Bab 4

Kriptografi Simetris Sederhana

4.1 Enkripsi Affine

Enkripsi yang digunakan Julius Caesar (*Caesar cipher*) menggunakan transformasi yang sederhana yaitu *shift transformation*. Pembahasan analisa statistik (lihat 2.3.2) menunjukkan bahwa *shift transformation* sangat rentan terhadap analisa frekuensi. Untuk mencoba mempersulit analisa frekuensi, enkripsi *affine* menggunakan *affine transformation*, dengan rumus:

$$C \equiv aP + b \pmod{n} \qquad \text{untuk enkripsi,} \tag{4.1}$$

dan

$$P \equiv a^{-1}C - a^{-1}b \pmod{n} \qquad \text{untuk dekripsi.} \tag{4.2}$$

Jadi kunci untuk enkripsi *affine* terdiri dari dua parameter: a dan b. Agar a mempunyai *inverse* a^{-1}, a harus mematuhi $\gcd(a, n) = 1$ (lihat teorema 11).

Sebagai contoh dari enkripsi *affine*, mari kita ganti *shift transformation* yang digunakan Julius Caesar dengan *affine transformation* menggunakan parameter $a = 7$, $b = 12$. Rumus untuk enkripsi menjadi:

$$C \equiv 7P + 12 \pmod{26}. \tag{4.3}$$

Nilai parameter $a = 7$ dapat digunakan karena $\gcd(7, 26) = 1$, jadi a mempunyai *inverse* a^{-1} dengan nilai $7^{-1} \equiv 15 \pmod{26}$. Oleh karena itu, rumus dekripsi menjadi:

$$P \equiv 15C - 15 \cdot 12 \equiv 15C + 2 \pmod{26}. \tag{4.4}$$

Dengan rumus 4.3 sebagai rumus enkripsi, huruf "a" yang mempunyai kode 0 mempunyai kode acak $7 \cdot 0 + 12 \equiv 12 \pmod{26}$ yang merupakan kode untuk huruf "m"; huruf "b" yang mempunyai kode 1 mempunyai kode acak $7 \cdot 1 + 12 \equiv 19 \pmod{26}$ yang merupakan kode untuk huruf "t"; dan seterusnya. Tabel 4.1 menunjukkan tabel lengkap untuk enkripsi *affine* dengan parameter $a = 7$, $b = 12$ dan $n = 26$.

Huruf Asli	Kode Asli	Kode Acak	Huruf Acak
a	0	12	m
b	1	19	t
c	2	0	a
d	3	7	h
e	4	14	o
f	5	21	v
g	6	2	c
h	7	9	j
i	8	16	q
j	9	23	x
k	10	4	e
l	11	11	l
m	12	18	s
n	13	25	z
o	14	6	g
p	15	13	n
q	16	20	u
r	17	1	b
s	18	8	i
t	19	15	p
u	20	22	w
v	21	3	d
w	22	10	k
x	23	17	r
y	24	24	y
z	25	5	f

Tabel 4.1: Tabel untuk contoh enkripsi affine

Tabel untuk enkripsi *affine transformation* menunjukkan bahwa sepasang huruf asli yang berurutan, sebagai contoh "c" dan "d", huruf acaknya ("a" dan "h") berjarak 7 $\pmod{26}$. Ini menunjukkan mengapa $\gcd(a, n) = 1$ diperlukan untuk enkripsi *affine*. Jika $\gcd(a, n) = 1$, seperti halnya dengan $a = 7$ dan $n = 26$, menambahkan $a \pmod{n}$ secara berulang akan mengembalikan

kita ke bilangan semula (sebut saja x) setelah n kali dan sedikitnya n kali penambahan:

$$x + an \equiv x \pmod{n} \text{ dan } x + an' \not\equiv x \pmod{n} \text{ untuk } 0 < n' < n.$$

Akibatnya semua bilangan bulat $0 \leq y < n$ akan "dikunjungi" setelah menambahkan a secara berulang sebanyak n kali. Jadi banyaknya kode acak yang digunakan sama dengan banyaknya kode asli, oleh sebab itu setiap kode asli mempunyai kode acak sendiri. Jadi setiap kode acak dapat didekripsi dengan unik.

