Pengantar Kriptografi[1]

Sugi Guritman
Fakultas Matematika dan IPA
Institut Pertanian Bogor
BOGOR

2003

[1]Diberikan sebagai *handout* untuk pengantar mata kuliah *Pengantar Kriptografi / Data Security* ILKOM IPB.

Contents

Chapter 1

Pendahuluan

Di bab pendahuluan ini akan tercakup empat bahasan, yaitu:

- Ringkasan Sejarah Kriptografi,
- Keamanan Informasi dan Kriptografi,
- Pendefinisian Beberapa Fungsi

1.1 Ringkasan Sejarah Kriptografi

Di dalam buku yang ditulis oleh Kahn berjudul *The CodeBreaker*, terlihat bahwa kriptografi mempunyai sejarah yang panjang. Buku ini melacak bahwa kriptografi sudah digunakan oleh bangsa Mesir Kuno sekitar 4000 tahun sampai abad 20 dimana kriptografi berperanan penting di dalam perang dunia pertama dan kedua. Juga diungkap bahwa latar belakang sejarah yang panjang itu sangat menentukan perkembangan ilmu kriptografi itu sendiri baik dari segi teoretik maupun aplikasinya.

Pada awalnya kriptografi sangat dominan digunakan dalam bidang-bidang yang berhubungan dengan militer, layanan diplomatik, dan pemerintahan secara umum. Dalam hal ini kriptografi digunakan sebagai suatu alat untuk melindungi strategi dan rahasia negara. Perkembangan sistem komunikasi dan komputer pada tahun 1960 an membawa kriptografi memasuki sektor swasta sebagai alat untuk melindungi informasi dalam bentuk dijitel dan untuk memberikan layanan keamanan.

Hasil kerja Feistel di IBM pada awal tahun 1970 an dan puncaknya pada tahun 1977, DES (*Data Ecryption Standart*) merupakan karya kriptografik yang paling terkenal di dalam sejarah. Karya ini menjadi alat kea-

manan komersial elektronik di banyak institusi keuangan di seluruh dunia hingga pertengahan tahun 1990-an. DES secara definitif terbukti tak-aman sejak Juli 1998. Walaupun demikian DES telah melandasi prinsip-prinsip sandi simetrik modern yang dewasa ini muncul produk-produk penggantinya seperti: AES (*Advanced Ecryption Standart*), Blowfish, 3DES, RC5, dan lain sebagainya.

Perkembangan yang cukup signifikan selanjutnya adalah pada tahun 1976 ketika Diffie dan Hellman mempublikasikan suatu artikel dengan judul *New Directions in Cryptography.* Artikel ini memperkenalkan konsep revolusioner tentang kriptografi kunci-publik (*public-key cryptography*) dan juga memberikan suatu metode baru untuk perubahan kunci dimana keamanan didasarkan pada pemecahan problem logaritme diskret. Walaupun penulis pada saat itu mengungkapkan hanya segi teoretiknya tanpa bentuk praktisnya, namun karya ini telah memberikan cakrawala baru bagi para ilmuwan kriptografik. Ini terbukti pada tahun 1978, Rivest, Shamir, dan Adleman menemukan bentuk praktis yang pertama untuk skema enkripsi dan penandaan kunci publik yang sekarang dikenal dengan *skema RSA*. Skema ini didasarkan pada problem matematika yang sulit lainnya, yaitu pemecahan masalah faktorisasi intejer besar. Bentuk praktis skema kunci-publik lainya ditemukan oleh ElGamal pada tahun 1985. Sebagaimana karya Diffie dan Hellman, skema ini juga didasarkan pada pemecahan problem logaritme diskret.

Salah satu sumbangan yang paling signifikan yang diberikan olen kriptografi kunci-publik adalah penandaan dijitel (*digital signature*). Pada tahun1991 standar internasional pertama untuk penandaan dijitel diadopsi dari ISO/IEC 9796. Standar internasional penandaan dijitel ini didasarkan pada skema kunci-publik RSA. Pada tahun 1994 pemerintah Amerika Serikat mengadopsi standar penandaan dijitel yang mekanismenya didasarkan pada skema kunci-publik ElGamal.

1.2 Keamanan Informasi dan Kriptografi

Konsep informasi dalam kriptografi harus dipahami sebagai *kuantitas* bukan kualitas. Kriptografi sendiri dipahami sebagai hal-hal yang terkait dengan keamanan informasi. Keamanan informasi menjelma di dalam banyak cara sesuai dengan situasi dan kebutuhan. Tanpa memandang siapa yang terlibat di dalam suatu transakasi informasi, mereka haruslah mempunyai tujuan yang sama tentang keamanan informasi. Beberapa tujuan keamanan informasi diantaranya adalah sebagai berikut:

1. *Kerahasiaan* (privacy/confidentiality), maksudnya menjaga rahasia informasi dari siapapun kecuali yang berwenang untuk mengetahuinya.

2. *Integritas data* (data integrity), maksudnya menjamin bahwa informasi tidak diganti oleh siapapun yang tidak berwenang.

3. *Identifikasi atau autentikasi entitas* (entity authentication or identification), maksudnya pembuktian yang kuat tentang identitas suatu entitas. *Entitas* adalah unsur-unsur yang menjadi komponen dalam suatu transaksi informasi, ini bisa berupa orang, terminal komputer, kartu kredit, dll.

4. *Autentikasi pesan* (message authentication), maksudnya pembuktian yang kuat bahwa pesan benar-benar berasal dari sumber informasi; istilah lainnya adalah *autentikasi asal data*.

5. *Penandaan* (signature), maksudnya suatu alat yang digunakan untuk memberikan ciri tertentu pada informasi yang ditujukan ke suatu entitas.

6. *Kewenangan* (authorization), maksudnya kesepakatan resmi yang diberikan kepada entitas lain untuk melakukan sesuatu atau menjadi sesuatu.

7. *Validasi* (validation), maksudnya suatu alat yang digunakan memberi tanda masa berlakunya kewenangan di dalam pemakaian atau manipulasi informasi.

8. *Sertifikasi* (certification), maksudnya penguasaan informasi oleh suatu entitas yang dipercaya.

9. *Non-repudiasi* (non-repudiation), maksudnya pencegahan dari pelanggaran kesepakatan-kesepakatan yang telah dibuat sebelumnya.

10. *Tanda terima* (receipt), maksudnya suatu bukti bahwa informasi telah diterima.

11. *Konfirmasi* (confirmation), maksudnya suatu bukti bahwa informasi telah diberikan.

Tujuan keamanan informasi dapat dicapai tidak semata-mata bergantung kepada algoritme matematik dan mekanismenya, tetapi juga pada faktor-faktor lainya seperti prosedur penyampaian yang digunakan atau perlindungan hukum. Misalnya, surat rahasia dikemas dalam amplop bersegel diki-

rimkan melalui kantor pos resmi. Keamanan surat itu secara fisik berada di dalam amplop bersegel, dikirimkan melalui kantor pos resmi akan mendapatkan perlindungan hukum bahwa barang siapa yang membuka amplop bersegel akan mendapatkan sanksi hukum. Kadangkala keamanan informasi juga dapat dicapai dari unsur fisik yang melekat pada dokumen informasi. Misalnya, pencetakan uang kertas menggunakan tinta khusus akan mencegah dari pemalsuan.

Secara konsepsual, cara informasi dicatat atau disimpan belum berubah secara drastik dari waktu ke waktu. Hal ini dapat dilihat bahwa media penyimpanan biasanya berupa kertas atau media magnetik dan pengirimannya melalui sistem telekomunikasi baik yang dengan kabel atau tanpa kabel. Yang berubah secara drastik justru kemampuan untuk menyalin, mengubah dan memanipulasi informasi. Dalam hitungan menit kita dapat membuat ratusan salinan informasi yang disimpan dalam media magnetik, bahkan mungkin kita dapat mengubah atau mengganti informasi itu dalam hitungan detik. Kenyataan ini menunjukkan bahwa keamanan informasi yang optimal akan sulit dicapai jika semata-mata hanya bertumpu kepada keamanan fisik. Untuk itu keamanan informasi perlu ditunjang dengan teknik-teknik keamanan yang bersifat non-fisik yang nantinya kita kenal dengan teknik keamanan kriptografi.

Definisi 1.1 *Kriptografi adalah studi teknik matematik yang berkaitan dengan aspek keamanan informasi seperti kerahasiaan, integritas data, autentikasi entitas, dan autentikasi asal data.*

Dari definisi tersebut tersirat bahwa kriptografi tidak hanya sebagai alat yang memberikan keamanan informasi, melainkan juga berupa seperangkat teknik-teknik atau prosedur-prosedur yang berhubungan dengan keamanan informasi.

Tujuan Kriptografi

Sebelas tujuan keamanan informasi telah dinyatakan di atas. Empat diantaranya akan ditekankan menjadi tujuan kriptografi sebagai berikut.

1. *Kerahasiaan* adalah suatu layanan yang digunakan untuk menjaga isi informasi dari semua yang tidak berwenang memilikinya.

2. *Integritas data* adalah suatu layanan yang berkaitan pengubahan data dari pihak-pihak yang tidak berwenang. Untuk menjamin integritas data, kita harus mampu mendeteksi manipulasi data dari pihak-pihak yang tidak berwenang. Manipulasi data diartikan sebagai hal-hal yang berkaitan dengan penghapusan, penyisipan, dan penggantian data.

3. *Autentikasi* adalah suatu layanan yang berhubungan dengan identifikasi entitas dan informasi itu sendiri. Dua pihak terlibat dalam komunikasi seharusnya mengidentikasi dirinya satu sama lain. Informasi yang disampaikan melalui suatu saluran (channel) seharusnya dapat diidentifikasikan asalnya, isinya, tanggal dan waktunya. Atas dasar ini autentikasi terbagi menjadi dua kelas besar, yaitu: *autentikasi entitas* dan *autentikasi asal data.*

4. *Non-repudiasi* adalah suatu layanan yang ditujukan untuk mencegah terjadinya pelanggaran kesepakatan yang telah dibuat sebelumnya oleh entitas. Apabila sengketa muncul ketika suatu entitas mengelak telah melakukan komitmen tertentu, maka suatu alat untuk menangai situasi tersebut diperlukan. Misalnya, suatu entitas mendapatkan wewenang dari entitas lainnya untuk melakukan aksi tertentu, kemudian dia mengingkari wewenang yang diberikan, maka suatu prosedur yang melibatkan pihak ketiga yang terpercaya diperlukan untuk menyelesaikan sengketa itu.

Keempat tujuan kriptografi di atas akan melandasi arah studi kriptografi baik segi teori maupun praktis. Intinya kriptografi merupakan hal-hal yang berkaitan dengan pencegahan dan pendeteksian akan kecurangan, sabotase dan semua aktifitas buruk lainnya yang berhubungan dengan kecurangan.

1.3 Pendefinisian Beberapa Fungsi Matematik untuk Kriptografi

Sebelum kita memasuki inti bahasan kriptografi lebih rinci, pada seksi ini akan dibahas beberapa kosep dasar matematika yang nantinya akan sangat berguna. Dalam hal ini pembicaraan akan ditekankan pada fungsi dan beberapa jenisnya yang berkaitan dengan studi kriptografi.

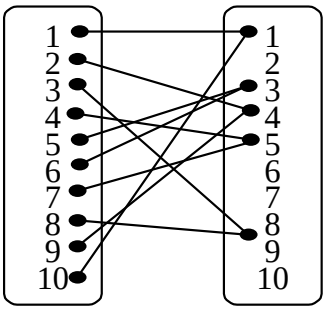
1.3.1 Bijeksi

Definisi 1.2 *Suatu* ***fungsi*** f *dari himpunan* A *ke himpunan* B*, dinotasikan* $f : A \to B$*, didefinisikan sebagai suatu* ***aturan*** *yang memetakan* ***setiap*** *anggota* A *ke* ***tepat satu*** *anggota* B*. Dalam hal ini,* A *disebut* ***domain*** *dan* B *disebut* ***kodomain*** *dari* f*. Jika* $a \in A$*, dan* $b \in B$ *adalah hasil pemetaan dari* a *oleh* f*, maka* b *disebut* ***imej dari*** a *dan* a *disebut* ***preimej dari*** b *oleh* f*, dan dinotasikan* $b = f(a)$*. Himpunan imej dari semua anggata* A *disebut* ***imej dari*** f *dan dinotasikan* $\mathrm{Im}(f)$*. Jelas bahwa* $\mathrm{Im}(f) \subseteq B$*.*

Contoh 1.1 *Misalkan* $A = B = \{1, 2, ..., 10\}$ *dan misalkan* f *adalah suatu aturan yang dinyatakan dengan rumus* $f(x) = r_x$, *dimana* r_x *adalah sisa dari* x^2 *dibagi* 11. *Kita peroleh*

$$\begin{aligned} f(1) &= 1, \quad f(2) = 4, \quad f(3) = 9, \quad f(4) = 5, \quad f(5) = 3 \\ f(6) &= 3, \quad f(7) = 5, \quad f(8) = 9, \quad f(9) = 4, \quad f(10) = 1 \end{aligned}$$

dan $\mathrm{Im}(f) = \{1, 3, 4, 5, 9\}$. *Pemetaan fungsi* f *dari himpunan* A *ke himpunan* B *diilustrasikan sebagai berikut.*



Definisi 1.3 *Suatu* $f : A \to B$ *disebut fungsi* ***satu-satu*** *(1-1) jika setiap anggota dari* B *merupakan imej dari* ***paling banyak satu*** *anggota* A.

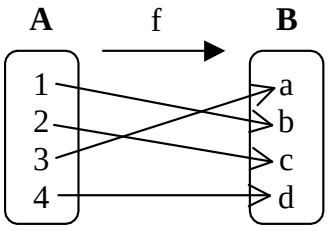
Definisi 1.4 *Suatu* $f : A \to B$ *disebut fungsi* ***onto*** *jika setiap anggota dari* B *merupakan imej dari* ***paling sedikit satu*** *anggota* A.

