# BAB II

# MATERI PENUNJANG

## 2.1. Konsep Dasar Matematika

Sebelum menjelaskan tentang Kriptografi dan metode DES yang akan dibahas pada bab III perlu diketahui terlebih dahulu beberapa hal yang mendasari keduanya, yaitu konsep matematika.

### 2.1.1. Fungsi

Sebuah himpunan berisi obyek yang berbeda dengan syarat keanggotaan yang jelas dan obyek tersebut dinamakan anggota atau elemen dari himpunan.

Contoh :

Sebuah himpunan X dengan anggota a, b dan c, dinotasikan dengan X = {a, b, c}.

**Definisi fungsi**

*Sebuah fungsi didefinisikan sebagai dua buah himpunan* X , Y *dan sebuah aturan f yang memasangkan setiap elemen* X *dengan tepat satu elemen pada* Y.

Himpunan X disebut dengan domain fungsi dan himpunan Y sebagai kodomain .

**Definisi bayangan**

*Jika* $x \in X$ *maka bayangan dari* x *adalah elemen* $y \in Y$ *sesuai dengan aturan f , sedemikian sehingga y = f (x). Sebaliknya, setiap* $x \in X$ *disebut pra bayangan dari f (x)* $\in$ Y.

Im( *f* ) adalah himpunan dari seluruh bayangan $x \in X$, yang terletak di Y.

Jadi Im(*f*) adalah himpunan bagian dari Y.

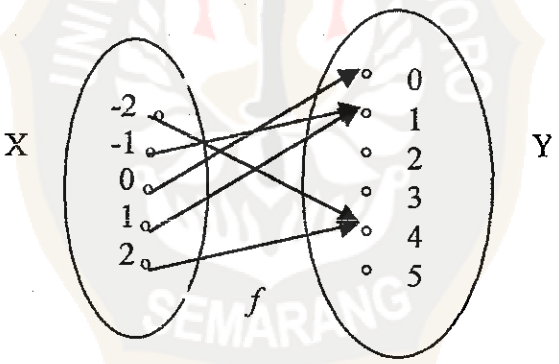
Contoh :

X = {a, b, c}, Y = {1, 2, 3, 4}, dan aturan $f$ didefinisikan $f(a) = 3, f(b) = 2, f(c) = 1$, maka pra bayangan dari elemen $2 \in Y$ adalah $b \in X$ dan Im($f$) adalah {1, 2, 3}.

Fungsi dapat pula dinyatakan dalam bentuk diagram fungsi, setiap elemen pada domain X memiliki tepat satu garis berarah yang berasal dari elemen tersebut. Setiap elemen pada kodomain Y bisa memiliki beberapa garis berarah yang menuju padanya termasuk garis kosong.

Contoh:

X = {-2, -1, 0, 1, 2}, Y = {0, 1, 2, 3, 4, 5}, dan aturan $f(x) = x^2$




Gambar 2.1. Sebuah fungsi dari himpunan X terhadap himpunan Y

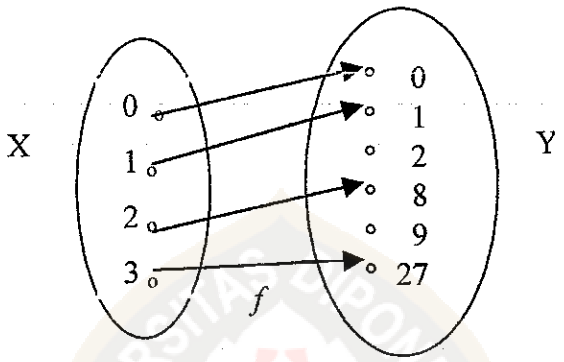
**Definisi fungsi satu-satu (1-1)**

*Sebuah fungsi dikatakan* 1 – 1 *(satu-satu) jika setiap 2 elemen yang berbeda pada domain* X *mempunyai bayangan yang berbeda pula elemen pada* Y.

Dengan kata lain, $f$: X→Y adalah fungsi satu-satu jika $f(x) = f(x^1)$ maka $x = x^1$. dan ekivalen jika $x \neq x^1$ maka $f(x) \neq f(x^1)$

Contoh:

X = (0, 1, 2, 3}, Y = {0, 1, 2, 8, 9, 27}, dan aturan $f(x) = x^3$ , secara eksplisit maka: $f(0) = 0$, $f(1) = 1$, $f(2) = 8$, $f(3) = 27$




Gambar 2.2. Sebuah fungsi satu-satu $f$

**Definisi fungsi onto**

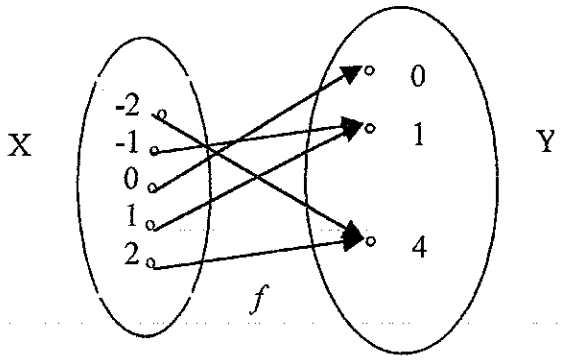
*Sebuah fungsi dikatakan onto jika setiap elemen pada kodomain Y adalah bayangan paling sedikit satu elemen pada domainnya.*

Ekivalen, sebuah fungsi $f : X \rightarrow Y$ adalah onto jika $\text{Im}(f) = Y$.