Akan tetapi, jika $d = \gcd(a, n) \neq 1$, maka:

$$x + a(n/d) \equiv x \pmod{n} \text{ dan } x + an' \not\equiv x \pmod{n} \text{ untuk } 0 < n' < n/d,$$

jadi hanya ada n/d bilangan yang dikunjungi, tidak semua bilangan y dengan $0 \leq y < n$ dikunjungi, jadi tidak semua kode acak digunakan. Akibatnya setiap kode acak yang digunakan merupakan kode acak untuk lebih dari satu kode asli, tepatnya sebanyak d kode asli mempunyai kode acak yang sama. Jadi kode acak tidak dapat didekripsi dengan unik (penjelasannya secara matematis, ini disebabkan a tidak mempunyai *inverse*). Sebagai contoh, jika $a = 8$, $b = 12$ dan $n = 26$, maka setiap kode acak mempunyai $d = \gcd(8, 26) = 2$ kode asli, kode acak 12 ("m") mempunyai dua kode asli: 0 ("a") dan 13 ("n"). Jadi huruf "m" sebagai huruf acak tidak dapat didekripsi dengan unik. Untuk $a = 8$, $b = 12$ dan $n = 26$, hanya bilangan genap yang lebih kecil dari 26 digunakan sebagai kode acak. Bilangan ganjil tidak dikunjungi, jadi tidak digunakan sebagai kode acak. Akibatnya hanya ada 13 kode acak yang digunakan.

Kembali ke enkripsi *affine transformation* dengan parameter $a = 7$, $b = 12$ dan $n = 26$. Mari kita coba lakukan analisa frekuensi terhadap naskah acak yang kita enkripsi dengan *affine transformation* dan kita bandingkan dengan analisa sebelumnya terhadap *shift transformation* (lihat 2.3.2).

Naskah Asli	`Jangan rahasiakan pesan ini!`
Naskah Acak	`Xmzcmz bmjmiqmemz noimz qzq!`

Tabel 4.2: Enkripsi dengan *affine transformation*

Dengan *shift transformation*, untuk setiap percobaan kita mencari satu parameter kunci menggunakan satu persamaan, jadi untuk setiap percobaan kita pasangkan satu kode asli dengan satu kode acak yang sesuai berdasarkan statistik dari pengamatan empiris. Strategi pencarian adalah menggunakan pasangan dimana karakter aslinya mempunyai statistik penggunaan terbesar.

Karena *affine transformation* menggunakan dua parameter, setiap percobaan kita harus mencari kedua parameter sedikitnya menggunakan dua persamaan dengan dua pasangan. Strategi pencarian adalah dengan mencoba dua

pasangan dimana dua karakter aslinya merupakan dua karakter dengan statistik penggunaan terbesar.

Untuk contoh diatas, dengan perumpamaan urutan tiga besar untuk penggunaan huruf dalam bahasa Indonesia adalah "A", "N" dan kemudian "I", kita coba dahulu pasangkan "A" dengan "M" dan "N" dengan "Z" untuk mendapatkan dua persamaan:

$$12 \equiv a \cdot 0 + b \pmod{26}$$

dan

$$25 \equiv a \cdot 13 + b \pmod{26}.$$

Dengan mengurangkan persamaan pertama dari persamaan kedua, kita hilangkan b mendapatkan:

$$13 \equiv a \cdot 13 \pmod{26}.$$

Akan tetapi karena 13 tidak mempunyai *inverse* (karena 13 membagi 26), kita harus membagi persamaan dengan 13 menjadi:

$$1 \equiv a \cdot 1 \pmod{2},$$

yang berarti a adalah bilangan ganjil. Kita dapat mencoba setiap bilangan ganjil $0 < a < 26$ untuk mencari hasil yang cocok, akan tetapi mungkin kita tidak sabar untuk melakukan hal itu.

Sebagai alternatif, kita dapat lanjutkan analisa frekuensi dengan mencoba huruf dengan statistik penggunaan terbesar nomor tiga. Kita pasangkan "I" dengan "Q", menggantikan persamaan pertama. Jadi dua persamaan yang kita gunakan adalah:

$$16 \equiv a \cdot 8 + b \pmod{26}$$

dan

$$25 \equiv a \cdot 13 + b \pmod{26}.$$

Kita kurangkan persamaan pertama dari persamaan kedua mendapatkan:

$$9 \equiv a \cdot 5 \pmod{26}.$$

Jadi $a \equiv 9 \cdot 5^{-1} \equiv 9 \cdot 21 \equiv 7 \pmod{26}$, dan $b \equiv 16 - (7 \cdot 8) \equiv 12 \pmod{26}$. Dengan percobaan ini kita dapatkan parameter $a = 7$ dan $b = 12$.

Contoh diatas menunjukkan bahwa, dengan *affine transformation*, dua pasangan tidak selalu menghasilkan nilai parameter secara langsung. Kadang kita harus mencoba pasangan lain. Analisa frekuensi terhadap enkripsi *affine* memang lebih sulit dibandingkan analisa frekuensi terhadap enkripsi yang menggunakan *shift transformation*, namun analisa frekuensi terhadap enkripsi *affine* masih tergolong mudah untuk dilakukan.