Definisi 1.5 *Jika* $f : A \to B$ *adalah fungsi* ***satu-satu*** *dan sekaligus fungsi* ***onto***, *maka* f *disebut* ***bijeksi***.

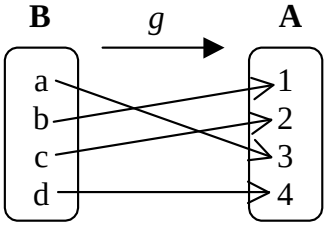
Dalam hal $f : A \to B$ adalah bijeksi, maka $\mathrm{Im}(f) = B$ dan $|A| = |B|$.

Definisi 1.6 *Jika* f *adalah suatu bijeksi dari* A *ke* B, *maka ada suatu bijeksi* g *dari* B *ke* A *sehingga untuk setiap* $b \in B$ *berlaku* $g(b) = a$ *jika dan hanya jika* $f(a) = b$. *Dalam hal demikian* f *disebut* ***invertibel***, g *merupakan* ***invers*** *dari* f *dengan notasi* $g = f^{-1}$.

Contoh 1.2 *Misalkan* $A = \{1, 2, 3, 4\}$ *dan* $B = \{a, b, c, d\}$. *Fungsi* $f : A \to B$ *yang didefinisikan dengan gambar berikut*



adalah bijeksi dan jika $g = f^{-1}$, *maka fungsi* $g : B \rightarrow A$ *yang digambarkan sebagai berikut:*



Di dalam kriptografi bijeksi digunakan untuk *mengenkripsi* pesan, sedangkan invers dari bijeksi digunakan untuk *mendekripsi* pesan. Istilah enkripsi dan dekripsi akan diperkenalkan pada seksi berikutnya. Catatan bahwa jika suatu transformasi bukan merupakan bijeksi, maka dekripsi tidak dijamin menghasilkan pesan yang tunggal.

1.3.2 Fungsi Satu-arah

Diantara tipe-tipe fungsi, di dalam kriptografi fungsi satu-arah (one-way function) memegang peranan sangat penting.

Definisi 1.7 *Suatu fungsi* $f : A \rightarrow B$ *disebut dengan* ***fungsi satu-arah*** *jika untuk setiap* $a \in A$, $f(a)$ *adalah* ***mudah*** *dihitung, tetapi untuk "kebanyakan"* $b \in \text{Im}(f)$ *adalah* ***secara perhitungan tak-layak*** *(computationally infeasible) dapat menentukan* $a \in A$ *sehingga* $b = f(a)$.

Istilah "kebanyakan" dalam definisi di atas mempunyai makna bahwa berdasarkan fakta bisa jadi masih ada sebagian kecil $b \in \text{Im}(f)$ yang secara perhitungan dapat ditentukan $a \in A$ sehingga $b = f(a)$.

Contoh 1.3 *Misalkan* $A = \{1, 2, 3, ..., 16\}$ *dan untuk setiap* $a \in A$ *didefinisikan fungsi* $f(a) = r_a$ *dimana* r_a *adalah sisa dari* 3^a *dibagi* 17, *maka fungsi* f *dapat ditabelkan*

a	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
r_a	3	9	10	13	5	15	11	16	14	8	7	4	12	2	6	1

Diberikan sembarang intejer dari 1 *sampai dengan* 16 *adalah relatif mudah menghitung* $f(a)$. *Akan tetapi apabila diberikan suatu bilangan misalnya* 7, *tanpa melihat tabel secara perhitungan sangat sulit menetukan* $a \in A$ *sehingga* $f(a) = 7$. *Tentu saja kalau bilangan yang diberikan adalah* 3, *maka mudah ditebak bahwa* $a = 1$. *Secara umum, jika diberikan anggota kodomain dari* f *adalah tidak mudah menetukan preimejnya.*

Contoh 1.4 *Pilih bilangan prima* $p = 48611$ *dan* $q = 53993$, *dibentuk bilangan* $n = p \times q = 2624653723$, *dibuat himpunan* $X = \{0, 1, 2, 3, ..., n-1\}$, *dan didefinisikan fungsi* f *pada* X *dengan* $f(x) = r_x$, $\forall x \in X$, *dimana* r_x *adalah sisa dari* x^3 *dibagi* n. *Sebagai ilustrasi,* $f(2489991) = 1981394214$. *Hasil itu diperoleh karena* $(2489991)^3 = 5881949859.n + 1981394214$. *Menghitung* $f(x)$ *adalah relatif mudah dilakukan, tetapi proses balikannya adalah jauh lebih sulit; yaitu diberikan bilangan* r *untuk mencari nilai* x *sedemikian sehingga* x^3 *dibagi* n *sisanya* r. *Prosedur ini dikenal sebagai* ***perhitungan modular*** *akar pangkat* 3 *dengan modulus* n. *Jika faktor dari* n *adalah besar dan tidak diketahui, maka perhitungannya menjadi sangat sulit; dalam arti sejauh ini belum ada algoritme yang efisien. Akan tetapi jika faktor dari* n, *yaitu* p *dan* q, *diketahui, maka telah ada algoritme yang efisien.*

Contoh terakhir di atas mengarah pendefinisian fungsi satu-arah dengan tipe khusus sebagai berikut.

Definisi 1.8 ***Fungsi satu-arah pintu jebakan*** *(trapdoor one-way fuction) adalah fungsi satu-arah* $f : A \rightarrow B$ *yang apabila diberikan informasi ekstra (yang disebut* ***informasi pintu jebakan****), maka perhitungan mencari nilai* $x \in A$, *diketahui* $y \in B$ *sehingga* $y = f(x)$, *menjadi mudah.*

Pada Contoh 1.4, apabila faktor dari $n = 2624653723$ diberikan, yaitu bilangan prima $p = 48611$ dan $q = 53993$, maka perhitungan menginverskan f menjadi lebih mudah (kita akan membahas algoritme tentang ini pada bahasan kriptografi kunci-publik). Akan tetapi bila faktor dari n tidak diberikan, maka problem mencari faktor dari n itu sendiri adalah sangat sulit. Apalagi kalau p dan q mempunyai sekitar 100 dijit desimal, maka komputer tercanggih saat ini pun tidak akan mampu menanganinya. Masalah inilah yang disebut dengan *problem faktorisasi intejer.*

Kembali ke definisi fungsi satu-arah, istilah "perhitungan mudah" dan "secara perhitungan tak-layak" adalah sangat relatif. Oleh karena itu, eksistensi "fungsi satu-arah" dan "fungsi satu-arah pintu jebakan" juga akan menjadi bahan perdebatan karena terkait dengan efisiensi algoritme-algoritmenya. Fakta inilah yang akan menjadi bahan diskusi dalam studi maupun penelitian kriptografi disamping bahasan segi praktisnya.

1.3.3 Permutasi

Ada satu lagi fungsi yang sering kali dipakai dalam berbagai konstruksi kriptografik, yaitu permutasi.

Definisi 1.9 *Misalkan S adalah himpunan berhingga. Suatu* ***permutasi*** *pada S adalah suatu bijeksi dari S ke S.*

Contoh 1.5 *Misalkan $S = \{1, 2, 3, 4, 5\}$, suatu permutasi $p : S \rightarrow S$ didefinisikan sebagai:*

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

Permutasi dapat dinyatakan dalam berbagai macam cara. Salah satu cara penulisan yang cukup sering dipakai adalah sebagai berikut:

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

dimana baris yang atas adalah larik (array) sebagai domain dari p, sedangkan baris yang bawah adalah larik sebagai imej dari p.

$$20!$$

Karena permutasi adalah suatu bijeksi, maka suatu permutasi pasti invertibel. Dalam contoh di atas, invers dari p adalah

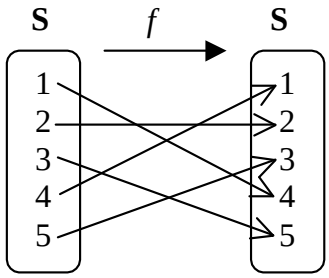
$$p^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 1 & 3 & 2 \end{pmatrix}.$$

Salah satu tipe khusus dari permutasi adalah *involusi* (involution) yang pengertiannya diberikan dalam definisi berikut ini.

Definisi 1.10 *Misalkan S adalah himpunan berhingga dan misalkan f adalah permutasi pada S. Permutasi f disebut* ***involusi***

Definisi 1.11 *jika $f = f^{-1}$. Ini berarti $f(f(x)) = x$ untuk setiap $x \in S$.*

Contoh 1.6 *Suatu involusi f pada $S = \{1, 2, 3, 4, 5\}$ dapat gambar berikut ini:*



Terlihat bahwa f membawa: 1 *ke* 4 *dan* 4 *ke* 1, 2 *ke* 2 *dan* 2 *ke* 2, 3 *ke* 5 *dan* 5 *ke* 3, 4 *ke* 1 *dan* 1 *ke* 4, 5 *ke* 3 *dan* 3 *ke* 5. *Jadi* $f(f(x)) = x$ *untuk setiap* $x \in S$.

1.4 Konsep dan Terminologi Dasar

Sebagaimana displin ilmu yang lainnya, disini kita juga akan membuat pendefinisian yang ketat yang berlandaskan konsep dasar. Seiring dengan itu dibuat pula istilah-istilah dan terminologi yang dalam hal ini kadangkala ada perbedaan antara penulis satu dengan yang lainnya.

1.4.1 Transformasi Enkripsi dan Dekripsi

- Dinotasikan $\mathcal{A}$ sebagai himpunan berhingga yang disebut *definisi alfabet*. Misalnya, $\mathcal{A} = \{0, 1\}$ disebut *alfabet biner* dan ini paling sering kali digunakan dalam pendefinisian alfabet. Catatan bahwa alfabet apapun dapat dikoding menjadi alfaber biner. Sebagai contoh, karena ada 32 string biner yang panjangnya 5, maka setiap huruf dalam alfabet Latin dapat diganti dengan satu string biner yang panjangnya

$$2^{64}$$

- : 18 446 744 073 709 551 616

- Dinotasikan $\mathcal{M}$ sebagai himpunan berhingga yang disebut *ruang pesan* (message space). $\mathcal{M}$ beranggotakan string-string simbol dari definisi alfabet. Suatu anggota $\mathcal{M}$ disebut dengan *pesan plainteks* (plaintext message), sederhananya disebut dengan *plainteks*. Misalnya, anggota-anggota $\mathcal{M}$ bisa berupa string biner, teks bahasa Inggris, kode komputer, dll.

- Dinotasikan $\mathcal{C}$ sebagai himpunan berhingga yang disebut *ruang sifer-teks* (ciphertext space). $\mathcal{C}$ beranggotakan string-string simbol dari

definisi alfabet yang boleh berbeda dari definisi alfabet untuk $\mathcal{M}$. Suatu anggota dari $\mathcal{C}$ disebut dengan *siferteks* (ciphertext).

- Dinotasikan $\mathcal{K}$ sebagai himpunan berhingga yang disebut *ruang kunci* (key space). Suatu anggota dari $\mathcal{K}$ disebut dengan *kunci.*

- Setiap anggota $e \in \mathcal{K}$ menentukan tepat satu bijeksi dari $\mathcal{M}$ ke $\mathcal{C}$, dinotasikan dengan E_e. Dalam hal ini, E_e disebut dengan *fungsi enkripsi* (encryption fucntion) atau *transformasi enkripsi* (encryption transformation). Catatan bahwa E_e harus merupakan bijeksi karena, jika proses diinverskan, akan didapatkan tepat satu plainteks untuk setiap siferteks yang berbeda. Proses penerapan transformasi E_e pada plainteks $m \in \mathcal{M}$ dinamakan *enkripsi dari* m dengan kunci e.

- Untuk setiap $d \in \mathcal{K}$, D_d menotasikan suatu bijeksi dari $\mathcal{C}$ ke $\mathcal{M}$. Dalam hal ini, D_d disebut dengan *fungsi dekripsi* (decryption fucntion) atau *transformasi dekripsi* (decryption transformation). Proses penerapan transformasi D_d pada siferteks $c \in \mathcal{C}$ dinamakan *dekripsi dari* c.

- Suatu *skema enkripsi* terdiri atas himpunan semua transformasi enkripsi $\{E_e/e \in \mathcal{K}\}$ dikaitkan himpunan transformasi dekripsi $\{D_d/d \in \mathcal{K}\}$ dengan sifat bahwa untuk setiap $e \in \mathcal{K}$ menentukan tepat satu $d \in \mathcal{K}$ sehingga $D_d = E_e^{-1}$. Dalam hal ini, $D_d(E_e(m)) = m$ untuk setiap $m \in \mathcal{M}$. Kadang kala skema enkripsi disebut juga dengan *sandi* (cipher).

- Dua kunci e dan d sedemikian sehingga $D_d = E_e^{-1}$ disebut dengan *pasangan kunci*, biasanya dinotasikan dengan (e, d). Catatan bahwa bisa terjadi $e = d$.