Contoh:

X = {-2, -1, 0, 1, 2}, Y = {0, 1, 4,}, dan aturan $f(x) = x^2$ , secara eksplisit maka:

$F(-2) = 4, f(-1) = 1, f(0) = 0, f(1) = 1, f(2) = 4$




Gambar 2.3. Sebuah fungsi onto $f$

**Definisi fungsi bijeksi**

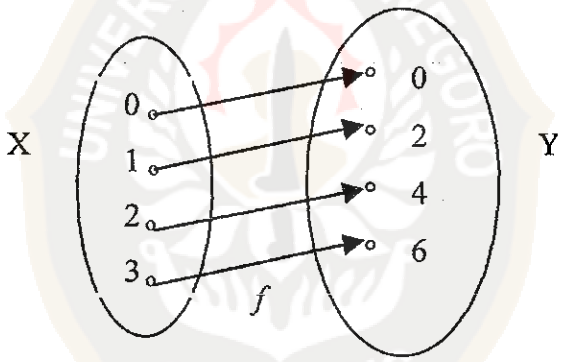
*Jika sebuah fungsi* $f : X \rightarrow Y$ *adalah* $1 - 1$ *dan* $\text{Im}(f) = Y$, *maka* $f$ *dikatakan fungsi bijeksi.*

Disamping itu jika $f : X \rightarrow Y$ merupakan fungsi $1 - 1$ serta X dan Y adalah himpunan berhingga dengan ukuran yang sama, maka $f$ dikatakan bijeksi pula.

Contoh:

$X = \{0, 1, 2, 3\}$, $Y = \{0, 2, 4, 6\}$, dan aturan $f(x) = 2x$, secara eksplisit maka:

$f(0) = 0, f(1) = 2, f(2) = 4, f(3) = 6$




Gambar 2.4. Sebuah fungsi bijeksi $f$

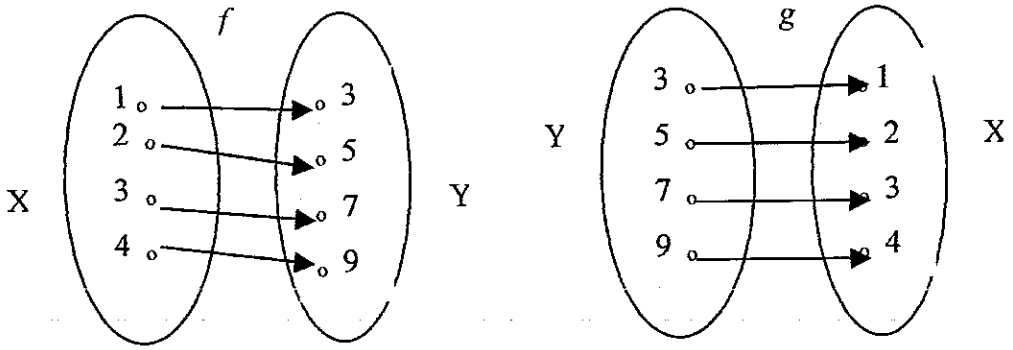
**Definisi fungsi invers**

*Jika f adalah fungsi bijeksi dari* X *terhadap* Y *maka dapat didefinisikan sebuah fungsi g dari* Y *ke* X *sebagai berikut : untuk setiap* $y \in Y$ *didefinisikan fungsi* $g(y) = x$ *dengan* $x \in X$ *dan* $f(x) = y$. *Fungsi g ini disebut fungsi invers dari f .*

Contoh:

$X = \{1, 2, 3, 4\}$, $Y = \{3, 5, 7, 9\}$, dan aturan $f(x) = 2x + 1$, secara eksplisit maka:

$f(1) = 3,\ f(2) = 5,\ f(3) = 7,\ f(4) = 9$




Gambar 2.5. Sebuah fungsi bijeksi $f$ dan inversnya $g = f^{-1}$

Dengan melihat Gambar 2.5. maka diketahui bahwa $f$ adalah fungsi bijeksi dan fungsi inversnya yaitu $g$ dibentuk dengan mensubstitusi persamaan $f(x)$.

$$2x + 1 = y$$
$$2x = y - 1$$
$$x = \frac{y-1}{2}$$

maka $g(y) = \frac{y-1}{2}$, secara eksplisit: $g(3) = 1, g(5) = 2, g(7) = 3, g(9) = 4$.

Catatan; jika fungsi $f$ merupakan fungsi bijeksi maka inversnya ( $f^{-1}$ ) juga merupakan fungsi bijeksi. "*Dalam kriptografi fungsi bijeksi digunakan sebagai alat untuk mengenkripsi pesan plaintext dan fungsi invers digunakan untuk mendekripsi sandi* ".

**Definisi fungsi satu arah**

*Sebuah fungsi* $f : X \rightarrow Y$ *dikatakan sebagai fungsi satu arah jika* $f(x)$ "mudah" *untuk dihitung untuk semua* $x \in X$ *akan tetapi untuk* "hampir seluruh elemen $y \in \text{Im}(f)$" *secara perhitungan invisibel untuk didapatkan nilai* $x \in X$ *sedemikian sehingga* $f(x) = y$.