4.2 Transformasi Digraph

Analisa frekuensi menjadi lebih dipersulit lagi jika enkripsi terhadap karakter tidak dilakukan satu persatu melainkan sekaligus terhadap beberapa karakter. Jika transformasi dilakukan terhadap dua karakter sekaligus, maka transformasi disebut transformasi *digraph* (*digraph transformation*). Sebelum enkripsi, jika jumlah karakter ganjil, naskah dapat ditambah dengan *padding character* agar jumlah karakter genap. Setelah dekripsi, *padding character* jika ada dapat dibuang.

Setiap *digraph* terdiri dari dua karakter dan diberi kode bilangan. Cara termudah untuk memberi kode bilangan adalah dengan rumus:

$$xn + y$$

dimana x adalah kode bilangan untuk karakter pertama, y adalah kode bilangan untuk karakter kedua, dan n adalah besarnya perbendaharaan karakter untuk enkripsi Jadi kode untuk digraph mempunyai nilai antara 0 dan $n^2 - 1$ inklusif.

Affine transformation dapat dilakukan terhadap digraph menggunakan rumus enkripsi:

$$C \equiv aP + b \pmod{n^2}. \tag{4.5}$$

Tentunya kita memerlukan $\gcd(a, n^2) = 1$ agar a mempunyai *inverse*. Akan tetapi kita cukup melakukan test $\gcd(a, n) = 1$ karena jika $\gcd(a, n) = 1$ maka kita tahu $\gcd(a, n^2) = 1$, sedangkan jika $\gcd(a, n) \neq 1$ maka kita tahu $\gcd(a, n^2) \neq 1$, meskipun $\gcd(a, n^2) \neq \gcd(a, n)$.

Rumus untuk dekripsi adalah:

$$P \equiv a^{-1}C - a^{-1}b \pmod{n^2}. \tag{4.6}$$

Sebagai contoh, kita gunakan perbendaharaan karakter terdiri dari semua huruf besar plus spasi, dengan kode bilangan 26 untuk spasi dan 0 sampai dengan 25 untuk huruf "A" sampai dengan "Z". Jadi $n = 27$ dan $n^2 = 729$. Menggunakan parameter $a = 614$ dan $b = 47$, rumus enkripsi menjadi:

$$C \equiv 614P + 47 \pmod{729}. \tag{4.7}$$

Parameter $a = 614$ dapat digunakan karena $\gcd(614, 729) = 1$, jadi a mempunyai *inverse* $a^{-1} \equiv 374 \pmod{729}$.

Rumus untuk dekripsi menjadi:

$$P \equiv 374C - 374 \cdot 47 \pmod{729} \equiv 374C + 647 \pmod{729}. \tag{4.8}$$

Mari kita coba enkripsi *digraph* dengan *affine transformation* diatas terhadap naskah: `"JANGAN RAHASIAKAN PESAN INI "`. Naskah harus ditambah *padding* ahir berupa spasi supaya jumlah karakter genap.

Digraph Asli	Kode Asli	Kode Acak	Digraph Acak
JA	243	533	TU
NG	357	545	UF
AN	13	10	AK
_R	719	468	RJ
AH	7	700	ZZ
AS	18	164	GC
IA	216	722	_U
KA	270	344	MU
N_	377	432	QA
PE	409	397	OT
SA	486	290	KU
N_	377	432	QA
IN	229	685	ZK
I_	242	648	YA

Tabel 4.3: Tabel untuk contoh enkripsi digraph

Naskah acak menjadi: `"TUUFAKRJZZGC UMUQAOTKUQAZKYA"`. Kita dapat melihat bahwa huruf acak tidak selalu sama untuk huruf asli yang sama. Contohnya, huruf asli "A" mempunyai huruf acak "A", "G", "U" dan "Z", meskipun diposisi kedua dalam *digraph*, "A" selalu ditukar dengan "U".

Mari kita coba lakukan analisa frekuensi terhadap naskah dengan menghitung frekuensi *digraph*. Namun naskah acak kurang panjang untuk mendapatkan statistik penggunaan *digraph* yang cukup baik. Hanya *digraph* "QA" yang digunakan lebih dari satu kali dalam naskah acak. Jadi kita bisa menyimpulkan bahwa analisa frekuensi terhadap *digraph* membutuhkan *sample* naskah acak yang lebih panjang daripada yang dibutuhkan jika transformasi dilakukan terhadap setiap huruf.