- Untuk mengkonstruksi skema enkripsi, kita memerlukan pendefinisian suatu ruang pesan $\mathcal{M}$, ruang siferteks $\mathcal{C}$, ruang kunci $\mathcal{K}$, himpunan transformasi enkripsi $\{E_e/e \in \mathcal{K}\}$ dan himpunan transformasi dekripsi yang terkait $\{D_d/d \in \mathcal{K}\}$.

$$2^{128}$$

: 340 282 366 920 938 463 463 374 607 431 768 211 456

1.4.2 Pencapaian Kerahasiaan

Suatu skema enkripsi dapat digunakan untuk tujuan kerahasiaan sebagai berikut. Dua pihak, yaitu Ali dan Budi, memilih atau saling tukar secara rahasia pasangan kunci (e, d). Kemudian jika Ali ingin mengirim pesan $m \in \mathcal{M}$ kepada Budi, maka yang dilakukan oleh Ali adalah mengubah pesan m

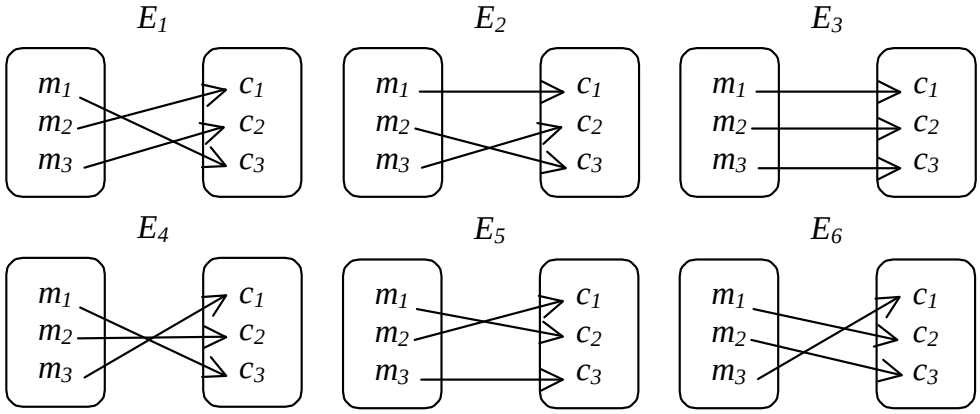
menjadi siferteks c dengan menggunakan kunci e (artinya Ali menghitung $c = E_e(m)$) kemudian c dikirimkan ke Budi. Untuk mengetahui pesan Ali, yang dilakukan oleh Budi adalah mengubah c menjadi m dengan cara menghitung $D_d(c) = m$.

Dari skema sederhana di atas, pertanyaan yang muncul adalah mengapa harus pasangan kunci yang dipilih (mengapa tidak pasangan transformasi enkripsi dan dekripsinya saja). Alasan yang cukup kuat menjawab pertanyaan tersebut adalah terkait dengan hal praktis, yaitu apabila kerahasiaan kunci sudah diragukan, maka yang diganti cukup kuncinya saja (bukannya membuat transformasi baru atau mengubah skema secara keseluruhan). Hal ini bisa diilustrasikan dalam kasus sehari-hari, misalnya sebuah pintu rumah dilengkapi dengan gembok kombinasi yang bisa disetel ulang. Pemilik rumah pertama kali menetapkan kombinasi rahasia pada gembok sehingga siapapun yang tidak mengetahui kombinasi rahasianya akan kesulitan membuka pintu. Begitu kerahasiaan kombinasi mulai terungkap, maka pemilik rumah cukup mengubah kombinasi rahasianya, bukan mengganti gemboknya.

Contoh 1.7 *(skema enkripsi) Misalkan*

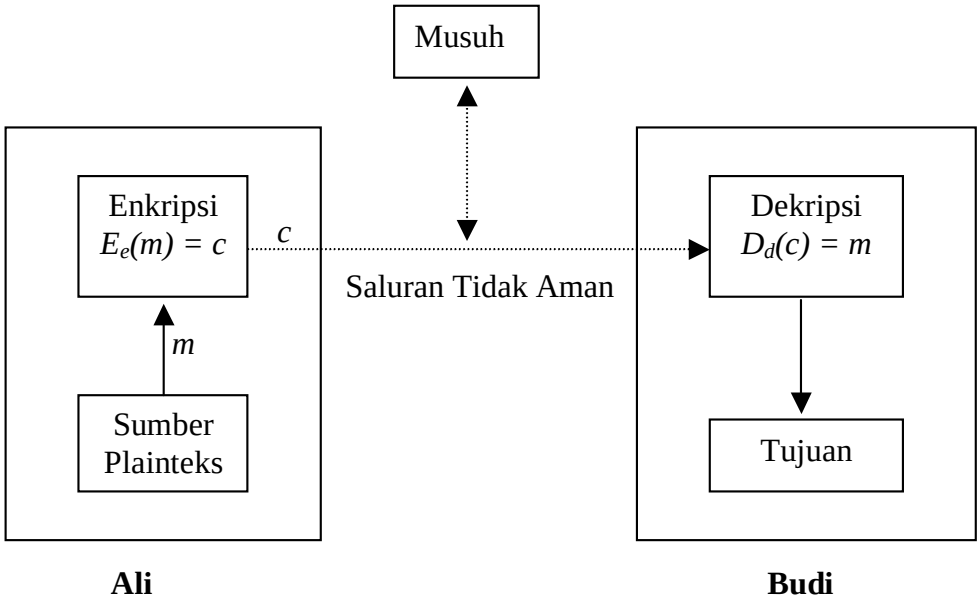
$$\mathcal{M} = \{m_1, m_2, m_3\} \text{ dan } \mathcal{C} = \{c_1, c_2, c_3\},$$

maka ada $3! = 6$ *bijeksi dari* $\mathcal{M}$ *ke* $\mathcal{C}$. *Didefinisikan ruang kunci* $\mathcal{K} = \{1, 2, 3, 4, 5, 6\}$ *sehingga* $i \in \mathcal{K}$ *menspesifikasikan transformasi* E_i *sebagai berikut.*



Ali dan Budi sepakat untuk menggunakan transformasi E_1. *Untuk mengenkripsi pesan* m_1, *Ali menghitung* $E_1(m_1) = c_3$ *dan mengirimkan* c_3 *ke Budi. Kemudian Budi mendekripsi* c_3 *dengan menginverskan tanda panah pada* E_1 *dan didapatkan* c_3 *menjadi* m_1.

Gambar 1 memberikan model yang sederhana dari komunikasi dua pihak dengan menggunakan enkripsi.



Gambar 1: Skematik komunikasi dua partai menggunakan enkripsi.

Mengacu pada Gambar 1 berikut ini beberapa terminologi yang terkait.

- Suatu *entitas* atau *partai* adalah seseorang atau sesuatu yang mengirim, menerima, atau memanipulasi informasi. Ali dan Budi adalah contoh entitas. Suatu entitas bisa berupa orang, terminal komputer, kartu kredit, dll.
- *Pengirim* (sender) adalah suatu entitas di dalam komunikasi dua partai yang secara sah berhak mengirimkan informasi.
- *Penerima* (receiver) adalah suatu entitas di dalam komunikasi dua partai yang secara sah berhak menerima informasi yang dikirim.
- *Musuh* (Adversary) adalah suatu entitas di dalam komunikasi dua partai yang bukan pengirim maupun penerima dan yang berusaha untuk membongkar keamanan informasi yang telah dibuat oleh pengirim dan penerima.
- *Saluran* (channel) adalah suatu alat yang digunakan untuk menyampaikan informasi dari suatu entitas ke entitas yang lain.
- *Saluran aman secara fisik* (secure channel) adalah saluran yang secara fisik tidak mungkin dibobol oleh musuh.

- *Saluran tidak aman* (unsecured channel) adalah saluran yang memungkinkan siapa saja dapat membaca, menyalin, menghapus, merekam, dan menyisipkan informasi.
- *Saluran aman* (secured channel) adalah saluran yang dijamin bahwa musuh tidak akan mampu membaca, menyalin, menghapus, merekam, dan menyisipkan informasi.

Perlu dicatat untuk dua terminologi yang berasal dari bahasa Inggris, yaitu: "secure channel" dan "secured channel". Secara harfiah kedua istilah itu mempunyai makna "saluran aman". Akan tetapi dalam bahasan kita nanti, kedua isilah tersebut mempunyai perbedaan makna yang cukup mendasar. "Secure channel" bermakna saluran yang aman secara fisik, sedangkan "secured channel" mempunyai makna aman yang lebih kuat, yaitu baik aman secara fisik maupun secara kriptografik.

1.4.3 Keamanan Kriptografik

Asumsi dasar yang dipakai dalam kriptografi adalah bahwa himpunan-himpunan $\mathcal{M}$, $\mathcal{C}$, $\mathcal{K}$, $\{E_e/e \in K\}$, dan $\{D_d/d \in K\}$ diketahui oleh publik. Jika dua partai ingin berkomunikasi secara aman, maka satu hal yang harus dijaga kerahasiaanya adalah pasangan kunci (e, d) yang mereka gunakan dan yang mereka harus pilih. Keamanan tambahan yang bisa digunakan adalah menjaga kerahasiaan himpunan transformasi enkripsi dan dekripsi. Namun berdasarkan pengalaman mempertahankan keamanan transformasi adalah sangat sulit.

Definisi 1.12 *Suatu skema enkripsi dikatakan* ***bobol*** *(breakable), apabila partai ketiga, tanpa pengetahuan sebelumnya tentang pasangan kunci* (e, d)*, secara sistematik mampu mendapatkan plainteks dari siferteks terkait dalam masa berlakunya kerahasiaan informasi.*

Suatu skema enkripsi dapat dibobol dengan cara mencoba semua kunci yang mungkin untuk mencari mana yang cocok (diasumsikan bahwa himpunan transformasi enkripsi diketahui publik). Cara ini disebut dengan *pelacakan lengkap* (exhaustive search). Untuk menangkal pelacakan itu, perancang skema enkripsi dianjurkan menggunakan ukuran ruang kunci yang cukup besar.

Serangkaian persyaratan untuk sistem sandi, di dalam literatur sering kali mengacu kepada *desiderata Kerckoff*. Diantaranya ada enam yang utama yang dinyatakan berikut ini.

1. Sistem seharusnya, jika tidak takterbobol secara teoritik, pasti takterbobol secara praktis.
2. Kesepakatan tentang rincian sistem seharusnya dibuat senyaman mungkin antar koresponden.
3. Kunci seharusnya mudah diingat tanpa catatan dan mudah diubah.
4. Kriptogram sebaiknya dikirim melalui telegrap.
5. Perangkat enkripsi seharusnya mudah dibawa dan mudah dioperasikan oleh satu orang.
6. Sistem seharusnya mudah, tanpa aturan yang berbelit-belit dan tidak mengakibatkan ketegangan jiwa.

Walaupun keenam pernyataan di atas disampaikan pada tahun 1883, sebagian besar masih relevan pada masa kini.

Terminologi yang telah diberikan di atas terbatas untuk enkripsi dan dekripsi dengan tujuan privasi (dalam arti kerahasiaan). Keamanan informasi sebenarnya mempunyai tujuan yang lebih luas lagi, mengarah ke hal-hal seperti autentikasi (keabsahan) dan integritas data (keutuhan data). Untuk itu, berikut ini akan diberikan beberapa definisi yang lebih umum.

- *Layanan keamanan informasi* adalah suatu metode untuk memberikan beberapa aspek khusus tentang keamanan. Misalnya, integritas dari suatu data yang dikirim adalah tujuan keamanan, dan metode yang menjamin keutuhan data itu merupakan layanan keamanan informasi.
- *Pembobolan* suatu layanan keamanan informasi mengakibatkan gagalnya tujuan layanan yang dimaksud.
- *Musuh pasif* adalah musuh yang hanya mampu membaca informasi dari saluran tak-aman.
- *Musuh aktif* adalah mampu memanipulasi informasi dari saluran takaman.
- *Kriptanalisis* (Cryptanalysis) adalah studi teknik matematik untuk mencoba mematahkan teknik kritografik, dan lebih umum lagi layanan keamanan informasi.
- *Kriptanalis* (Cryptanalyst) adalah orang yang menggeluti kriptanalisis.

- *Kriptologi* adalah studi tentang kriptografi dan kriptanalisis.
- *Kriptosistem* adalah istilah umum untuk himpunan primitif kriptografi yang digunakan untuk memberikan layanan keamanan kriptografik. Paling sering istilah ini digunakan untuk hal-hal yang berhubungan dengan primitif enkripsi.

1.4.4 Serangan Keamanaan Skema Enkripsi

Tujuan serangan keamaan pada dasarnya secara sistematik mendapatkan plainteks dari diketahuinya siferteks, atau lebih kuat lagi bertujuan untuk mendapatkan kunci dekripsi. Berikut diberikan empat diatara beberapa tipe serangan pada skema enkripsi.

1. *Serangan hanya-siferteks* (Ciphertext-only attack). Kriptanalis mempunyai semua siferteks dari plainteks yang telah dienkripsi dengan menggunakan algoritme yang sama. Tujuannya mendapatkan plainteks sebanyak mungkin, atau lebih baik lagi mendapatkan kuncinya untuk digunakan mendekripsi semua siferteks yang dimilikinya.

   Diberikan: $c_1 = E_e(m_1)$, $c_2 = E_e(m_2)$, $c_i = E_e(m_i)$.

   Diperoleh: $m_1, m_2, ..., m_i$, atau kunci dekripsi d sehingga didapatkan semua $m_i = D_d(c_i)$

2. *Serangan tahu-plainteks* (Known-plaintext attack). Kriptanilis tidak hanya mengetahui semua siferteks, tetapi juga mempunyai akses (mengetahui) beberapa pasangan plainteks-siferteks. Tujuannya mendapatkan sebanyak mungkin plainteks, atau lebih baik lagi mendapatkan kunci dekripsi untuk memperoleh semua plainteks.

   Diberikan: $m_1, c_1 = E_e(m_1)$; $m_2, c_2 = E_e(m_2)$; $m_i, c_i = E_e(m_i)$.

   Diperoleh: m_{i+1}, m_{i+2}, ..., m_s, atau kunci dekripsi d sehingga didapatkan semua $m_i = D_d(c_i)$.

3. *Serangan pilih-plainteks* (Chosen-plaintext attack). Kriptanilis tidak hanya mengetahui semua siferteks dan beberapa pasangan plainteks-siferteks, tetapi juga mampu memilih plainteks tersebut dan enkripsinya. Tujuannya mendapatkan sebanyak mungkin plainteks, atau lebih baik lagi mendapatkan kunci dekripsi untuk memperoleh semua plainteks.

Diberikan: $m_1, c_1 = E_e(m_1)$; $m_2, c_2 = E_e(m_2)$; $m_i, c_i = E_e(m_i)$. Dalam hal ini, m_i didapatkan dengan cara memilih.

Diperoleh: m_{i+1}, m_{i+2}, ..., m_s, atau kunci dekripsi d sehingga didapatkan semua $m_i = D_d(c_i)$.