Untuk memperoleh kejelasan tentang *definisi fungsi satu arah* ini, akan diilustrasikan dengan contoh berikut ini

Contoh :

$X = \{1, 2, 3, 4, \ldots, 16\}$ dan ditentukan $f(x) = r_x = 3^x \bmod 15$ untuk semua $x \in X$.

Secara eksplisit dapat diperlihatkan tabel berikut ini :

| x | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f(x)$ | 3 | 9 | 12 | 6 | 3 | 9 | 12 | 6 | 3 | 9 | 12 | 6 | 3 | 9 | 12 | 6 |

Untuk setiap $1 \leq x \leq 16$, $x \in B$, maka relatif mudah untuk menemukan bayangan dari setiap bilangan dengan fungsi yang telah didefinisikan tersebut, akan tetapi apabila diberikan sebuah bilangan misalnya $f(x) = 7$, tanpa melihat pada tabel maka akan dirasakan cukup sulit untuk mengetahui harga x sedemikian sehingga $f(x) = 7$. Tentu saja apabila diberikan bilangan $f(x) = 3$ maka dengan mudah harga dari $x = 1$, akan tetapi hampir untuk semua bilangan pada kodomain yang diberikan tidak mudah untuk menghitung harga x yang bersesuaian tanpa menggunakan alat bantu berupa tabel tersebut.

Pada contoh tersebut masih menggunakan bilangan yang kecil, hal penting dalam contoh di atas adalah perbedaan proses untuk menghitung $f(x)$ dan untuk mendapatkan harga dari x apabila diberikan nilai dari $f(x)$, bahkan untuk bilangan yang sangat besar $f(x)$ dapat dihitung, tetapi proses untuk menemukan harga x dari $f(x)$ yang diketahui akan membutuhkan waktu yang lebih lama.

### 2.1.1.1. Permutasi

Permutasi adalah fungsi yang sering digunakan dalam berbagai bentuk metode kriptografi, karena merupakan pemetaan suatu himpunan terhadap dirinya sendiri.

**Definisi permutasi**

S *adalah sebuah himpunan berhingga, permutasi p pada* S *adalah fungsi bijeksi dari S terhadap dirinya sendiri* ($p$ : S→S)

Contoh :

Himpunan S = {1,2,3,4,5}. Sebuah permutasi $p$ : S→S didefinisikan sebagai berikut :

$p(1) = 5, p(2) = 3, p(3) = 4, p(4) = 1, p(5) = 2.$

sebuah permutasi dapat dideskripsikan dengan berbagai cara, selain seperti dinyatakan di atas dapat dinyatakan pula sebagai sebuah array berikut ini :

$$p = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 3 & 4 & 1 & 2 \end{pmatrix}$$

baris pertama dalam array merupakan domain sedangkan baris kedua merupakan bayangan sesuai dengan pemetaan oleh $p$.

Oleh karena sebuah permutasi merupakan fungsi bijeksi maka memiliki invers. Jika sebuah permutasi dinyatakan sebagai array seperti di atas, maka untuk mengetahui inversnya yaitu dengan cara menukarkan baris-baris dalam array dan menyusun elemen-elemen baris pertama yang baru disesuaikan dengan pasangannya. Sehingga invers dari $p$ pada contoh di atas adalah :

$$p^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 5 & 2 & 3 & 1 \end{pmatrix}$$

### 2.1.1.2. Involusi

Tipe fungsi yang lain yang berkaitan dengan komposisi fungsi adalah involusi, involusi ini memiliki sifat bahwa inversnya adalah dirinya sendiri.

**Definisi involusi**

S *adalah himpunan berhingga dan f merupakan fungsi bijeksi dari* S *terhadap* S (*yaitu*, $f$: S→S). *Fungsi f disebut involusi jika* $f = f^{-1}$.

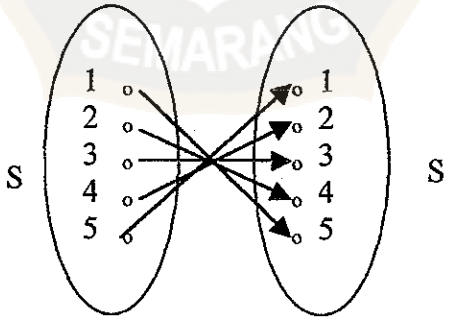
Hal ini ekivalen dengan pernyataan $f(f(x)) = x$ untuk semua $x \in S$.

Contoh :

S = {1, 2, 3, 4, 5} dan aturan $f(x) = 6 - x$, $x \in S$ secara eksplisit maka:

$f(1) = 5$, $f(2) = 4$, $f(3) = 3$, $f(4) = 2$, $f(5) = 1$. Maka inversnya $g(y) = 6 - y$, yang secara eksplisit sama dengan $f(x)$.

*Gambar 2.6.* berikut ini merupakan contoh dari involusi. Dalam fungsi involusi, bahwa jika *j* merupakan bayangan *i* maka *i* juga merupakan bayangan dari *j*.




Gambar 2.6. Involusi himpunan S dengan 5 buah elemen

### 2.1.2. Teori Bilangan

#### 2.1.2.1. Integer

Himpunan dari semua bilangan bulat {...,-3,-2,-1,0,1,2,3,...} dinyatakan dengan simbol $\mathbb{Z}$.dan dinamakan himpunan bilangan integer.