Mari kita umpamakan bahwa disamping naskah acak diatas, kita mendapatkan naskah acak lain yang cukup panjang untuk mendapatkan statistik penggunaan yang cukup baik dengan urutan empat terbesar penggunaan *digraph* sebagai berikut: "AK", "QA", "YA" dan "_A", dimana karakter "_" dalam *digraph* merepresentasikan spasi. Mari umpamakan juga bahwa dalam bahasa Indonesia, empat terbesar penggunaan *digraph* secara berurut adalah: "AN", "N_", "I_" dan "A_". Kita pasangkan *digraph* acak dengan *digraph* asli menurut urutan penggunaan.

Keempat pasangan menghasilkan empat persamaan yang dapat digunakan untuk mencari parameter a dan b:

$$10 \equiv 13a + b \pmod{729}$$

Digraph Acak (C)	Digraph Asli (P)
AK (10)	AN (13)
QA (432)	N_(377)
YA (648)	I_ (242)
_A (702)	A_ (26)

Tabel 4.4: Pasangan digraph menurut urutan frekuensi penggunaan

$$\begin{aligned} 432 &\equiv 377a + b \pmod{729} \\ 648 &\equiv 242a + b \pmod{729} \\ 702 &\equiv 26a + b \pmod{729} \end{aligned}$$

Dengan mengurangkan persamaan pertama dari persamaan kedua, kita hilangkan parameter b mendapatkan:

$$422 \equiv 364a \pmod{729}$$

Karena $\gcd(364, 729) = 1$, kita bisa mengkalkulasi $364^{-1} \pmod{729}$ menggunakan *extended Euclidean algorithm* menghasilkan $727 \pmod{729}$. Jadi:

$$a \equiv 422 \cdot 364^{-1} \equiv 422 \cdot 727 \equiv 614 \pmod{729}.$$

Dengan memasukkan nilai parameter a kedalam persamaan pertama, kita dapatkan:

$$b \equiv 10 - 13 \cdot 614 \equiv 47 \pmod{729}.$$

Jadi analisa frekuensi menghasilkan parameter $a = 614$ dan $b = 47$.

Kita beruntung mendapatkan nilai untuk parameter a dan b hanya dengan satu percobaan menggunakan persamaan pertama dan kedua. Seperti halnya analisa frekuensi *affine transformation* per karakter, analisa frekuensi *affine transformation* per *digraph* tidak selalu mendapatkan nilai a dan b secara langsung dari dua persamaan. Bahkan, pada umumnya, analisa frekuensi per *digraph* jauh lebih sulit dibandingkan analisa frekuensi per karakter. Namun enkripsi dengan *affine transformation* terhadap *digraph* masih dianggap rentan terhadap analisa frekuensi.

Pengamatan terhadap *affine transformation* terhadap *digraph* juga menunjukkan bahwa huruf kedua dalam *digraph* acak ditentukan hanya oleh huruf kedua dalam *digraph* asli ("U" sebagai huruf kedua dalam *digraph* acak selalu berasal dari "A" sebagai huruf kedua dalam *digraph* asli, "K" berasal dari "N", dan seterusnya). Ini disebabkan huruf kedua *digraph* acak hanya tergantung nilai *digraph* asli modulo n, jadi hanya tergantung huruf kedua *digraph* asli. Akibatnya analisa frekuensi terhadap huruf kedua semua *digraph* acak dapat menghasilkan parameter a dan b modulo n. Meskipun belum menentukan a

dan b modulo n^2, informasi ini cukup berharga dalam memecahkan enkripsi, karena ruang pencarian diperkecil secara signifikan. Kita tidak perlu mengindahkan berbagai nilai untuk parameter a dan b yang tidak sesuai dengan nilai a dan b modulo n. Sebagai contoh, jika $a \equiv 5 \pmod{27}$, kita tidak perlu mengindahkan $a \equiv 33 \pmod{729}$ karena $33 \equiv 6 \not\equiv 5 \pmod{27}$.

4.3 Matrik Enkripsi

Kita sudah melihat bagaimana *digraph* dapat direpresentasikan menggunakan bilangan bulat modulo n^2. Sudut pandang lain adalah melihat *digraph* sebagai vektor dua dimensi. Sebagai contoh, *digraph* "DI" dapat dipandang sebagai vektor

$$\begin{bmatrix} 3 \\ 8 \end{bmatrix}.$$

Enkripsi terhadap *digraph* dapat dipandang sebagai transformasi vektor yang bersifat *one-to-one*. Sifat *one-to-one* diperlukan agar kode acak dapat didekripsi secara unik. Karena *domain* dan *codomain* (target) sama dan bersifat *finite* (yaitu ruang vektor $(\mathbf{Z}/n\mathbf{Z})^2$ dimana n adalah besarnya perbendaharaan karakter), transformasi *one-to-one* vektor adalah *bijective map*. Jadi transformasi dekripsi, yang merupakan *inverse* dari transformasi enkripsi, mempunyai *domain* yang meliputi seluruh ruang vektor $(\mathbf{Z}/n\mathbf{Z})^2$.