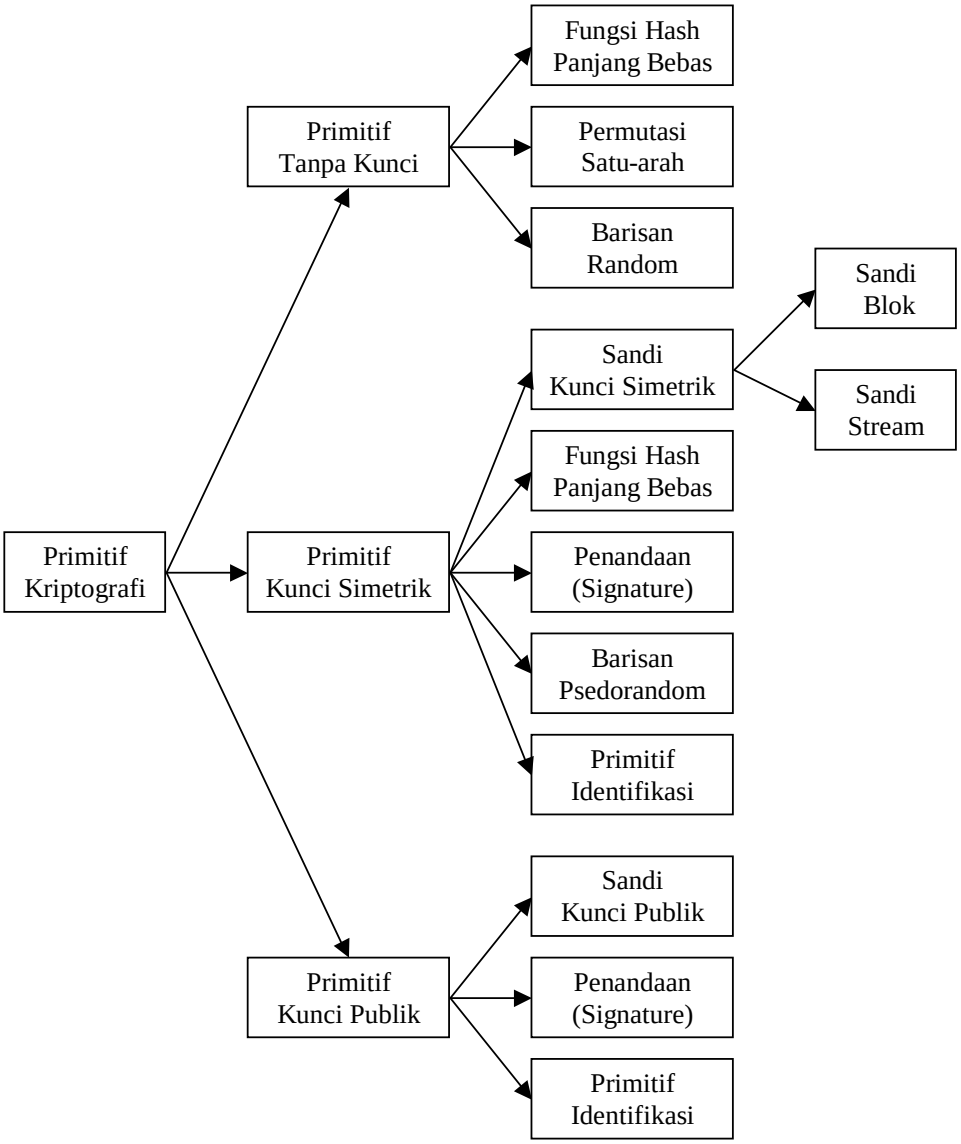
4. *Serangan pilih-siferteks* (Chosen-ciphertext attack). Kriptanilis tidak hanya mengetahui semua siferteks, tetapi juga mampu memilih beberapa siferteks dan dekripsinya. Tujuannya mendapatkan kunci dekripsi untuk memperoleh semua plainteks.

   Diberikan: $c_1, m_1 = D_d(c_1)$; $c_2, m_2 = D_d(c_2)$; $c_i, m_i = D_d(c_i)$. Dalam hal ini, c_i didapatkan dengan cara memilih.

   Diperoleh: kunci dekripsi d sehingga didapatkan semua $m_i = D_d(c_i)$.

1.4.5 Taksonomi Primitif Kriptografi

Sebagai akhir bab ini, berikut diberikan sejumlah primitif kriptografi yang digunakan untuk memberikan keamanan informasi. Hubungan antar primitif dapat digambarkan dalam bentuk skema pada Gambar 2. Beberapa primitif akan dibahas pada bab-bab selanjutnya sebagai sebagai materi pada kuliah ini.



Gambar 2: Taksonomi Primitif Kriptografi.

Chapter 2

Enkripsi Kunci Simetrik

Enkripsi kunci simetrik membahas primitif kriptografi yang utamanya untuk tujuan kerahasiaan dengan bahasan meliputi:

- Konsep Enkripsi Kunci Simetrik,
- Teknik Enkripsi Klasik.
- Sandi Blok Kunci Simetrik
- Sandi Alir Kunci Simetrik

2.1 Konsep Enkripsi Kunci Simetrik

Pada subbab ini bahasan akan ditekankan pada salah satu jenis enkripsi yaitu kunci simetrik. Enkripsi ini didasarkan pada skema yang intinya menggunakan satu kunci, dan berikut ini diberikan definisinya.

Definisi 2.1 *Suatu skema enkripsi disebut* ***enkripsi kunci simetrik****, jika untuk setiap pasangan kunci enkripsi dan dekripsi* (e, d)*, secara komputasi* d *"mudah" dihitung apabila* e *diketahui, dan* e *"mudah" dihitung apabila* d *diketahui.*

Karena di dalam praktik paling sering kali digunakan bahwa $d = e$, maka enkripsi kunci simetrik disebut juga dengan enkripsi *satu kunci, kunci tunggal, kunci pribadi,* atau *konvensional.*

Contoh 2.1 *Diberikan* $\mathcal{A} = \{A, B, C, ..., Y, Z\}$ *adalah alfabet dari bahasa Inggris,* $\mathcal{M}$ *dan* $\mathcal{C}$ *merupakan himpunan semua string atas* $\mathcal{A}$ *yang panjangnya* 5. *Kunci* e *dipilih suatu permutasi pada* $\mathcal{A}$. *Untuk mengenkripsi, suatu pesan dalam bahasa Inggris dipilah-pilah dalam grup dengan anggota lima huruf (jika panjang pesan bukan kelipatan dari lima, ditambahkan huruf khusus) dan permutasi* e *diterapkan pada masing-masing huruf satu demi satu. Untuk mendekripsi, invers permutasi* $d = e^{-1}$ *diterapkan pada masing-masing huruf dari siferteks. Sebagai contoh, dipilih* e *adalah permutasi*

$$\begin{pmatrix} A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z \\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z\ A\ B\ C \end{pmatrix}.$$

dan misalkan pesannya adalah "this chiper is certainly not secure", berarti

$$m = THISC\ IPHER\ ISCER\ TAINL\ YNOTS\ ECURE.$$

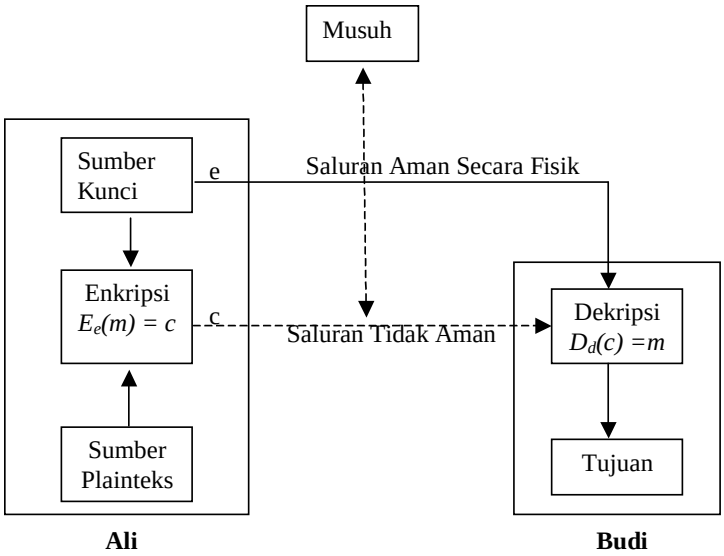
Maka m *dienkripsi menjadi*

$$c = E_e(m) = WKLVF\ LSKHU\ LVFHU\ WDLQO\ BQRWV\ HFXUH.$$

26!

: 403 291 461 126 605 635 584 000 000

Skema enkripsi dua partai menggunakan kunci simetrik diberikan pada bagan berikut ini.



Gambar 3: Komunikasi dua partai menggunakan enkripsi kunci simetrik.

2.2 Teknik Enkripsi Klasik

2.2.1 Sandi Substitusi

Sandi substitusi adalah siferteks blok yang mengganti simbol (atau grup simbol) dengan simbol (atau grup simbol) yang lain.

Sandi Caesar

Pada sandi Caesar, diberikan alfabet $\mathcal{A} = \{A, B, C, ..., Y, Z\}$ adalah alfabet dari bahasa Inggris. $\mathcal{M}$ dan $\mathcal{C}$ merupakan himpunan semua string atas $\mathcal{A}$ yang panjangnya t (nantinya t disebut ukuran blok). Didefinisikan pula ruang kunci $\mathcal{K} = \{0, 1, 2, ..., 25\}$, dan setiap kunci $e \in \mathcal{K}$, transformasi enkripsi E_e adalah permutasi *geser ke kiri putar melingkar sejauh e satuan* pada pada $\mathcal{A}$. Sedengkan, transformasi dekripsi D_e adalah permutasi *geser ke kanan putar melingkar sejauh e satuan* pada pada $\mathcal{A}$. Secara numerik sandi Caesar didefinisikan sebagai berikut.

Definisi 2.2 *Didefinisikan:*

$$\begin{aligned} \mathcal{A} &= \{0, 1, 2, ..., 24, 25\}, \\ \mathcal{M} &= \{\mathbf{m} = (m_1 m_2 ... m_t)/m_i \in \mathcal{A}\}, \\ \mathcal{C} &= \{\mathbf{c} = (c_1 c_2 ... c_t)/c_i \in \mathcal{A}\}, \ \textit{dan} \\ \mathcal{K} &= \{0, 1, 2, ..., 25\}. \end{aligned}$$

Untuk suatu $e \in \mathcal{K}$, *transformasi enkripsi*

$$E_e(m) = (m + e) \bmod 26, \quad \forall m \in \mathcal{A},$$

sehingga E_e *mengenkripsi pesan* $\mathbf{m} = (m_1 m_2 ... m_t)$ *menjadi siferteks*

$$\begin{aligned} E_e((m_1 m_2 ... m_t)) &= (c_1 c_2 ... c_t), \ \textit{dimana} \\ c_i &= (m_i - e) \bmod 26. \end{aligned}$$

Transformasi dekripsinya adalah

$$D_e(c) = (c - e) \bmod 26, \quad \forall p \in \mathcal{A}$$

$$\begin{aligned} D_e((c_1 c_2 ... c_t)) &= (m_1 m_2 ... m_t), \ \textit{dimana} \\ m_i &= (c_i + e) \bmod 26 \Leftrightarrow \end{aligned}$$

Dapat ditulis,

$$m_i = (c_i + d) \bmod 26, \ \textit{dengan } d = 26 - e.$$

Dari definisi di atas, perhatikan bahwa tranformasi dekripsinya

$$D_e = E_e^{-1} = E_{-e} = E_d, \text{ dimana } d = -e = 26 - e.$$

Hal ini bisa dimaknai bahwa algoritme dekripsi dan enkripsi adalah sama, hanya kuncinya yang berbeda: e adalah kunci enkripsi sedangkan $d = 26 - e$ adalah kunci dekripsi.

Sandi Caesar memberikan ukuran kunci yang sangat kecil, $|\mathcal{K}| = 26$, sehingga pelacakan lengkap hanya membutuhkan paling banyak 26 percobaan kunci. Contoh 2.1 merupakan suatu contoh sandi Caesar dengan $t = 5$, $e = 3$, dan $d = 23$.

Substitusi Sederhana

Lebih umum dari pada Sandi Caesar adalah sandi subtitusi sederhana yang didefinisikan berikut ini.

Definisi 2.3 *Misalkan $\mathcal{A}$ adalah alfabet yang beranggota q simbol, $\mathcal{M}$ adalah himpunan semua string atas $\mathcal{A}$ dengan panjang t, dan $\mathcal{K}$ adalah himpunan semua permutasi pada $\mathcal{A}$. Didefinisikan untuk setiap $e \in \mathcal{K}$, tranformasi enkripsi sebagai*

$$E_e(\mathbf{m}) = (e(m_1)e(m_2)...e(m_t)) = (c_1c_2...c_t) = \mathbf{c},$$

dimana $\mathbf{m} = (m_1m_2...m_t) \in \mathcal{M}$. Dengan kata lain, untuk setiap simbol dalam rangkai-t diganti simbol lain dalam $\mathcal{A}$ berdasarkan aturan permutasi dari e. Untuk mendekripsi $\mathbf{c} = (c_1c_2...c_t)$ dihitung invers $d = e^{-1}$ dan

$$D_d(\mathbf{c}) = (d(c_1)d(c_2)...d(c_t)) = (m_1m_2...m_t) = \mathbf{m}.$$

*E_e disebut **sandi substitusi sederhana** atau **sandi substitusi mono-alfabetik**.*

Dari definisi di atas jelas bahwa ukuran kunci adalah $|\mathcal{K}| = q!$ dan tidak bergantung kepada ukuran blok t. Sandi substitusi sederhana dengan ukuran blok yang kecil memberikan keamanan yang tak-layak bahkan walaupun ukuran kuncinya sangat besar. Dalam sandi modern rata-rata ukuran bloknya ≥ 64 bit.

26!