**Definisi bilangan pembagi**

*Jika a,b dan c adalah integer dan b = ac maka a disebut pembagi atau divisor dari b, atau a faktor dari b dan dinotasikan sebagai* $a \mid b$.

Contoh :

a. $-4 \mid 28$, karena $28 = (-4)(-7)$.

b. $17 \mid 0$, karena $0 = (17)(0)$.

**Definisi pembagian integer**

*Jika a dan b adalah integer dengan* $b \geq 1$, *maka pembagian biasa bilangan a oleh b menghasilkan q dan sisa r sedemikian sehingga*

$$a = qb + r, \quad \text{dengan } 0 \leq r < b.$$

*Sisa dari pembagian dinyatakan dengan* ***a mod b****, dan hasil bagi dinyatakan dengan* ***a Div b****.*

Misal $a,b \in \mathbb{Z}$ dengan $b \neq 0$, maka $a \text{ Div } b = \lfloor a/b \rfloor$ dan $a \bmod b = a - b\lfloor a/b \rfloor$.

Contoh :

a = 75, b = 13, maka q = 5 dan r = 10, karena 75 mod 13 = 10 sedangkan 75 Div 13 = 5

**Definisi faktor persekutuan**

*Sebuah integer c disebut faktor persekutuan (common divisor) dari a dan b jika* $c \mid a$ *dan* $c \mid b$.

**Definisi faktor persekutuan terbesar (gcd)**

*Sebuah integer non negatif d disebut faktor persekutuan terbesar (greatest common divisor) dari integer a dan b, dinyatakan dengan d = gcd(a,b), jika*

*(i) d adalah common divisor a dan b*

*(ii) bilamana* $c \mid a$ *dan* $c \mid b$, *maka* $c \mid d$

ekivalen, gcd(a,b) adalah integer positif terbesar yang membagi baik a maupun b, dengan pengecualian yaitu gcd(0,0) = 0

Contoh :

Common divisor dari 14 dan 21 adalah {±1,±2, ±7}, dan gcd (14,21) = 7

**Definisi kelipatan persekutuan terkecil (lcm)**

*Sebuah integer non negatif d disebut kelipatan persekutuan terkecil (least common multiple) dari integer a dan b, ditulis d = lcm(a,b), jika*

*(i)* $a \mid d$ *dan* $b \mid d$

*(ii) bilamana* $a \mid c$ *dan* $b \mid c$, *maka* $d \mid c$

secara ekivalen, lcm(a,b) adalah integer non negatif terkecil yang dapat dibagi oleh a dan b.

Jika a dan b integer positif, maka lcm(a,b) = a . b/ gcd(a,b).

Contoh :

gcd(14,21) = 7 maka lcm(14,21) = 14 . 21/7 = 42

**Definisi relatif prima**

*Dua buah integer a dan b dikatakan* ***relatif prima*** *(relatively prime) atau* ***koprima*** *(coprime) jika gcd(a,b) = 1*

**2.1.2.2. Integer Modulo n**

Diambil n adalah sebuah integer positif

**Definisi kongruen**

*Jika a dan b merupakan integer, maka a dikatakan* ***kongruen*** *terhadap b modulo n, ditulis* $a \equiv b \pmod{n}$ *jika* $n \mid (a-b)$. *Integer n disebut modulus kongruensi.*

Contoh :

(i) $124 \equiv 9 \pmod 5$ karena $124 - 9 = 23 \cdot 5$

(ii) $-13 \equiv 17 \pmod 5$ karena $-13-17 = -6 \cdot 5$

**Definisi Integer modulo n**

*Integer modulo n, ditulis* $\mathbb{Z}_n$ *adalah himpunan yang anggota-anggotanya adalah integer {0,1,2,3,…,n-1}. Operasi penjumlahan, pengurangan dan perkalian dalam* $\mathbb{Z}_n$ *dibentuk oleh modulo n.*

Contoh :

$\mathbb{Z}_{25} = \{1,2,3,\ldots,24\}$. Dalam $\mathbb{Z}_{25}$, $23 + 16 = 14$, karena $23 + 16 = 39 \equiv 14 \pmod{25}$.

Begitu pula untuk operasi perkalian $23 \cdot 16 = 18$ dalam $\mathbb{Z}_{25}$.

**Definisi invers bilangan modulo n**

*$a \in \mathbb{Z}_n$, invers perkalian dari a modulo n adalah integer $x \in \mathbb{Z}_n$ sedemikian sehingga $ax \equiv 1 \pmod{n}$. Jika x tersebut ada, maka a dikatakan dapat diinverskan, invers dari a dituliskan $a^{-1}$.*

Misal $a \in \mathbb{Z}_n$, maka a dapat diinverskan jika dan hanya jika gcd(a,n) = 1.

**Definisi pembagian bilangan modulo n**

*$a,b \in \mathbb{Z}_n$, pembagian a oleh b modulo n adalah hasil kali dari a dan $b^{-1}$ modulo n, dan hanya didefinisikan jika b dapat diinverskan modulo n.*

Contoh :

Elemen yang dapat diinverskan dalam $\mathbb{Z}_9$ adalah 1, 2, 4, 5, 7, dan 8. Misalnya $4^{-1}$ = 7, karena $4 \cdot 7 \equiv 1 \pmod{9}$

### 2.1.3. Bilangan Biner

**Definisi basis (radix) sistem bilangan**

*Basis atau radix suatu sistem bilangan adalah menyatakan banyaknya lambang yang dipergunakan dalam sistem bilangan tersebut.*

$L_n = \{l_1, l_2, l_3, \ldots, l_n\}$ adalah himpunan lambang yang digunakan dalam sistem bilangan basis n.