Sudut pandang enkripsi sebagai transformasi vektor berlaku juga untuk blok yang berisi lebih dari dua karakter, dengan dimensi vektor sama dengan banyaknya karakter dalam blok. Dimasa silam, teknik enkripsi yang sangat populer selama ratusan tahun adalah *Vigenère cipher*. Dengan *Vigenère cipher*, setiap blok yang terdiri dari k karakter direpresentasikan menggunakan vektor dengan ruang $(\mathbf{Z}/n\mathbf{Z})^k$, dan enkripsi berupa *shift transformation* sebagai berikut:

$$C = P + b, \tag{4.9}$$

dimana b yang berupa vektor dengan dimensi k adalah kunci enkripsi. Transformasi dilakukan dengan menambahkan vektor b ke vektor asli P sebagai berikut:

$$P = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_k \end{bmatrix}, b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}, C = P + b = \begin{bmatrix} p_1 + b_1 \\ p_2 + b_2 \\ \vdots \\ p_k + b_k \end{bmatrix}.$$

Pertambahan skalar $p_i + b_i$ untuk $1 \leq i \leq k$ menggunakan aritmatika modulo n (aritmatika $\mathbf{Z}/n\mathbf{Z}$). Dekripsi dilakukan sebagai berikut:

$$P = C + b', \tag{4.10}$$

dengan

$$b' = -b = \begin{bmatrix} -b_1 \\ -b_2 \\ \vdots \\ -b_k \end{bmatrix}.$$

Setiap $-b_i$ untuk $1 \leq i \leq k$ menggunakan *inverse* pertambahan dalam aritmatika $\mathbf{Z}/n\mathbf{Z}$.

Vigenère cipher sangat rentan terhadap analisa frekuensi. Analisa frekuensi *shift transformation* terhadap karakter seperti yang dijelaskan di 2.3.2 dapat dilakukan terhadap setiap posisi dalam vektor menghasilkan komponen vektor b. Jadi analisa posisi 1 menghasilkan b_1, analisa posisi 2 menghasilkan b_2, dan seterusnya sampai semua komponen vektor kunci b ditemukan.

Selain *shift transformation* dimana *shift vector* ditambahkan ke vektor asal, transformasi terhadap vektor juga dapat dilakukan dengan mengalikan matrik ke vektor asal. Untuk keperluan enkripsi, matrik pengali yang disebut matrik enkripsi (*enciphering matrix*) harus mempunyai *inverse* matrik. Perkalian matrik dengan vektor sesuai dengan aljabar linear. Sebagai contoh, mengalikan matrik berdimensi 2×2 dengan vektor dimensi 2 kita dapatkan:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix},$$

dimana $ax + by$ dan $cx + dy$ dikalkulasi menggunakan aritmatika $\mathbf{Z}/n\mathbf{Z}$.

Dalam aljabar linear dengan bilangan nyata (*real*), sebuah matrik mempunyai *inverse* jika dan hanya jika ($\Longleftrightarrow$) determinan (*determinant*) matrik D mempunyai *inverse* D^{-1}, yang dalam aritmatika bilangan nyata berarti $D \neq 0$. Demikian juga untuk matrik enkripsi A, A mempunyai *inverse* jika dan hanya jika ($\Longleftrightarrow$) deteminan A, yang dikalkulasi menggunakan aritmatika $\mathbf{Z}/n\mathbf{Z}$, D mempunyai inverse D^{-1}. Tetapi berbeda dengan aritmatika bilangan nyata, ini berarti $\gcd(D, n) = 1$ (lihat teorema 11).

Mari kita tinjau kembali konsep-konsep aljabar linear yang dapat digunakan untuk mengkalkulkasi *inverse* matrik. Konsep utama adalah konsep determinan, yang dapat didefinisikan secara rekursif (*recursive definition* atau *inductive definition*). Untuk matrik dengan dimensi 1×1, definisi determinan sangat sederhana:

$$A = [a_{1,1}],\ \det(A) = a_{1,1}.$$

Untuk matrik dengan dimensi $n \times n$ dimana $n > 1$, kita dapat memilih baris i dengan $1 \leq i \leq n$, dan deteminan dapat dikalkulasi sebagai berikut:

$$\det(A) = a_{i,1} \cdot \text{cofactor}_{i,1}(A) + a_{i,2} \cdot \text{cofactor}_{i,2}(A) + \ldots + a_{i,n} \cdot \text{cofactor}_{i,n}(A),$$