: 403 291 461 126 605 635 584 000 000:

Contoh 2.2 *Pada Contoh 2.1, dipilih kunci enkripsi e adalah permutasi*

$$e = \begin{pmatrix} A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z \\ Y\ Q\ F\ C\ H\ X\ J\ K\ T\ M\ V\ O\ B\ E\ R\ S\ L\ U\ N\ W\ I\ D\ Z\ A\ P\ G \end{pmatrix},$$

maka dengan mudah didapatkan kunci dekripsi d adalah permutasi

$$d = \begin{pmatrix} A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z \\ X\ M\ D\ V\ N\ C\ Z\ E\ U\ G\ H\ Q\ J\ S\ L\ Y\ B\ O\ P\ I\ R\ K\ T\ F\ A\ W \end{pmatrix}.$$

Enkripsi dari

$$m = THISC\ IPHER\ ISCER\ TAINL\ YNOTS\ ECURE.$$

adalah

$$c = WKTNF\ TSKHU\ TNFHU\ WYTEO\ PERWN\ HFIUH.$$

Contoh 2.2 merupakan suatu contoh sandi substitusi sederhana dengan panjang blok lima. Sandi ini memberikan ukuran ruang kunci yang cukup besar, yaitu $26! \approx 4 \times 10^{26}$, akan tetapi kunci yang digunakan mudah dilacak dengan cara memeriksa modus jumlah siferteks. Ini akibat dari pengamatan sederhana distribusi frekuensi huruf yang ada di dalam siferteks. Misalnya, fakta bahwa huruf E adalah paling sering dipakai di dalam kebanyakan teks bahasa Inggris. Dengan demikian huruf yang paling sering muncul di dalam blok siferteks adalah huruf yang paling mungkin berasal dari huruf E di dalam plainteks. Jadi, dengan mengamati modus jumlah blok siferteks, kriptanalis bisa menentukan kuncinya.

$$2^{56}$$

Sandi Substitusi Homofonik

Definisi 2.4 *Pada masing-masing simbol $a \in \mathcal{A}$, didefinisikan himpunan $H(a)$ beranggota string dengan t simbol, dengan syarat saling asing satu sama lain.* ***Sandi substitusi homofonik*** *mengganti masing-masing simbol a dengan satu string yang dipilih secara random dari anggota-anggota $H(a)$. Mendekripsi suatu string $\mathbf{c}$ dengan t simbol berarti menentukan $a \in \mathcal{A}$ sedemikian sehingga $\mathbf{c} \in H(a)$. Kunci untuk sandi ini terbentuk dari himpunan-himpunan $H(a)$.*

Contoh 2.3 *Diberikan $\mathcal{A} = \{a, b\}$, $H(a) = \{00, 10\}$, dan $H(b) = \{01, 11\}$. Blok pesan plainteks ab dienkripsi ke salah satu dari:* 0001, 0011, 1001, 1011. *Perhatikan bahwa kodomain dari transformasi enkripsi untuk pesan dua simbol adalah himpunan yang terbentuk dari bitstring 4-simbol, dan himpunan itu yang saling asing sebagaimana digambarkan berikut:*

$$\begin{aligned} aa &\longrightarrow \{0000, 0010, 1000, 1010\} \\ ab &\longrightarrow \{0001, 0011, 1001, 1011\} \\ ba &\longrightarrow \{0100, 0110, 1100, 1110\} \\ bb &\longrightarrow \{0101, 0111, 1101, 1111\} \end{aligned}$$

Sembarang bitstring 4-simbol menentukan satu anggota dari kodomain, dan pada gilirannya menentukan satu pesan plainteks.

Seringkali simbol tidak terjadi dengan frekuensi yang sama di dalam pesan plainteks. Dengan sandi substitusi sederhana sifat frekuensi tak-seragam itu direfleksikan ke dalam siferteks sebagaimana terlihat pada Contoh 2.2. Sandi homofonik dapat digunakan untuk membuat frekuensi kejadian simbol siferteks lebih seragam.

Sandi Substitusi Polialfabetik

Definisi 2.5 *Sandi substitusi polialfabetik adalah sandi blok dengan panjang blok t atas alfabet $\mathcal{A}$ memenuhi sifat-sifat berikut:*

(i) ruang kunci adalah himpunan semua rangkai–t terurut $(p_1, p_2, ..., p_t)$, dimana p_i, $1 \leq i \leq t$, adalah permutasi pada $\mathcal{A}$,

(ii) enkripsi pesan $\mathbf{m} = (m_1, m_2, ..., m_t)$ dengan kunci $\mathbf{e} = (p_1, p_2, ..., p_t)$ diberikan oleh

$$\begin{aligned} E_{\mathbf{e}}(\mathbf{m}) &= (p_1(m_1), p_2(m_2), ..., p_t(m_t)) \\ &= (c_1, c_2, ..., c_t) \\ &= \mathbf{c}, \end{aligned}$$

(iii) dekripsi $\mathbf{c}$ menggunakan pasangan kunci $\mathbf{d} = \mathbf{e}^{-1} = (p_1^{-1}, p_2^{-1}, ..., p_t^{-1})$, diberikan oleh

$$\begin{aligned} D_{\mathbf{d}}(\mathbf{c}) &= (p_1^{-1}(c_1), p_2^{-1}(c_2), ..., p_t^{-1}(c_t)) \\ &= (m_1, m_2, ..., m_t) \\ &= \mathbf{m}, \end{aligned}$$

Contoh 2.4 *(Sandi Vigenère) Misalkan $\mathcal{A} = \{A, B, C, ..., Z\}$ dan $t = 3$. Pilih $\mathbf{e} = (p_1, p_2, p_3)$ dengan*

$$p_1 = \begin{pmatrix} A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z \\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z\ A\ B\ C \end{pmatrix}$$

$$p_2 = \begin{pmatrix} A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z \\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z\ A\ B\ C\ D\ E\ F\ G \end{pmatrix}$$

$$p_3 = \begin{pmatrix} A\ B\ C\ D\ E\ F\ G\ H\ I\ J\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z \\ K\ L\ M\ N\ O\ P\ Q\ R\ S\ T\ U\ V\ W\ X\ Y\ Z\ A\ B\ C\ D\ E\ F\ G\ H\ I\ J \end{pmatrix}$$

Jika

$$\mathbf{m} = THI\ SCI\ PHE\ RIS\ CER\ TAI\ NLY\ NOT\ SEC\ URE,$$

maka

$$\mathbf{c} = WOS\ VJS\ SOO\ UPC\ FLB\ WHS\ QSI\ QVD\ VLM\ XYO.$$

Sandi polialfabetik mempunyai kelebihan dibandingkan dengan sandi substitusi sederhana karena frekuensi simbol tidak dipertahankan. Pada contoh di atas terlihat bahwa huruf E dienkripsi menjadi dua huruf, yaitu O dan L. Walaupun demikian sandi polialfabetik tidak secara signifikan lebih sulit untuk dikriptanalisis, bahkan pendekatan pemecahannya pun sama dengan sandi substitusi sederhana. Faktanya, begitu panjang blok t dapat ditentukan, huruf-huruf pada siferteks dapat dibagi menjadi t grup. Grup i, untuk $1 \leq i \leq t$, terdiri dari huruf-huruf siferteks hasil transformasi permutasi p_i, dan kemudian masing-masing grup dianalisis frekuensinya.

Sandi Hill

Sandi Hill merupakan salah satu sandi yang teknik enkripsi dan dekripsinya menggunakan sifat operasi aljabar (dalam hal ini aljabar matriks). Keteraturan struktur di dalam aljabar mengakibatkan rentan terhadap kriptanalisis.

Definisi 2.6 *Didefinisikan:*

$$\begin{aligned} \mathcal{A} &= \{0, 1, 2, ..., 24, 25\} \text{ sebagai representasi alfabet Inggris.} \\ \mathcal{M} &= \{\mathbf{m} = (m_1 m_2 ... m_t)/m_i \in \mathcal{A}\}, \\ \mathcal{C} &= \{\mathbf{c} = (c_1 c_2 ... c_t)/c_i \in \mathcal{A}\}, \text{ dan} \end{aligned}$$

$$\mathcal{K} = \{K = [a_{ij}]_{i,j=1}^{t} / a_{ij} \in \mathcal{A}, \quad \gcd(|K|, 26) = 1\}$$

Untuk suatu kunci enkripsi $K \in \mathcal{K}$, transformasi enkripsi

$$E_K(\mathbf{m}) = K\mathbf{m}^T \bmod 26 = \mathbf{c}^T$$

Kunci dekripsinya adalah K^{-1}, sehingga transformasi dekripsinya

$$E_{K^{-1}}(\mathbf{c}) = K^{-1}\mathbf{c}^T \bmod 26 = \mathbf{m}^T.$$

$$K = \begin{bmatrix} 1 & 2 \\ 7 & 5 \end{bmatrix}$$

$$K^{-1} = \begin{bmatrix} 1 & 2 \\ 7 & 5 \end{bmatrix}^{-1} \bmod 26$$

$$: \begin{bmatrix} 11 & 6 \\ 21 & 23 \end{bmatrix} \begin{bmatrix} 4 \\ 7 \end{bmatrix} \bmod 26 = \begin{bmatrix} 8 \\ 11 \end{bmatrix} = \begin{bmatrix} 86 \\ 245 \end{bmatrix} :$$

$$21 \times 7 \bmod 26$$

$$17 \times 11 \bmod 26$$

$: 5 : 15 : 21 : 9$

$$\begin{bmatrix} 17 & 1 \\ 11 & 14 \end{bmatrix} \begin{bmatrix} 9 \\ 10 \end{bmatrix} \bmod 26 = \begin{bmatrix} 7 \\ 5 \end{bmatrix}$$

$$\begin{bmatrix} 17 & 1 \\ 11 & 14 \end{bmatrix}^{-1} \begin{bmatrix} 7 \\ 5 \end{bmatrix} \bmod 26$$

: $\begin{bmatrix} 9 \\ 10 \end{bmatrix}$

$$19^{-1} \bmod 26$$

: 11 : $\frac{1}{19}$

Contoh 2.5 *Dipilih kunci enkripsi*

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

*Pesan "PAY MORE MONEY" mempunyai representasi dalam blok-3: (*15 0 24*) (*12 14 17*) (*5 12 14*) (*13 5 24*). Sebagai misal, enkripsi blok pertama adalah*

$$\begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \begin{bmatrix} 15 \\ 0 \\ 24 \end{bmatrix} \bmod 26 = \begin{bmatrix} 11 \\ 13 \\ 18 \end{bmatrix}$$

artinya "PAY" dienkripsi menjadi "LNS". Secara keseluruhan, pesan dienkripsi menjadi "LNS HDLE WMTRW". Kunci dekripsinya adalah

$$K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix},$$

sehingga "LNS" dienkripsi menjadi "PAY" dengan menggunakan operasi

$$\begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix} \begin{bmatrix} 11 \\ 13 \\ 18 \end{bmatrix} \bmod 26 = \begin{bmatrix} 15 \\ 0 \\ 24 \end{bmatrix}.$$

Pada Contoh 2.5, dari segi ukuran ruang kunci, serangan pelacakan lengkap tak-layak dilakukan. Akan tetapi dengan tipe serangan *tahu-plainteks,* kriptanalis hanya membutuhkan pengetahuan 3 pasang plainteks-siferteks: $(\mathbf{m}_1, \mathbf{c}_1)$, $(\mathbf{m}_2, \mathbf{c}_2)$, dan $(\mathbf{m}_3, \mathbf{c}_3)$ sedemikian sehingga matriks $M = [\mathbf{m}_1^T\ \mathbf{m}_2^T\ \mathbf{m}_3^T]$ mempunyai invers ($\gcd(|M|, 26) = 1$). Dalam hal ini,

$$\begin{aligned} K[\mathbf{m}_1^T\ \mathbf{m}_2^T\ \mathbf{m}_3^T] &= [\mathbf{c}_1^T\ \mathbf{c}_2^T\ \mathbf{c}_3^T] \Leftrightarrow \\ KM &= [\mathbf{c}_1^T\ \mathbf{c}_2^T\ \mathbf{c}_3^T] \Leftrightarrow \\ K &= [\mathbf{c}_1^T\ \mathbf{c}_2^T\ \mathbf{c}_3^T]M^{-1}, \end{aligned}$$

dan kunci K dengan mudah didapatkan.

2.2.2 Sandi Transposisi

Definisi 2.7 *(Sandi Transposisi) Diberikan skema enkripsi blok kunci simetrik dengan panjang blok t. Misalkan $\mathcal{K}$ adalah himpunan semua permutasi pada $\{1, 2, ..., t\}$. Untuk setiap $e \in \mathcal{K}$, didefinisikan tranformasi enkripsi*

$$\begin{aligned} E_e(\mathbf{m}) &= (m_{e(1)}, m_{e(2)}, ..., m_{e(t)}) \\ &= (c_1, c_2, ..., c_t) \\ &= \mathbf{c}, \end{aligned}$$

*dimana $\mathbf{m} = (m_1, m_2, ..., m_t) \in \mathcal{M}$, ruang pesan. Himpunan semua tranformasi tersebut, yaitu $\{E_e/e \in \mathcal{K}\}$, disebut **sandi transposisi sederhana**. Dekripsi yang terkait dengan pasangan e diberikan oleh*

$$\begin{aligned} D_d(\mathbf{c}) &= (c_{d(1)}, c_{d(2)}, ..., c_{d(t)}) \\ &= (m_1, m_2, ..., m_t) \\ &= \mathbf{m}, \end{aligned}$$

dimana $d = e^{-1}$.

15!