Contoh:

Sistem bilangan desimal mempunyai basis atau radix sepuluh (10) karena sistem ini menggunakan 10 lambang angka, yaitu: $L_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9.\}$

Untuk menyajikan notasi lambang bilangan yang lebih dari basisnya maka dibentuk kombinasi lambang dari lambang dasar dalam sistem bilangan.

$l_2l_1, l_2l_2, \ldots l_2ln, l_3l_1, l_3l_2, l_3l_3, \ldots l_2l_n, \ldots l_nl_n, l_2l_1l_1, l_2l_1l_2, \ldots$

Contoh:

Setelah mencapai nilai 9 pada sistem bilangan desimal, kemudian dibentuk kombinasi angka-angka desimal untuk memperoleh 10, 11, 12, dan seterusnya. Dengan kata lain, bilangan desimal berikutnya setelah 9 diperoleh dengan menggunakan angka kedua diikuti oleh angka pertama untuk memperoleh 10. Bilangan desimal sesudah 10 diperoleh dengan menggunakan angka kedua untuk mendapatkan 11, dan seterusnya.

**Definisi bilangan biner**

*Bilangan biner ialah sistem bilangan yang mempunyai basis atau radix dua, dengan menggunakan lambang 0 dan 1. Dan dinotasikan dengan ß.*

Dalam sistem bilangan biner digunakan pendekatan yang sama dalam menyajikan lambang yang lebih dari dua. Setelah mencapai lambang 1, maka lambang biner telah habis (tidak ada 2, 3, ... dalam sistem bilangan biner). Untuk menyatakan nilai dua, gunakan angka biner kedua diikuti oleh angka biner pertama untuk mendapatkan 10. Untuk menyatakan nilai tiga, gunakan 11. Oleh karenanya dalam biner pencacahannya sebagai berikut: 0, 1, 10, 11. Untuk menghindari kekacauan dengan bilangan-bilangan desimal, ada baiknya untuk membaca bilangan-bilangan biner ini dengan nol, satu, satu-nol, dan satu-satu.

Selanjutnya dapat dibentuk berbagai sistem bilangan lainya dengan memilih kelompok-kelompok lambang dasar atau angka lain. Sebagai contoh, sistem bilangan *heksadesimal*, yang segera akan dibahas, mempunyai basis sebesar 16.

**2.1.4. Bilangan Heksadesimal**

Sistem bilangan heksadesimal mempunyai basis 16. Walaupun dapat digunakan 16 buah lambang yang manapun, namun lazimnya digunakan 0 sampai 9 dan A sampai F seperti terlihat pada tabel 2.1. Setelah mencapai 9 pada sistem heksadesimal, pencacahan dilanjutkan sebagai berikut:

A, B, C, D, E, F

Setelah kehabisan lambang dasar, maka dibentuk kombinasi 2-angka, dengan mengambil angka kedua diikuti oleh angka pertama, kemudian kedua diikuti oleh angka kedua, dan seterusnya. Maka bilangan berikutnya setelah F dalam heksadesimal adalah 10.

Kemudian dikuti oleh

11, 12, 13, 14, 15, 16, 17, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20, 21, dan seterusnya.

Ada baiknya untuk memahami Tabel 2.1. Salah satu bidang pengembangan yang paling luas dewasa ini adalah mikrokomputer. Pada saat memprogram, menganalisa, maupun memeriksa sebuah komputer, akan dibutuhkan bilangan heksadesimal. Disamping itu juga harus menguasai perubahan-perubahan heksadesimal sebagai berikut.

**Tabel 2.1. Konversi desimal, biner dan heksadesimal**

| Desimal | Biner | Heksadesimal |
|---|---|---|
| 0 | 0000 | 0 |
| 1 | 0001 | 1 |
| 2 | 0010 | 2 |
| 3 | 0011 | 3 |
| 4 | 0100 | 4 |
| 5 | 0101 | 5 |
| 6 | 0110 | 6 |
| 7 | 0111 | 7 |
| 8 | 1000 | 8 |
| 9 | 1001 | 9 |
| 10 | 1010 | A |
| 11 | 1011 | B |
| 12 | 1100 | C |
| 13 | 1101 | D |
| 14 | 1110 | E |
| 15 | 1111 | F |

### 2.1.5. Konversi Antar Basis

Komputer akan melakukan proses perhitungan dalam bentuk biner akan tetapi dari sudut pandang pengguna komputer mungkin melakukan proses perhitungan dalam bentuk desimal. Oleh sebab itu dalam komputer terdapat mekanisme konversi bilangan yang memudahkan komunikasi antara komputer digital dengan pengguna komputer tersebut.