dimana $\text{cofactor}_{i,j}(A)$ adalah determinan matrik yang didapat dengan menghapus baris i dan kolom j dari matrik A, dikalikan dengan $(-1)^{i+j}$ (jika $i + j$

ganjil, determinan dikalikan dengan -1, jika $i + j$ genap, determinan tidak dikalikan). Sebagai contoh, untuk matrik dengan dimensi 2×2:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix},$$

terdapat empat *cofactor*, masing-masing:

$$\begin{aligned} \text{cofactor}_{1,1}(A) &= a_{2,2}, \\ \text{cofactor}_{1,2}(A) &= -a_{2,1}, \\ \text{cofactor}_{2,1}(A) &= -a_{1,2}, \\ \text{cofactor}_{2,2}(A) &= a_{1,1}, \end{aligned}$$

dan determinan dapat dikalkulasi dengan:

$$\begin{aligned} \det(A) &= a_{1,1} \cdot \text{cofactor}_{1,1}(A) + a_{1,2} \cdot \text{cofactor}_{1,2}(A) \\ &= a_{1,1} \cdot a_{2,2} + a_{1,2} \cdot (-a_{2,1}) \end{aligned}$$

atau

$$\begin{aligned} \det(A) &= a_{2,1} \cdot \text{cofactor}_{2,1}(A) + a_{2,2} \cdot \text{cofactor}_{2,2}(A) \\ &= a_{2,1} \cdot (-a_{1,2}) + a_{2,2} \cdot a_{1,1}. \end{aligned}$$

Selain determinan dan *cofactor*, kita memerlukan konsep *transpose* matrik:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,n} \\ a_{2,1} & a_{2,2} & \dots & a_{2,n} \\ \vdots & & & \\ a_{n,1} & a_{n,2} & \dots & a_{n,n} \end{bmatrix}, A^T = \begin{bmatrix} a_{1,1} & a_{2,1} & \dots & a_{n,1} \\ a_{1,2} & a_{2,2} & \dots & a_{n,2} \\ \vdots & & & \\ a_{1,n} & a_{2,n} & \dots & a_{n,n} \end{bmatrix},$$

dan *adjoint* matrik:

$$\text{adj}(A) = \begin{bmatrix} \text{cofactor}_{1,1}(A) & \text{cofactor}_{1,2}(A) & \dots & \text{cofactor}_{1,n}(A) \\ \text{cofactor}_{2,1}(A) & \text{cofactor}_{2,2}(A) & \dots & \text{cofactor}_{2,n}(A) \\ \vdots & & & \\ \text{cofactor}_{n,1}(A) & \text{cofactor}_{n,2}(A) & \dots & \text{cofactor}_{n,n}(A) \end{bmatrix}^T.$$

Jadi *transpose* sebuah matrik didapatkan dengan mempertukarkan baris dengan kolom: baris 1 menjadi kolom 1, kolom 1 menjadi baris 1, baris 2 menjadi kolom 2, kolom 2 menjadi baris 2, dan seterusnya. Sedangkan *adjoint* dari sebuah matrik adalah *transpose* dari matrik *cofactor*.

Kita dapat mengkalkulasi *inverse* matrik sebagai berikut:

$$A^{-1} = \det(A)^{-1} \cdot \text{adj}(A). \tag{4.11}$$

Untuk matrik dengan dimensi 2×2, rumus untuk *inverse* menjadi:

$$A^{-1} = (a_{1,1} \cdot a_{2,2} + a_{1,2} \cdot (-a_{2,1}))^{-1} \cdot \begin{bmatrix} a_{2,2} & -a_{1,2} \\ -a_{2,1} & a_{1,1} \end{bmatrix}.$$

Kita coba kalkulasi *inverse* matrik enkripsi:

$$A = \begin{bmatrix} 2 & 3 \\ 7 & 8 \end{bmatrix},$$

dimana $n = 26$, jadi aritmatika yang digunakan adalah aritmatika $\mathbf{Z}/26\mathbf{Z}$. Determinan $D = \det(A) = (2 \cdot 8 + (-3) \cdot 7) \bmod 26 = -5 \bmod 26 = 21$. Karena $\gcd(21, 26) = 1$, D mempunyai *inverse*, yang dikalkulasi menggunakan *extended Euclidean algorithm* menghasilkan 5. Jadi,

$$A^{-1} = 5 \cdot \begin{bmatrix} 8 & -3 \\ -7 & 2 \end{bmatrix} = \begin{bmatrix} 40 \bmod 26 & -15 \bmod 26 \\ -35 \bmod 26 & 10 \bmod 26 \end{bmatrix} = \begin{bmatrix} 14 & 11 \\ 17 & 10 \end{bmatrix}.$$