: 1307 674 368 000 *:* 3628 800 *:* 720

Contoh 2.6 *(transposisi sederhana) Diberikan sandi transposisi sederhana dengan $t = 6$ dan kunci dipilih*

$$e = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 6 & 4 & 1 & 3 & 5 & 2 \end{pmatrix}.$$

Misalkan pesan $\mathbf{m} = CAESAR$, maka $\mathbf{m}$ dienkripsi dengan kunci e menjadi $\mathbf{c} = RSCEAA$. Proses dekripsi menggunakan kunci pasangan

$$d = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 6 & 4 & 2 & 5 & 1 \end{pmatrix}.$$

Komposisi Sandi

Suatu cara yang cukup mudah untuk mendapatkan fungsi-fungsi yang lebih kompleks dalam suatu enkripsi adalah dibuat dari komposisi fungsi-fungsi yang lebih sederhana. Fungsi sederhana yang sukup sering digunakan dalam komposisi adalah involusi. Hal ini didasarkan pada dua fakta, yaitu: *involusi mempunyai invers dirinya sendiri* dan *komposisi dua involusi tidak harus*

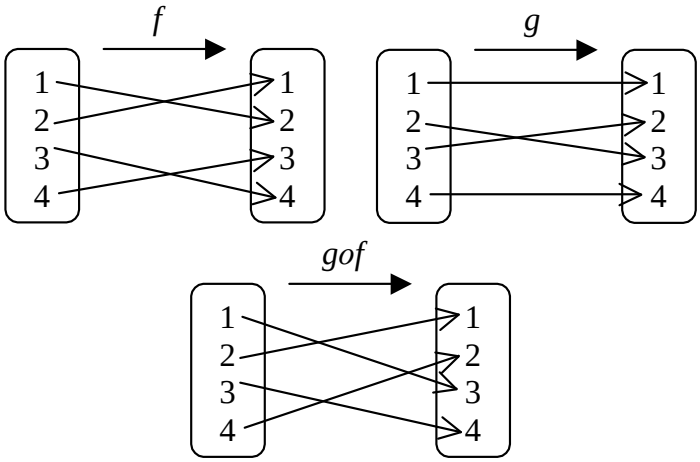
menghasilkan involusi. Misalnya, E_k adalah hasil komposisi dari t involusi: E_{k_1}, E_{k_2}, ..., dan E_{k_t}, ditulis

$$E_k = E_{k_1} E_{k_2} ... E_{k_t},$$

maka invers dari E_k dengan mudah dapat ditentukan

$$\begin{aligned} E_k^{-1} &= (E_{k_1} E_{k_2} ... E_{k_t})^{-1} \\ &= E_{k_t}^{-1} ... E_{k_2}^{-1} E_{k_1}^{-1} \\ &= E_{k_t} ... E_{k_2} E_{k_1}. \end{aligned}$$

Komposisi dua involusi tidak harus menghasilkan involusi diilutrasikan pada gambar berikut.



Sandi substitusi sederhana dan transposisi secara individual tidak memberikan tingkat keamanan yang tinggi. Akan tetapi kombinasi diantara mereka sangat mungkin menghasilkan sandi dengan tingkat keamanan yang cukup kuat. Kombinasi yang demikian disebut *sandi produk* (product cipher). Salah satu contoh sandi produk adalah *ronde* (round), yaitu komposisi dari t ($t \geq 2$) transformasi E_{k_1}, E_{k_2}, ..., dan E_{k_t}, dimana E_{k_i}, $1 \leq i \leq t$, merupakan sandi substitusi atau transposisi.

Contoh 2.7 *(Sandi produk) Diberikan $\mathcal{M} = \mathcal{C} = \mathcal{K}$ adalah himpunan yang beranggotakan semua bitstring dengan panjang 6, berarti $|\mathcal{M}| = 2^6 = 64$. Misalkan $\mathbf{m} = (m_1 m_2 m_3 m_4 m_5 m_6)$ dan didefinisikan*

$$\begin{aligned} E_{\mathbf{k}}^{(1)}(\mathbf{m}) &= \mathbf{m} \oplus \mathbf{k}, \text{ dimana } \mathbf{k} \in \mathcal{K}, \\ E^{(2)}(\mathbf{m}) &= (m_4 m_5 m_6 m_1 m_2 m_3). \end{aligned}$$

*Disini, $\oplus$ adalah operasi **exclusive-or (XOR)** yang didefinisikan:* $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$. $E_{\mathbf{k}}^{(1)}$ *merupakan sandi substitusi polialfabetik dan* $E^{(2)}$ *merupakan sandi transposisi (tidak melibatkan kunci). Produk* $E_{\mathbf{k}}^{(1)} E^{(2)}$ *adalah suatu ronde.*

Suatu substitusi dalam suatu ronde disebut menambah *konfusi* (confusion) pada proses enkripsi, sedangkan suatu transposisi dikatakan menambah *difusi* (diffusion). Konfusi dimaksudkan untuk membuat hubungan antara kunci dan siferteks sekompleks mungkin. Difusi mengakibatkan penyebaran bit dalam pesan, sehingga apabila ada *redundancy* dalam plainteks akan menyebar di dalam siferteks.

2.3 Sandi Blok Kunci Simetrik

Sebenarnya sandi substitusi dan transposisi yang diberikan pada subbab sebelumnya dapat dipandang sebagai jenis sandi blok. Pada subbab ini sandi blok akan dibahas lebih spesifik (dikhususkan pada alfabet biner) dan dibatasi hanya dengan kunci simetrik (sandi blok bisa dengan kunci publik).

2.3.1 Pendefinisan Sandi Blok

Sandi blok adalah fungsi yang memetakan plainteks blok n-bit ke siferteks blok n-bit, dalam hal ini n disebut *panjang blok*. Fungsi ini diberi parameter $K \in \mathcal{K}$ sebagai kunci k-bit. $\mathcal{K}$ adalah ruang kunci yang anggotanya bitstring k-bit dalam $\mathcal{V}_k$, dimana $\mathcal{V}_k$ adalah himpunan semua bitstring k-bit (jelas bahwa $|\mathcal{V}_k| = 2^k$). Sebagai ilustrasi

$$\begin{aligned} \mathcal{V}_2 &= \{00, 01, 10, 11\} \\ \mathcal{V}_3 &= \{000, 001, 010, 011, 100, 101, 110, 111\}. \end{aligned}$$

Diasumsikan bahwa parameter K dipilih secara random dari ruang kunci. Penggunaan blok plainteks dan siferteks dengan ukuran yang sama dimaksudkan untuk menghindari ekspansi data. Berikut ini diberikan definisi formal sandi blok.

Definisi 2.8 ***Sandi blok*** *n-bit adalah fungsi $E : \mathcal{V}_n \times \mathcal{K} \rightarrow \mathcal{V}_n$, sehingga untuk setiap kunci $K \in \mathcal{K}$, $E(P, K)$, dinotasikan $E_K(P)$ dan disebut **fungsi enkripsi** untuk K, merupakan pemetaan invertibel dari $\mathcal{V}_n$ ke $\mathcal{V}_n$. Invers dari $E_K(P)$ adalah **fungsi dekripsi** yang terkait, dinotasikan $D_K(C)$. Dalam hal ini $C = E_K(P)$ menotasikan bahwa siferteks hasil enkripsi plainteks P oleh K.*

Umumnya sandi blok memproses plainteks dengan panjang blok relatif besar (misalnya, $n \geq 64$). Sandi blok yang mempunyai ukuran blok terlalu kecil rentan terhadap serangan yang menggunakan analisis statistik. Fungsi enkripsi bersifat deterministik. Setiap pasangan blok plainteks P dan kunci K menghasilkan satu blok siferteks.

2.3.2 Mode Operasi

Sandi blok mengenkripsi plainteks dalam ukuran yang ditetapkan n-bit (biasanya digunakan $n = 64$). Untuk pesan yang melampaui n-bit harus diubah dulu menjadi blok n-bit. Suatu pendekatan yang paling sederhana adalah mempartisikan pesan menjadi blok n-bit, masing-masing dienkripsi secara terpisah.

Pada bagian ini dibahas empat metode (mode operasi) yang paling umum dipakai dalam proses enkripsi (encipherment) dan proses dekripsi (decipherment) untuk sandi blok, yaitu:

(i) mode ECB (electronic codebook),

(ii) mode CBC (cipher block chaining),

(iii) mode CFB (cipher feedback), dan

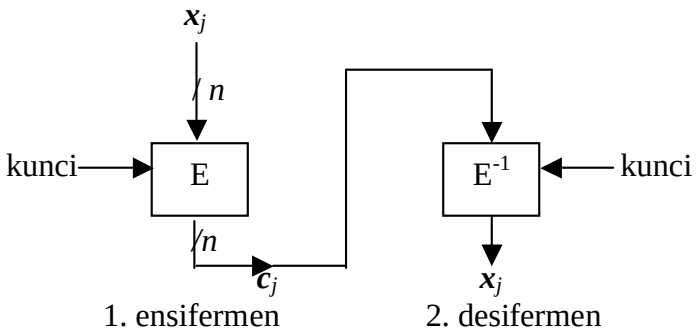
(iv) mode OFB (output feedback).

Mode EBC

Berikut adalah algoritme mode ECB, dilustrasikan pada Gambar 4.

Algoritme 2.1 *(Mode Operasi ECB)*
INPUT: $K \leftrightarrow$ *kunci k-bit;* $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t \leftrightarrow$ *blok plainteks n-bit.*
PROSES: enkripsi $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t$ *ke blok siferteks n-bit:* $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_t$;
dekripsi $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_t$ *ke* $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t$.

1. *Enkripsi: untuk* $1 \leq j \leq t$, $\mathbf{c}_j \leftarrow E_K(\mathbf{x}_j)$.
2. *Dekripsi: untuk* $1 \leq j \leq t$, $\mathbf{x}_j \leftarrow E_K^{-1}(\mathbf{c}_j)$.



Gambar 4: Skema mode ECB.

Sifat-sifat mode ECB:

1. Keidentikkan plainteks: dengan kunci yang sama, blok plainteks yang identik akan menghasilkan siferteks yang identik.

2. Ketergantungan ikatan: masing-masing blok diensifer secara bebas, satu dengan yang lainnya tidak saling tergantung.

3. Perambatan galat: terjadinya galat satu bit atau lebih pada satu blok siferteks mempengaruhi desifermen hanya pada blok yang bersangkutan, tidak berpengaruh pada blok yang lain.

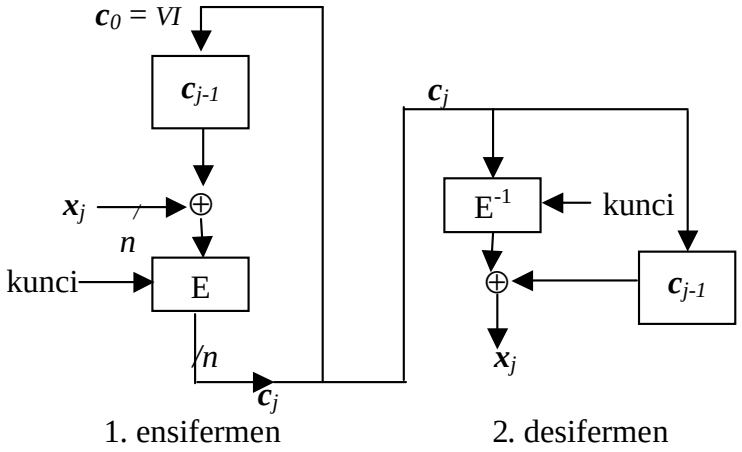
Mode CBC

Mode ini melibatkan *vektor inisial* n-bit, dinotasikan dengan VI. Berikut diberikan algoritme mode CBC dan dilustrasikan pada Gambar 5.

Algoritme 2.2 *(Mode Operasi CBC)*
INPUT: $K \leftrightarrow$ *kunci* k*-bit;* $VI \leftrightarrow$ *vektor inisial* n*-bit;* $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t \leftrightarrow$ *blok plainteks* n*-bit.*
PROSES: enkripsi $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t$ *ke blok siferteks* n*-bit:* $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_t$;
dekripsi $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_t$ *ke* $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t$.

1. *Enkripsi:* $\mathbf{c}_0 \leftarrow VI$. *Untuk* $1 \leq j \leq t$, $\mathbf{c}_j \leftarrow E_K(\mathbf{c}_{j-1} \oplus \mathbf{x}_j)$.

2. *Dekripsi:* $\mathbf{c}_0 \leftarrow VI$. *Untuk* $1 \leq j \leq t$, $\mathbf{x}_j \leftarrow \mathbf{c}_{j-1} \oplus E_K^{-1}(\mathbf{c}_j)$.



Gambar 5: Skema mode CBC.

Sifat-sifat mode CBC:

1. Keidentikkan plainteks: hasil blok siferteks akan identik apabila diensifer oleh kunci dan VI yang sama. Mengubah VI, kunci, atau blok plainteks yang pertama mengakibatkan hasil siferteks yang berbeda.

2. Ketergantungan ikatan: mekanisme berantai menyebabkan siferteks $\mathbf{c}_j$ bergantung pada $\mathbf{x}_j$ dan semua blok plainteks sebelumnya. Akibatnya, mengubah urutan blok siferteks mempengaruhi dekripsi.

3. Perambatan galat: terjadinya galat satu bit pada blok siferteks $\mathbf{c}_j$ akan mempengaruhi desifermen blok $\mathbf{c}_j$ dan $\mathbf{c}_{j+1}$ karena $\mathbf{x}_j$ bergantung pada $\mathbf{c}_j$ dan $\mathbf{c}_{j-1}$.

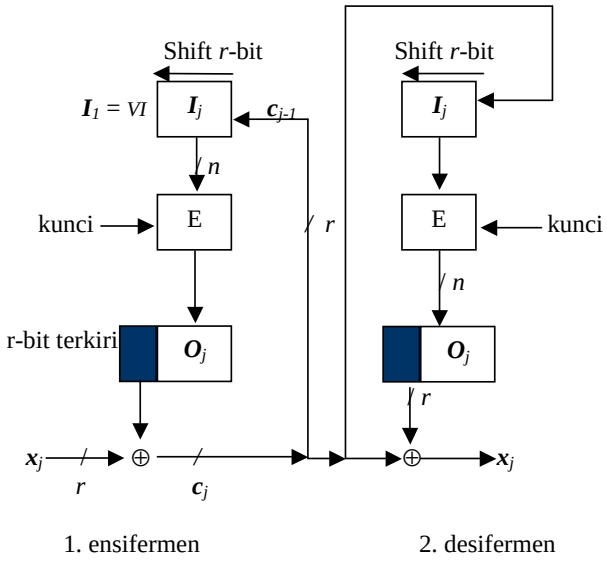
Dalam mode CBC perlu dicatat bahwa VI tidak perlu rahasia, tetapi integritasnya harus dilindungi karena modifikasi yang tidak sah menyebabkan musuh dapat meprekdiksi perubahan bit pada blok plainteks pertama hasil desifer.

