecah M menjadi blok yang lebih kecil (misal 3 digit):

  m_1 = 726 m_4 = 273
  m_2 = 582 m_5 = 787
  m_3 = 733 m_6 = 003

  (Perhatikan, m_i masih terletak di dalam antara 0 sampai n – 1)

RSA

- *Enkripsi setiap blok:*

  $c_1 = 726^{79} \bmod 3337 = 215$

  $c_2 = 582^{79} \bmod 3337 = 776$, dst

  Chiperteks $C = 215\ 776\ 1743\ 933\ 1731\ 158$.

- *Dekripsi (menggunakan kunci privat d = 1019)*

  $m_1 = 215^{1019} \bmod 3337 = 726$

  $m_2 = 776^{1019} \bmod 3337 = 582$ dst untuk sisi blok lainnya

  Plainteks $M = 726582733273 7873$ yang dalam ASCII karakternya adalah HARI INI.

RSA

- ***Kekuatan dan Keamanan RSA***
  - Kekuatan algoritma *RSA* terletak pada tingkat kesulitan dalam memfaktorkan bilangan non prima menjadi faktor primanya, yang dalam hal ini $n = a \times b$.
  - Sekali n berhasil difaktorkan menjadi a dan b, maka $\phi(n) = (a - 1)\times(b - 1)$ dapat dihitung. Selanjutnya, karena kunci enkripsi e diumumkan (tidak rahasia), maka kunci dekripsi d dapat dihitung dari persamaan $ed \equiv 1 \pmod{n}$.
  - Penemu algoritma RSA menyarankan nilai a dan b panjangnya lebih dari 100 digit. Dengan demikian hasil kali n = a □ b akan berukuran lebih dari 200 digit.
  - Menurut Rivest dan kawan-kawan, usaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun! (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik).

Algoritma MD5

Fungsi HASH

MD5

- MD5 adalah fungsi hash satu-arah yang dibuat oleh Ron Rivest.
- MD5 merupakan perbaikan dari MD4 setelah MD4 berhasil diserang oleh kriptanalis.
- Algoritma MD5 menerima masukan berupa pesan dengan ukuran sembarang dan menghasilkan message digest yang panjangnya 128 bit.
- Dengan panjang message digest 128 bit, maka secara brute force dibutuhkan percobaan sebanyak 2128 kali untuk menemukan dua buah pesan atau lebih yang mempunyai message digest yang sama.

MD5 (Algoritma)

- Penambahan Bit-bit Pengganjal
  - Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 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

MD5 (Algoritma)

- 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

MD5 (Algoritma)

- Inisialisai Penyangga MD
  - *MD5* membutuhkan 4 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 4 × 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

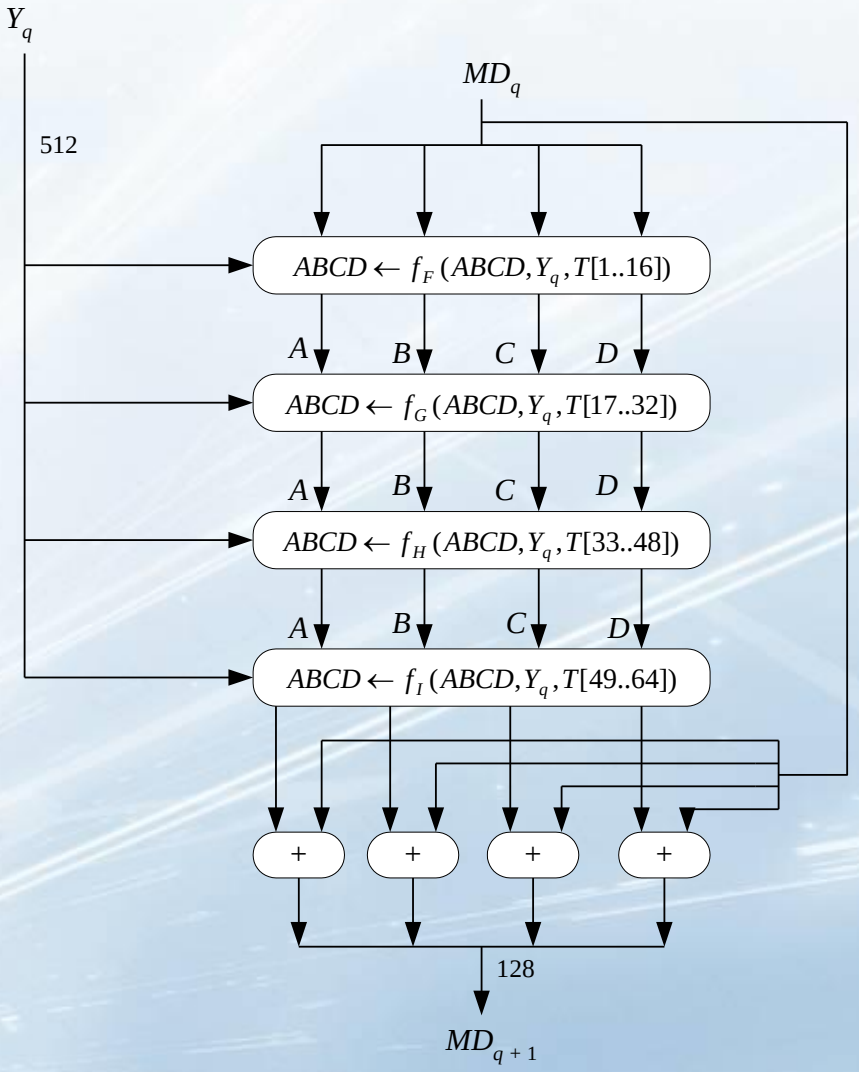
    D = 76543210

MD5 (Algoritma)

- 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}

MD5 (Algoritma)

- Yq :blok 512-bit ke-q dari pesan + bit-bit pengganjal + 64 bit nilai panjang pesan semula
- 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



MD5 (Algoritma)

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 $\wedge$, $\vee$, $\sim$, $\oplus$

MD5 (Algoritma)

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

MD5 (Algoritma)

- Putaran 1 : 16 kali operasi dasar dengan *g(b,c,d) = F(b,c,d)*

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]

MD5 (Algoritma)

- Putaran 2 : 16 kali operasi dasar dengan *g(b,c,d) = G(b,c,d)*

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]

MD5 (Algoritma)

- Putaran 3 : 16 kali operasi dasar dengan *g(b,c,d) = H(b,c,d)*

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]

MD5 (Algoritma)

- Putaran 4 : 16 kali operasi dasar dengan *g(b,c,d) = I(b,c,d)*

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]

MD5 (Algoritma)

- 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*.

Fungsi Hash

- Tujuan: pengalamatan di memori

- Bentuk: **h(k) = k mod m**
  - **m** : jumlah lokasi memori yang tersedia
  - **k** : kunci (integer)
  - **h(k)** : lokasi memori untuk record dengan kunci k

Contoh: m = 11 mempunyai sel-sel memori yang diberi indeks 0 sampai 10. Akan disimpan data *record* yang masing-masing mempunyai kunci 15, 558, 32, 132, 102, dan 5.

$h(15) = 15 \bmod 11 = 4$
$h(558) = 558 \bmod 11 = 8$
$h(32) = 32 \bmod 11 = 10$
$h(132) = 132 \bmod 11 = 0$
$h(102) = 102 \bmod 11 = 3$
$h(5) = 5 \bmod 11 = 5$

132			102	15	5			558		32
0	1	2	3	4	5	6	7	8	9	10