Kita dapat periksa:

$$\begin{bmatrix} 14 & 11 \\ 17 & 10 \end{bmatrix} \begin{bmatrix} 2 & 3 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 105 \bmod 26 & 130 \bmod 26 \\ 104 \bmod 26 & 131 \bmod 26 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Mari kita coba enkripsi "DIMANASAJA" menggunakan matrik enkripsi diatas. Naskah terdiri dari 5 *digraph* yang direpresentasikan menggunakan 5 vektor:

$$\begin{bmatrix} 3 \\ 8 \end{bmatrix} \begin{bmatrix} 12 \\ 0 \end{bmatrix} \begin{bmatrix} 13 \\ 0 \end{bmatrix} \begin{bmatrix} 18 \\ 0 \end{bmatrix} \begin{bmatrix} 9 \\ 0 \end{bmatrix}.$$

Kelima vektor dapat digabung menjadi satu matrik:

$$\begin{bmatrix} 3 & 12 & 13 & 18 & 9 \\ 8 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Kita kalikan matrik enkripsi dengan matrik naskah asli untuk mendapatkan matrik naskah acak:

$$\begin{aligned} C &= AP \\ &= \begin{bmatrix} 2 & 3 \\ 7 & 8 \end{bmatrix} \begin{bmatrix} 3 & 12 & 13 & 18 & 9 \\ 8 & 0 & 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 30 \bmod 26 & 24 \bmod 26 & 26 \bmod 26 & 36 \bmod 26 & 18 \bmod 26 \\ 85 \bmod 26 & 84 \bmod 26 & 91 \bmod 26 & 126 \bmod 26 & 63 \bmod 26 \end{bmatrix} \\ &= \begin{bmatrix} 4 & 24 & 0 & 10 & 18 \\ 7 & 6 & 13 & 22 & 11 \end{bmatrix}. \end{aligned}$$

Jadi kita dapatkan naskah acak: "EHYGANKWSL".

Kita periksa apakah *inverse* matrik enkripsi dapat digunakan untuk dekripsi matrik naskah acak:

$$\begin{aligned} P &= A^{-1}C \\ &= \begin{bmatrix} 14 & 11 \\ 17 & 10 \end{bmatrix} \begin{bmatrix} 4 & 24 & 0 & 10 & 18 \\ 7 & 6 & 13 & 22 & 11 \end{bmatrix} \\ &= \begin{bmatrix} 133 & 402 & 143 & 382 & 373 \\ 138 & 468 & 130 & 390 & 416 \end{bmatrix} \bmod 26 \\ &= \begin{bmatrix} 3 & 12 & 13 & 18 & 9 \\ 8 & 0 & 0 & 0 & 0 \end{bmatrix}. \end{aligned}$$

Jadi mengalikan *inverse* matrik enkripsi dengan matrik naskah acak menghasilkan matrik naskah asli.

Mari kita selidiki bagaimana transformasi dengan matrik enkripsi dapat dianalisa. Kita andaikan naskah acak "YPKWPRHODEZKCJ" dienkripsi dengan matrik enkripsi terhadap *digraph* dan kita mengetahui bahwa naskah asli diahiri dengan nama pengirim yaitu "NETTY". Jadi kita mengetahui bahwa dua *digraph* terahir dalam naskah asli adalah "ET" dan "TY". Dua *digraph* terahir dalam naskah acak adalah "ZK" dan "CJ", jadi "ZK" dan "CJ" adalah enkripsi dari "ET" dan "TY". Kita representasikan dua *digraph* asli dengan matrik:

$$P = \begin{bmatrix} 4 & 19 \\ 19 & 24 \end{bmatrix}$$

dan dua *digraph* acak dengan matrik:

$$C = \begin{bmatrix} 25 & 2 \\ 10 & 9 \end{bmatrix}.$$

Menggunakan rumus untuk dekripsi $P = A^{-1}C$, kita dapatkan:

$$\begin{bmatrix} 4 & 19 \\ 19 & 24 \end{bmatrix} = A^{-1} \begin{bmatrix} 25 & 2 \\ 10 & 9 \end{bmatrix}.$$

Jadi kita bisa mendapatkan A^{-1} sebagai berikut:

$$A^{-1} = \begin{bmatrix} 4 & 19 \\ 19 & 24 \end{bmatrix} \begin{bmatrix} 25 & 2 \\ 10 & 9 \end{bmatrix}^{-1} = \begin{bmatrix} 4 & 19 \\ 19 & 24 \end{bmatrix} \begin{bmatrix} 23 & 18 \\ 12 & 9 \end{bmatrix} = \begin{bmatrix} 8 & 9 \\ 23 & 12 \end{bmatrix}.$$