Mode CFB

Dua mode sebelumnya memproses n bit plainteks sekali waktu. Beberapa aplikasi menghendaki bahwa satuan plainteks r-bit dienkripsi dan langsung ditransfer tanpa tunda, dimana ditetapkan nilai $r < n$ (sering kali $r = 1$ atau 8). Dalam hal demikian, mode CFB cukup sesuai untuk digunakan. Berikut diberikan algoritme mode CFB dan dilustrasikan pada Gambar 6.

Algoritme 2.3 *(Mode Operasi CFB-r)*
INPUT: $K \leftrightarrow$ *kunci k-bit;* $VI \leftrightarrow$ *vektor inisial n-bit;* $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_u \leftrightarrow$ *blok plainteks r-bit* $(1 \leq r \leq n)$.
PROSES: enkripsi $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_u$ *ke blok siferteks r-bit:* $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_u$;
dekripsi $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_u$ *ke* $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_u$.

1. *Ensifermen:* $\mathbf{I}_1 \leftarrow VI$ *(*$\mathbf{I}_j$ *adalah nilai input dalam suatu "shift register"). Untuk* $1 \leq j \leq u$*:*
   (a) $\mathbf{O}_j \leftarrow E_K(\mathbf{I}_j)$. *(Menghitung output sandi blok.)*
   (b) $\mathbf{t}_j \leftarrow r$*-bit terkiri dari* $\mathbf{O}_j$. *(Diasumsikan bit terkiri adalah* 1.*)*
   (c) $\mathbf{c}_j \leftarrow \mathbf{x}_j \oplus \mathbf{t}_j$. *(Mentransfer blok siferteks r-bit* $\mathbf{c}_j$.*)*
   (d) $\mathbf{I}_{j+1} \leftarrow 2^r.\mathbf{I}_j + \mathbf{c}_j \bmod 2^n$. *(Menggeser* $\mathbf{c}_j$ *ke kanan sampai akhir shift register)*
2. *Desifermen:* $\mathbf{I}_1 \leftarrow VI$ *Untuk* $1 \leq j \leq u$*:* $\mathbf{x}_j \leftarrow \mathbf{c}_j \oplus \mathbf{t}_j$.



Gambar 6: Skema mode CFB.

2.4 Sandi Alir Kunci Simetrik

Sandi alir (stream cipher) merupakan bentuk klasifikasi yang cukup penting dari skema enkripsi kunci simetrik. Sandi ini dapat dipahami sebagai sandi blok substitusi sederhana dengan panjang blok sama dengan satu. Fakta yang membuat sandi ini cukup bermanfaat adalah bahwa transformasi enkripsi dapat mengubah setiap simbol plainteks yang dienkripsi. Di dalam situasi bahwa peluang terjadinya galat pada transmisi cukup tinggi, sandi alir mempunyai keuntungan karena tidak mempunyai perambatan galat. Sandi ini juga bisa digunakan ketika data harus diproses setiap simbol sekali waktu.

Definisi 2.9 *Misalkan $\mathcal{K}$ adalah ruang kunci untuk himpunan semua transformasi enkripsi. Barisan simbol $e_1e_2e_3...$, $e_i \in \mathcal{K}$, disebut* ***alirkunci*** *(keystream).*

Definisi 2.10 *Diberikan $\mathcal{A}$ adalah alfabet dengan q simbol dan E_e adalah sandi substitusi sederhana dengan panjang blok 1. Misalkan $m_1m_2m_3...$ adalah string plainteks dan $e_1e_2e_3...$ adalah alirkunci dari $\mathcal{K}$.* ***Sandi alir*** *mentransformasikan string plainteks ke string siferteks $c_1c_2c_3...$ dengan rumus $c_i = E_{e_i}(m_i)$. Jika d_i menotasikan invers dari e_i, maka $D_{d_i}(c_i) = m_i$ mendekripsi string siferteks ke string plainteks.*

Sandi alir menerapkan transformasi enkripsi sederhana berdasarkan alirkunci yang digunakan. Alirkunci bisa dibangkitkan secara random atau dengan algoritme yang membangkitkan alirkunci dari suatu alirkunci kecil (inisial) yang disebut dengan *biji* (seed). Algoritme itu disebut *generator alirkunci* (keystream generator).

Suatu contoh sederhana yang termasuk sandi alir adalah sandi Vernam yang diberikan dalam definisi berikut.

Definisi 2.11 ***Sandi Vernam*** *adalah sandi alir yang didefinisikan pada alfabet $\mathcal{A} = \{0, 1\}$. Pesan biner $m_1m_2m_3...m_t$ dioperasikan dengan bitstring kunci $k_1k_2k_3...k_t$ untuk menghasilkan string siferteks $c_1c_2c_3...c_t$ dengan rumus*

$$c_i = m_i \oplus k_i, \quad 1 \leq i \leq t.$$

Jika string kunci dipilih secara random dan tidak pernah digunakan lagi, sandi Vernam disebut ***sistem satu-kali*** *(one-time system) atau* ***pad satu-kali*** *(one-time pad)*

Dikaitkan dengan Definisi 2.10, terlihat hanya ada dua sandi substitusi, yaitu E_0 dan E_1. Pemetaan E_0 mengubah 0 ke 0 dan 1 ke 1; sedangkan E_1 mengubah 0 ke 1 dan 1 ke 0. Apabila alirkunci memuat 0, maka E_0 mempertahankan simbol plainteks; alirkunci memuat 1 berarti E_1 mengubah simbol plainteks.

Jika string kunci digunakan lebih dari satu kali, maka ada cela untuk menyerang sistem. Hal ini didasarkan fakta: jika $c_1c_2c_3...c_t$ dan $c'_1c'_2c'_3...c'_t$ adalah dua string siferteks yang dihasilkan dengan alirkunci yang sama $k_1k_2k_3$ $...k_t$, maka

$$c_i = m_i \oplus k_i, \quad c'_i = m'_i \oplus k_i,$$

akibatnya, $c_i \oplus c'_i = m_i \oplus m'_i$.

Sandi Alir pada umumnya diklasifikasikan menjadi menjadi dua: *sandi alir selaras* (synchronous stream ciphers) dan *sandi alir selaras-diri* (self-synchronous stream ciphers).

2.4.1 Sandi Alir Selaras

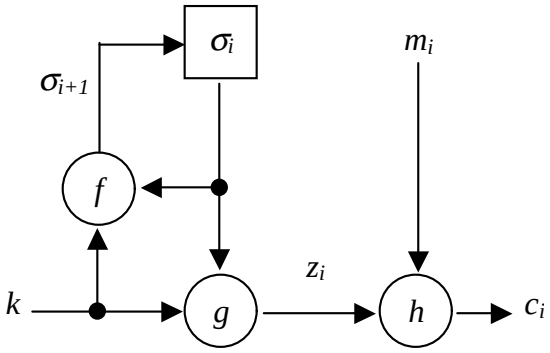
Definisi 2.12 *Sandi alir selaras adalah sandi alir yang alirkuncinya dibangkitkan secara bebas dari baik plainteks maupun siferteks.*

Proses enkripsi dari sandi alir selaras dapat dinyatakan dengan persamaan

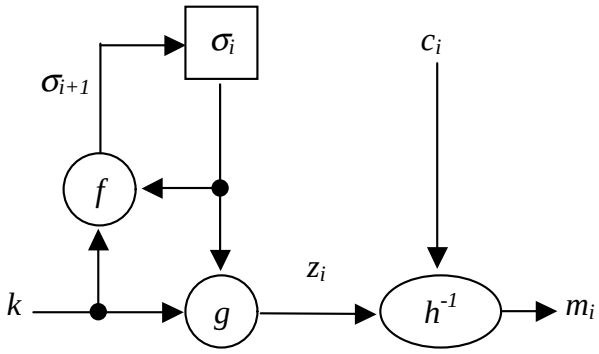
$$\begin{aligned} \sigma_{i+1} &= f(\sigma_i, k), \\ z_i &= g(\sigma_i, k), \\ c_i &= h(z_i, m_i), \end{aligned}$$

dimana σ_0 adalah *status inisial* (initial state) dan bisa ditentukan dari kunci k, f adalah *fungsi status-berikutnya,* g adalah fungsi yang memproduksi alirkunci z_i, dan h adalah *fungsi output* yang mengkombinasikan alirkunci dan plainteks m_i untuk memproduksi siferteks c_i. Proses enkripsi dan dekripsi ilustrasikan pada Gambar 7.

Enkripsi



Dekripsi

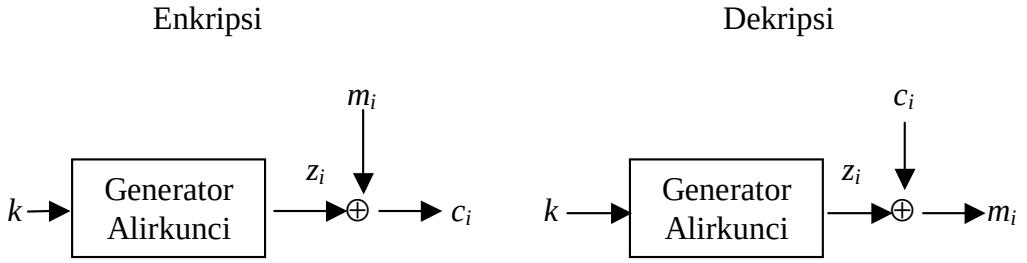


Gambar 7: Model umum sandi alir selaras.

Di dalam sandi alir selaras, pengirim dan penerima harus diselaraskan (dalam arti menggunakan kunci yang sama dan mengopersikan kunci itu pada status yang sama) untuk mendapatkan dekripsi yang benar. Jika kehilangan keselarasan akibat adanya beberapa dijit siferteks yang tersisipi atau terhapus selama transmisi, maka dekripsi akan gagal dan dapat pulihkan hanya melalui tambahan teknik untuk keselarasan ulang. Teknik-teknik keselarasan ulang meliputi inisialisasi ulang, penempatan tanda khusus pada interval reguler dalam siferteks.

Definisi 2.13 ***Sandi alir aditif biner*** *adalah sandi alir selaras yang alir-kunci, plainteks, dan siferteksnya berdijit biner, dan fungsi output h adalah fungsi XOR.*

Sandi alir aditif biner diilustrasikan pada Gambar 8.



Gambar 8: Model umum sandi alir aditif biner.

Sandi Alir Selaras-diri

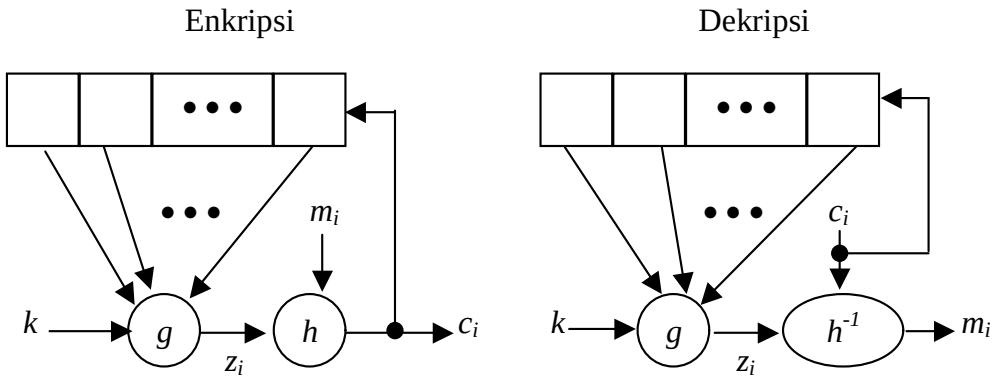
Definisi 2.14 *Sandi alir selaras-diri adalah sandi alir yang alirkuncinya dibangkitkan oleh fungsi dari kunci dan sejumlah tetap dijit siferteks sebelumnya.*

Proses enkripsi dari sandi alir selaras-diri dapat dinyatakan dengan persamaan

$$\begin{aligned} \sigma_i &= (c_{i-t}, c_{i-t+1}, ..., c_{i-1}), \\ z_i &= g(\sigma_i, k), \\ c_i &= h(z_i, m_i), \end{aligned}$$

dimana $\sigma_0 = (c_{-t}, c_{-t+1}, ..., c_{-1})$ adalah *status inisial* yang tidak rahasia, k adalah kunci, g adalah fungsi yang memproduksi alirkunci z_i, dan h adalah *fungsi output* yang mengkombinasikan alirkunci dan plainteks m_i untuk memproduksi siferteks c_i. Proses enkripsi dan dekripsi ilustrasikan pada Gambar 9.

Karena di dalam sandi alir selaras-diri transformasi dekripsi hanya bergantung kepada sejumlah tetap karakter siferteks sebelumnya, maka. seandainya ada beberapa dijit siferteks yang tersisipi atau terhapus selama tansmisi, keselarasan masih mungkin dilakukan. Sandi ini mampu menstabilkan kembali dekripsi yang benar tanpa kehilangan keselarasan dengan hanya menggunakan sejumlah tetap karakter plainteks yang tidak dapat dipulihkan.



Gambar 9: Model umum sandi alir selaras-diri.

2.4.2 Register Geser Umpanbalik

Register geser umpanbalik (feedback shift register) diklasifikasikan menjadi dua kelompok, yaitu: *register geser umpanbalik linear* dan *register geser umpanbalik tak-linear*. Register geser umpanbalik, khususnya yang linear, merupakan komponen dasar dari banyak generator alirkunci.

Register Geser Umpanbalik Linear

Sebagaimana dinyatakan di atas bahwa register geser umpanbalik yang linear telah banyak dipakai untuk generator alirkunci. Hal ini didasarkan pada beberapa alasan diantaranya:

1. sangat sesuai untuk implementasi hardware;
2. dapat memproduksi barisan simbol dengan periode yang cukup besar;
3. dapat memproduksi barisan simbol dengan sifat statistik yang baik; dan
4. strukturnya dapat dianalisis dengan cepat menggunakan teknik aljabar.