Usaha mengkonversi bilangan tersebut dapat dilakukan dengan metode pembagian yang berdasarkan pada aritmatika modular, proses pembagian tersebut bersifat rekursif hal ini dapat nyatakan dalam algoritma berikut ini :

- n adalah sebuah bilangan (sesuai dengan basis asal , misalnya basis 10 )
- Jika (n > 0) maka lakukan proses berikut :
  1. Lakukan pembagian pada n oleh basis yang dinginkan (basis 2)
  2. Hasilnya adalah sisa pembagian : (n mod 2)
  3. n⇐Hasil bagi : (nDiv 2)
     selanjutnya proses berulang sampai diperoleh bilangan yang dimaksud

dalam konversi tersebut.

Contoh :

Akan dikonversikan bilangan $100_{10}$ terhadap basis 2 :

| Perhitungan | Output |
|---|---|
| N⇐100 | |
| | (100 mod 2)⇒0 |
| N⇐(100 Div 2) = 50 | |
| | ( 50 mod 2) ⇒0 |
| N⇐(50 Div 2) = 25 | |
| | (25 mod 2) ⇒1 |
| N⇐(25 Div 2) = 12 | |
| | (12 mod 2) ⇒0 |
| N⇐(12 Div 2) = 6 | |
| | (6 mod 2) ⇒0 |
| N⇐(6 Div 2) = 3 | |
| | (3 mod 2) ⇒1 |
| N⇐(3 Div 2) = 1 | |
| | (1 mod 2) ⇒1 |
| N⇐(1 Div 2) = 0 | |

Berdasarkan algoritma rekursif ini, maka hasil konversi binernya adalah $1100100_2$ , dengan penulisan terbalik yaitu dari hasil output paling bawah sampai pada output paling atas.

### 2.1.6. Bit *(Binary digit)*

**Definisi bit (*binary digit*)**

*Bit ialah jumlah dari angka biner dalam satu kesatuan. Kata ini merupakan singkatan dari binary digit (angka biner).*

Contoh:

Bilangan biner seperti 1101 mempunyai 4 angka biner, atau 4 bit. Bilangan 111010 mempunyai 6 angka biner, atau 6 bit.

#### 2.1.6.1. Bit Paritas

Dalam sistem digital *kata (word)* adalah sekelompok bit yang diperlakukan, disimpan, dan dipindahkan sebagai suatu kesatuan. Sebagai

contoh, misalkan sebuah komputer akan menambahkan 0101 1000 0011 dan 0010 0100 0110. (Ini adalah 586 dan 246) Masing-masing bilangan ini merupakan sebuah kata. Komputer membawa masing-masing kata ini keluar dari memori dan meletakkannya ke dalam suatu aritmatika. Jumlahnya merupakan sebuah kata baru yang kemudian diletakkan kembali ke dalam memori.

Pada saat kata-kata sedang dipindahkan dan disimpan, kesalahan dapat masuk ke dalam kata-kata tersebut. Sebagai contoh, salah satu 0 dalam suatu kata secara tidak disengaja mungkin berubah menjadi 1 akibat pemutusan sesaat, akibat derau, akibat peralihan, dan sebagainya. Di bawah kondisi operasi normal perubahan semacam itu tidak mungkin terjadi, namun suatu kesalahan dapat merusak segalanya. Oleh karenanya, dibutuhkan metode-metode untuk mendeteksi kesalahan yang timbul pada saat suatu kata sedang dipindahkan atau disimpan.

Pendekatan yang paling luas penggunaannya untuk mendeteksi kesalahan yang timbul selama penyimpanaan dan pemindahan kata-kata adalah membubuhkan suatu *bit paritas* pada kata tersebut.

**2.1.6.1.1. Paritas Genap dan Ganjil**

**Definisi paritas genap (even parity)**

*Paritas genap ialah membubuhkan sebuah bit tambahan kepada sekelompok bit supaya menghasilkan jumlah bit 1 pada kelompok tersebut genap.*

Contoh:

Misalkan diberikan sebuah kata seperti 0111. Di sini terdapat tiga buah 1 dalam kata ini (banyaknya 1 yang ganjil). Kemudian dibubuhkan sebuah 1 tambahan pada kata tersebut untuk mendapatkan 0111 1. Sekarang banyaknya 1 menjadi genap. Kata baru ini dapat dipindahkan dan disimpan oleh komputer, dan dapat diperiksa paritas genapnya pada berbagai titik untuk menyakinkan bahwa tidak ada kesalahan yang telah memasuki kata tersebut.

Sebagai gambaran lain bagi paritas genap, Tabel 2.2. memperlihatkan sandi heksadesimal dengan sebuah bit paritas-genap. Bit paritas menghasilkan banyaknya 1 yang genap bagi masing-masing kelompok sandi.

**Definisi paritas ganjil (odd parity)**

*Paritas ganjil ialah bit paritas tambahan yang membuat jumlah bit 1 menjadi ganjil pada sekelompok bit.*

Tabel 2.2. memperlihatkan sandi heksadesimal dengan sebuah bit paritas ganjil, perhatikan bahwa bit paritas-ganjil selalu merupakan komplemen bagi bit paritas-genap.

**Tabel 2.2. Bit paritas**

| Heksadesimal | Bit tambahan | |
|---|---|---|
| | Paritas genap | Paritas ganjil |
| 0000 | 0 | 1 |
| 0001 | 1 | 0 |
| 0010 | 1 | 0 |
| 0011 | 0 | 1 |
| 0100 | 1 | 0 |
| 0101 | 0 | 1 |
| 0110 | 0 | 1 |
| 0111 | 1 | 0 |
| 1000 | 1 | 0 |
| 1001 | 0 | 1 |
| 1010 | 0 | 1 |
| 1011 | 1 | 0 |
| 1100 | 0 | 1 |
| 1101 | 1 | 0 |
| 1110 | 1 | 0 |
| 1111 | 0 | 1 |

Kedua jenis paritas ini biasa digunakan, dan tidak ada alasan yang kuat untuk lebih memilih salah satu jenis.