Kita periksa apakah A^{-1} bisa mendapatkan naskah asli yang "masuk akal:"

$$\begin{aligned} P &= \begin{bmatrix} 8 & 9 \\ 23 & 12 \end{bmatrix} \begin{bmatrix} 24 & 10 & 15 & 7 & 3 & 25 & 2 \\ 15 & 22 & 17 & 14 & 4 & 10 & 9 \end{bmatrix} \\ &= \begin{bmatrix} 15 & 18 & 13 & 0 & 8 & 4 & 19 \\ 4 & 0 & 3 & 17 & 13 & 19 & 24 \end{bmatrix}. \end{aligned}$$

Naskah asli yang kita dapatkan yaitu "PESANDARINETTY" cukup masuk akal. Jadi kita bisa cukup yakin bahwa A^{-1} yang kita dapatkan adalah benar.

Kita baru saja berhasil melakukan *known-plaintext attack* terhadap transformasi dengan matrik enkripsi. Analisa frekuensi juga dapat dilakukan terhadap transformasi matrik enkripsi, dengan matrik acak dan matrik asli dipasangkan menurut analisa frekuensi, bukan berdasarkan pengetahuan pasti mengenai naskah asli. Rumus untuk mendapatkan A^{-1} hanya dapat digunakan apabila matrik acak mempunyai *inverse*, jadi ada kemungkinan lebih dari satu pasangan harus dicoba (ini berlaku juga untuk *known-plaintext attack*). Matrik acak dan asli harus berdimensi $n \times n$ dimana n adalah jumlah karakter dalam blok.

Transformasi dengan matrik enkripsi tergolong *linear transformation*, yang berarti jika C_1 adalah naskah acak untuk P_1 dan C_2 adalah naskah acak untuk P_2, maka $C_1 + C_2$ adalah naskah acak untuk $P_1 + P_2$. Transformasi matrik enkripsi dapat digabung dengan *shift transformation* menghasilkan *affine transformation*. Matrik enkripsi A dikalikan dengan vektor asli, kemudian hasilnya ditambahkan dengan *shift vector* B:

$$C = AP + B. \tag{4.12}$$

Sebagai contoh, untuk *digraph* rumus menjadi:

$$\begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} a_{1,1}p_1 + a_{1,2}p_2 + b_1 \\ a_{2,1}p_1 + a_{2,2}p_2 + b_2 \end{bmatrix}.$$

Rumus untuk dekripsi adalah:

$$P = A^{-1}C - A^{-1}B. \tag{4.13}$$

Analisa *affine transformation* matrik enkripsi terhadap *digraph* membutuhkan minimum 3 pasang *digraph* dengan 3 persamaan:

$$\begin{aligned} P_1 &= A^{-1}C_1 - A^{-1}B \\ P_2 &= A^{-1}C_2 - A^{-1}B \\ P_3 &= A^{-1}C_3 - A^{-1}B \end{aligned}$$

Persamaan ketiga dikurangkan dari dua persamaan pertama untuk menghilangkan *shift vector* B. Teknik untuk *linear transformation* yang telah dijelaskan dapat digunakan untuk mencari A^{-1} menggunakan dua persamaan yang sudah tidak mengandung B (kedua persamaan dapat digabung dengan menggabungkan vektor menjadi matrik). Setelah A^{-1} didapat, maka B dapat dikalkulasi.

Secara umum, untuk *affine transformation* menggunakan matrik enkripsi dengan besar blok n karakter, dibutuhkan $n+1$ persamaan untuk mengkalkulasi A^{-1} dan B. (Lagi n vektor dapat digabung menjadi matrik.)

4.4 Ringkasan

Enkripsi *Caesar cipher* sangat mudah untuk dipecahkan, baik menggunakan analisa statistik, maupun dengan cara *known plaintext attack*. Di bab ini telah diperkenalkan teknik-teknik enkripsi yang mempersulit analisa statistik dan cara pemecahan lainnya. Teknik *affine transformation* membuat enkripsi semakin tidak linear jadi lebih sukar analisanya. Enkripsi beberapa karakter sekaligus juga merupakan kemajuan dalam kriptografi karena analisanya lebih sulit dibandingkan jika setiap karakter dienkripsi sendiri. Konsep *digraph* digunakan jika dua karakter sekaligus dienkripsi. Transformasi beberapa karakter sekaligus dapat direpresentasikan menggunakan matrik enkripsi. Meskipun teknik-teknik yang diperkenalkan dalam bab ini membuat enkripsi lebih kuat dibandingkan dengan *Caesar cipher*, hasilnya masih rentan terhadap berbagai analisa statistik. Dalam bab-bab selanjutnya, kita akan bahas teknik-teknik yang dapat membuat enkripsi lebih kuat lagi.