Penggunaan bit paritas untuk mendeteksi kesalahan berdasarkan dua asumsi yang berlaku pada kebanyakan sistem digital:

- Probabilitas kesalahan sangat kecil.

  Dalam hal suatu kesalahan masuk ke dalam sebuah kata, kesalahan tersebut hampit dapat dipastikan merupakan kesalahan 1-bit. Kemungkinan 2 atau lebih bit berubah secara tak disengaja sangatlah kecil kecuali jika terjadi

gangguan total, yang akan terdeteksi oleh sarana lain. Dengan kata lain pemeriksaan paritas akan menangkap semua kesalahan 1-bit namun tidak menangkap kesalahan ganda. Dalam kebanyakan sistem digital kemungkinan terjadi kesalahan ganda sangat kecil.

- Pemeriksaan paritas khususnya biasa dilakukan dalam piranti penyimpanan seperti disket, pita, memori inti magnetis.

**2.1.7. Ukuran Bilangan**

Sistim penyimpanan data dalam komputer digital adalah dalam satuan bit atau byte (1 byte sama dengan 8 bit).

Logaritma adalah pengukuran terhadap besarnya sebuah bilangan, pemakaian logaritma basis 10 atau logaritma basis e (natural) lebih dikenal bagi kebanyakan orang. Tetapi dalam ilmu komputer penggunaan logaritma basis 2 merupakan hal yang mendasar, karena sebuah logaritma adalah ukuran terhadap jumlah digit yang diperlukan oleh komputer untuk merepresentasikan sebuah nilai dari basis tertentu. Berikut ini adalah tabel yang memperlihatkan bilangan terbesar yang dapat diwakili oleh digit biner :

**Tabel 2.3. Ukuran Bilangan**

| Digit | Biner | Ekivalen dalam desimal |
|---|---|---|
| 1 | $1=2^1-1$ | 1 |
| 2 | $11=2^2-1$ | 3 |
| 3 | $111=2^3-1$ | 7 |
| 4 | $1111=2^4-1$ | 15 |
| 5 | $11111=2^5-1$ | 31 |
| 6 | $111111=2^6-1$ | 63 |

Bilangan yang cukup penting dalam satuan penyimpanan data diantaranya adalah $2^{10}$ = 1024, oleh karena bilangan tersebut mendekati 1000, para pakar komputer sepakat bilangan tersebut sebagai satuan Kilobit disingkat **K**, misalnya data / file komputer dengan kapasitas 2048 bit dapat ditulis 2 **K**.

### 2.1.8. Operasi Logika

Operasi logika digunakan oleh komputer terutama pada bahasa pemrograman tingkat tinggi misalnya Pascal, Basic, C dan sebagainya. Operasi logika berguna dalam menentukan proses-proses yang dilakukan oleh komputer digital yaitu untuk pengetesan kondisi sehingga prosessor komputer dapat mengeksekusi suatu program aplikasi atau membatalkan suatu perintah eksekusi. Berikut ini adalah tabel dari operasi logika AND, OR, XOR dan NOT yang biasa dipakai dalam proses enkripsi data :

**Tabel 2.3. Operasi logika AND**

| P | Q | $P \wedge Q$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 0 |

**Tabel 2.4. Operasi logika OR**

| P | Q | $P \vee Q$ |
|---|---|---|
| 1 | 1 | 1 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

**Tabel 2.5. Operasi logika NOT**

| P | $\sim P$ |
|---|---|
| 1 | 0 |
| 0 | 1 |

**Tabel 2.6. Operasi logika X-OR**

| P | Q | $P \oplus Q$ |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

## 2.2. File

**Definisi item (elemen data)**

*Item atau elemen data ialah nilai dari suatu data yang mempunyai tipe tertentu, seperti integer, real, karakter, atau tipe yang terstruktur.*

**Definisi record**

*Record adalah struktur data yang tersusun dari satu atau lebih item (elemen data) yang mewakili suatu data.*

**Definisi File**

*File adalah kumpulan dari berkas (record) yang mempunyai hubungan satu sama lain.*

Contoh :

Kartu Mahasiswa bisa dikatakan sebuah record yang terdiri dari item-item:

Nama, Nim, Jurusan, Alamat, Golongan darah. Sedangakan File yang menghimpunnya adalah data mahasiswa pada suatu Fakultas.

## 2.3. Pengantar Kriptografi

**Definisi Kriptografi**

*Kriptografi (cryptography) adalah kajian/studi tentang teknik-teknik bersifat matematis yang berkaitan dengan aspek keamanan data atau informasi.*

Dalam kriptografi digunakan metode atau sistem, yang disebut kriptosistem (cryptosystem). Di dalam sebuah kriptosistem, pengirim (sender) mengubah pesan sebelum mengirimkannya, sehingga diharapkan hanya penerima (receiver) yang berhak untuk bisa menyusun ulang pesan yang asli yakni pesan

yang belum diubah (plaintext atau cleartext). Kegiatan pengirim pesan ini disebut menyandikan atau mengkriptosasi (*encrypt*) pesan, sedangkan pesan yang terenkripsi disebut ciphertext dan penerima pesan dikatakan melakukan penguraian sandi atau dekriptosasi (*decrypt*) pesan. Jika kriptosistem ini aman, orang-orang yang tidak berhak tidak akan bisa menemukan teknik dekripsi, sehingga meskipun mereka membaca pesan yang telah dikriptosasi, mereka tidak akan bisa melakukan dekriptosasi pesan itu.

Kriptografi dilakukan oleh seorang kriptografer (*cryptographers*), sedangkan kripta analis (*cryptanalysis*) adalah kajian/studi terhadap teknik-teknik untuk memecahkan atau membongkar metode kriptografi.

Pada salah satu sistem tertua dan paling sederhana, pengirim dan penerima masing-masing mempunyai sebuah kunci yang mendefinisikan sebuah karakter pengganti untuk setiap karakter potensial yang dikirimkan. Lagipula, pengirim dan penerima tidak memperlihatkan kuncinya. Kunci seperti itu dikatakan kunci private.

Plaintext dinotasikan dengan M (*Message*) atau P (*Plaintext*), plaintext berupa text. M secara sederhana adalah data biner. Ciphertext dinotasikan dengan C yang juga merupakan data biner. Fungsi enkripsi E, mengoperasikan M (sebagai input) sehingga diperoleh C, secara matematis dapat dinotasikan sebagai berikut :

$$E(M) = C$$

Sedangkan kebalikannya adalah fungsi dekripsi :

$$D(C) = M$$

Karena proses enkripsi dan dekripsi adalah untuk memperoleh kembali (*recover*) original plaintext, maka harus memenuhi identitas berikut :

$$D(E(M)) = M$$

Kritografi modern menyelesaikan masalah ini dengan sebuah kunci (*key*), dinotasikan dengan K. Kunci ini dapat berupa nilai bilangan yang cukup besar. Range dari nilai kunci yang mungkin tersebut disebut Ruang Kunci (*keyspace*). Baik proses enkripsi maupun dekripsi menggunakan kunci tersebut sehingga fungsi diatas menjadi :

$$E_k(M) = C$$

$$D_k(C) = M$$

Fungsi ini memiliki sifat yang sama dengan yang sebelumnya yaitu :

$$D_k(E_k(M)) = M$$

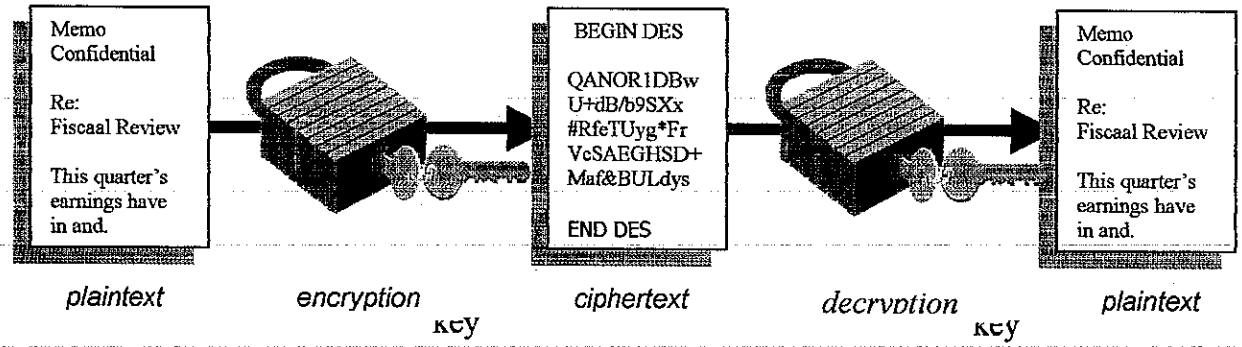
Beberapa algoritma menggunakan kunci enkripsi dan dekripsi yang berbeda, misalnya k1 yang berbeda dengan kunci dekripsi k2 sehingga memenuhi:

$$E_{k1}(M) = C$$

$$D_{k2}(C) = M$$

$$D_{k2}(E_{k1}(M)) = M$$

Proses enkripsi dan dekripsi memakai kunci dapat digambarkan sebagai berikut :




Gambar 2.7. Proses Enkripsi dan Dekripsi dengan Kunci

Teknik enkripsi terbagi atas dua tipe yaitu enkripsi kunci simetrik dan enkripsi kunci publik. Dan untuk enkripsi kunci simetrik terdapat dua kelas enkripsi yaitu sandi blok (*Block Cipher*) dan sandi berurut (*Stream Cipher*).

**Sandi blok** adalah skema enkripsi yang membagi pesan plantext menjadi string-string dengan ukuran panjang yang tetap (misalkan t disebut blok dari alpabet) serta mengenkripsi satu blok dalam satu waktu.

**Sandi berurut** adalah skema enkripsi terhadap pesan plantext ($m_1,m_2,m_3\ldots$) dan menghasilkan sandi $c_1,c_2,c_3,\ldots$ dengan aturan $c_i=Ee_i(m_i)$. Jadi prosesnya berjalan pada panjang blok sama dengan satu dan dengan ruang kunci (*keystream*) untuk transformasi enkripsinya.