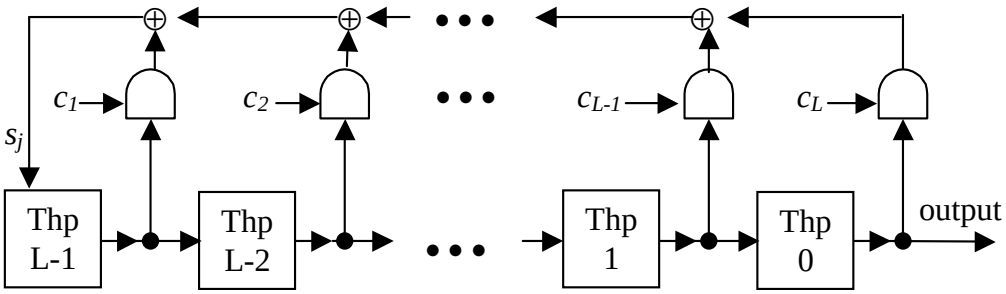
Definisi 2.15 ***Register geser umpanbalik linear*** *(LFSR) dengan panjang L terdiri dari L* ***tahap*** *(stage) yang dinomori dengan* 0, 1, 2, ..., $L-1$, *masing-masing mampu menyimpan* 1 *bit dan mempunyai* 1 *input dan* 1 *output; dan dilengkapi suatu klok (clock) yang mengontrol pergerakan data. Selama setiap satuan waktu operasi berikut dikerjakan:*

(i) isi tahap 0 *adalah output dan membentuk bagian dari* ***barisan output****;*

(ii) isi tahap i *digerakkan ke tahap* $i-1$ *untuk setiap* i, $1 \leq i \leq L-1$; *dan*

(iii) isi baru dari tahap $L-1$ *adalah* ***bit umpanbalik*** *(feedback bit)* s_j *yang dihitung dengan menambahkan sekaligus isi subhimpunan tetap dari* $\{0, 1, 2, ..., L-1\}$ *modulo* 2.

Skematik LFSR diberikan pada Gambar 10, setengah lingkaran tertutup menunjukkan operasi logika AND.



Gambar 10: Skematik LFSR dengan panjang L.

Definisi 2.16 *LFSR pada Gambar 10 dinotasikan dengan* $\langle L, C(D)\rangle$, *dimana*

$$C(D) = 1 + c_1 D + c_2 D^2 + ... + c_L D^L \in \mathbb{Z}_2[D]$$

[1] *adalah* ***polinomial penghubung*** *(connection polynomial). LFSR dikatakan non-singular jika derajat* $C(D)$ *sama dengan* L *(yaitu* $c_L = 1$*). Jika isi awal dari tahap* i *adalah* $s_i \in \{0, 1\}$ *untuk setiap* i, $0 \leq i \leq L-1$, *maka* $[s_{L-1}, ..., s_1, s_0]$ *disebut* ***status inisial*** *(initial state) dari LSFR.*

Proposition 2.1 *Jika status inisial dari LFSR dalam Gambar 10 adalah* $[s_{L-1}, ..., s_1, s_0]$, *maka barisan* $\mathbf{s} = s_0, s_1, s_2, ...$ *secara tunggal ditentukan oleh rekursi berikut*

$$s_j = (c_1 s_{j-1} + c_2 s_{j-2} + ... + c_L s_{j-L}) \mod 2 \text{ untuk } j \geq L. \tag{2.1}$$

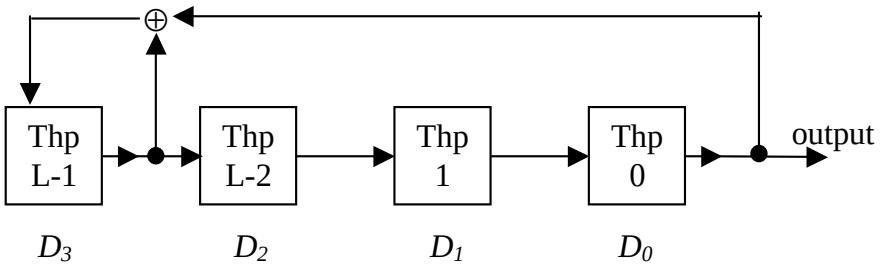
[1] $\mathbb{Z}_2[D]$ merupakan himpunan semua polinomial dalam variabel D dan koefisien biner

Contoh 2.8 *Diberikan LFSR* $\langle 4, 1 + D + D^4 \rangle$. *Jika status inisial dari LFSR tersebut adalah* $[0, 0, 0, 0]$, *maka barisan outputnya adalah barisan nol. Jika status inisial diberikan* $[0, 1, 1, 0]$, *berikut ini diberikan tabel yang menunjukkan isi tahap* D_3, D_2, D_1, D_0 *pada akhir setiap satuan waktu* t.

t	D_3	D_2	D_1	D_0
0	0	1	1	0
1	0	0	1	1
2	1	0	0	1
3	0	1	0	0
4	0	0	1	0
5	0	0	0	1
6	1	0	0	0
7	1	1	0	0

t	D_3	D_2	D_1	D_0
8	1	1	1	0
9	1	1	1	1
10	0	1	1	1
11	1	0	1	1
12	0	1	0	1
13	1	0	1	0
14	1	1	0	1
15	0	1	1	0

Barisan outputnya adalah $\mathbf{s} = 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, ...$ *mempunyai periode* 15. *Skematik LFSR ini diberikan pada Gambar 11.*



Gambar 11: Skematik LFSR $\langle 4, 1 + D + D^4 \rangle$.

Proposition 2.2 *Untuk semua status inisial yang mungkin, setiap barisan output dari LFSR* $\langle L, C(D) \rangle$ *adalah periodik jika dan hanya jika polinomial penghubung* $C(D)$ *mempunyai derajat* L.

Proposition 2.3 *Misalkan* $C(D) \in \mathbb{Z}_2[D]$ *adalah polinomial berderajat* L.

(i) Jika $C(D)$ *takterfaktorkan (irreducible) atas* $\mathbb{Z}_2$, *maka masing-masing dari* $2^L - 1$ *status inisial tak-nol dari non-singular LFSR* $\langle L, C(D) \rangle$ *memproduksi barisan output dengan periode intejer positif terkecil* N *sehingga* $C(D)$ *membagi* $1 + D^N$ *di dalam* $\mathbb{Z}_2[D]$. *Catatan bahwa dalam hal ini* N *selalu merupakan pembagi dari* $2^L - 1$

(ii) Jika $C(D)$ adalah polinomial primitif, maka masing-masing dari $2^L - 1$ status inisial tak-nol dari non-singular LFSR $\langle L, C(D)\rangle$ memproduksi barisan output dengan periode yang mungkin maksimum $2^L - 1$.

Definisi 2.17 *Jika $C(D) \in \mathbb{Z}_2[D]$ adalah polinomial berderajat L, maka $\langle L, C(D)\rangle$ disebut LFSR dengan* ***panjang maksimum****. Output LFSR dengan panjang maksimum dan status insial tak-nol disebut* ***barisan-****m.*

Register Geser Umpanbalik Tak-linear

Suatu fungsi yang mengambil n input biner dan satu menghasilkan output biner disebut *fungsi Boolean*. Sebelum pengertian register geser umpanbalik tak-linear, berikut ini diberikan definisi *register geser umpanbalik* (feedback shift register) umum.

Definisi 2.18 ***Register geser umpanbalik*** *(FSR) dengan panjang L terbentuk dari L tahap yang diberi nomor 0, 1, 2, ..., $L-1$, dan masing-masing tahap mampu menyimpan satu bit, mempunyai satu bit input, mempunyai satu bit output, dan dilengkapi suatu klok yang mengatur pergerakan data. Selama setiap satuan waktu operasi berikut dikerjakan:*

(i) isi tahap 0 adalah output dan membentuk bagian dari ***barisan output****;*

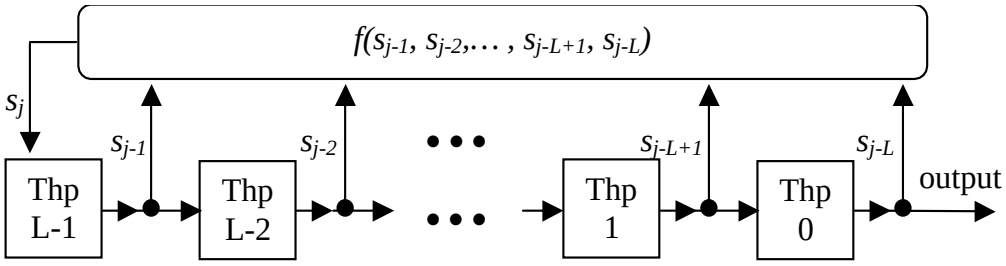
(ii) isi tahap i digerakkan ke tahap $i-1$ untuk setiap i, $1 \leq i \leq L-1$; dan

(iii) isi baru dari tahap $L-1$ adalah ***bit umpanbalik*** *(feedback bit)*

$$s_j = f(s_{j-1}, s_{j-2}, ..., s_{j-L})$$

dimana f disebut ***fungsi umpanbalik*** *yang merupakan fungsi Boolean dan s_{j-i} adalah isi tahap sebelumnya, yaitu $L-i$ untuk $1 \leq i \leq L$.*

Jika isi awal dari tahap i adalah $s_i \in \{0, 1\}$ untuk setiap i, $0 \leq i \leq L-1$, maka $[s_{L-1}, ..., s_1, s_0]$ disebut *status inisial* (initial state) dari SFR. Skematik FSR diberikan pada Gambar 12.



Gambar 12: Skematik FSR dengan panjang L.

Catatan bahwa fika fungsi umpanbalik f adalah linear (lihat Persamaan 2.1), maka FSR menjadi LFSR. Sebaliknya jika f tak-linear, FSR disebut *tak-linear*.

Proposition 2.4 *Jika status inisial dari FSR dalam Gambar 12 adalah*

$$[s_{L-1}, ..., s_1, s_0],$$

maka barisan $\mathbf{s} = s_0, s_1, s_2, ...$ *secara tunggal ditentukan oleh rekursi berikut*

$$s_j = f(s_{j-1}, s_{j-2}, ..., s_{j-L}) \text{ untuk } j \geq L.$$

Definisi 2.19 *FSR dikatakan* ***non-singular*** *jika dan hanya jika setiap barisan outputnya adalah periodik.*

Proposition 2.5 *FSR dengan fungsi umpanbalik.* $f(s_{j-1}, s_{j-2}, ..., s_{j-L})$ *adalah* ***non-singular*** *jika dan hanya jika* f *mempunyai bentuk*

$$f = s_{j-L} \oplus g(s_{j-1}, s_{j-2}, ..., s_{j-L+1})$$

untuk suatu fungsi Boolean g.

Definisi 2.20 *Untuk sembarang status inisial, jika periode barisan output dari suatu FSR non-singular dengan panjang* L *adalah* 2^L, *maka FSR itu disebut* ***FRS de Bruijn****, dan barisan outputnya disebut* ***barisan de bruijn****.*

Contoh 2.9 *Diberikan FSR de Bruijn dengan panjang* 3 *mempunyai fungsi umpanbalik tak-linear*

$$f(x_1, x_2, x_3) = 1 \oplus x_2 \oplus x_3 \oplus x_1 x_2.$$

Tabel berikut menunjukkan isi dari tiga tahap FSR tersebut pada akhir setiap satuan waktu t apabila diisikan status inisial $[0, 0, 0]$.

t	*Tahap 2*	*Tahap 1*	*Tahap 0*
0	0	0	0
1	1	0	0
2	1	1	0
3	1	1	1

t	*Tahap 2*	*Tahap 1*	*Tahap 0*
4	0	1	1
5	1	0	1
6	0	1	0
7	0	0	1

.

Barisan output de Bruijn mempunyai cycle: 0, 0, 0, 1, 1, 1, 0, 1.

$$2^{52} - 1$$

2.4.3 Sandi Alir RC4

Sandi alir RC4 merupakan suatu contoh sandi alir kontemporer. RC4 didisain oleh Ron Rivest pada tahun 1987 untuk keamanan sandi blok kunci publik RSA. Sandi ini mempunyai ukuran kunci yang bersifat variabel dengan operasi berorientasi byte. Algoritmenya didasarkan pada penggunaan permutasi random. Analisis menunjukkan bahwa periode dari sandi ini lebih dari 10^{100}. Delapan sampai enam belas operasi mesin diperlukan untuk keluaran per bytenya, dan diharapkan sangat cepat dalam implementasi software.

Ongoing Work!!!

Chapter 3

Algoritme Sandi Blok Kunci Simetrik

Pada bab pendahuluan telah disinggung bahwa prinsip sandi blok kontemporer untuk kunci simetrik diawali dari terciptanya *Data Encryption Standart* (DES). Sejak kekuatan DES (terutama pada panjang kuncinya) mulai diragukan, terciptalah beberapa penggantinya seperti *Advanced Encryption Standard* (AES) dan *Tripel DES*. Disamping itu beberapa lagi yang masuk dalam kategori sandi blok kontemporer untuk kunci simetrik diantaranya: *Blowfish, LUCIFER, FEAL, NewDES* dan *RC5*. Namun demikian dalam tulisan ini hanya akan dibahas Algoritme DES dengan alasan DES merupakan landasan bagi prinsip sandi blok kontemporer yang lain.

3.1 Algoritme DES

Data Encryption Standart (DES) merupakan sandi blok dengan panjang blok 64 bit dan panjang kunci efektif 56 bit. Di dalam operasinya, panjang kunci dibuat 64 bit dengan menambahkan 8 bit *paritas* ditempatkan pada posisi ke- 8, 16, 24, 32, 40, 48, 56, 64. Struktur algoritme DES menggunakan dua kosep umum: *sandi produk* dan *sandi Feistel*.

Sandi produk mengkombinasikan dua atau lebih transformasi dengan maksud memperoleh sandi yang lebih aman dibandingkan komponen-komponen penyusunnya. Komponen penyusun sandi produk berupa:

1. *transposisi,* dengan operasi permutasi,
2. *substitusi,* dengan operasi $\oplus$ (XOR).

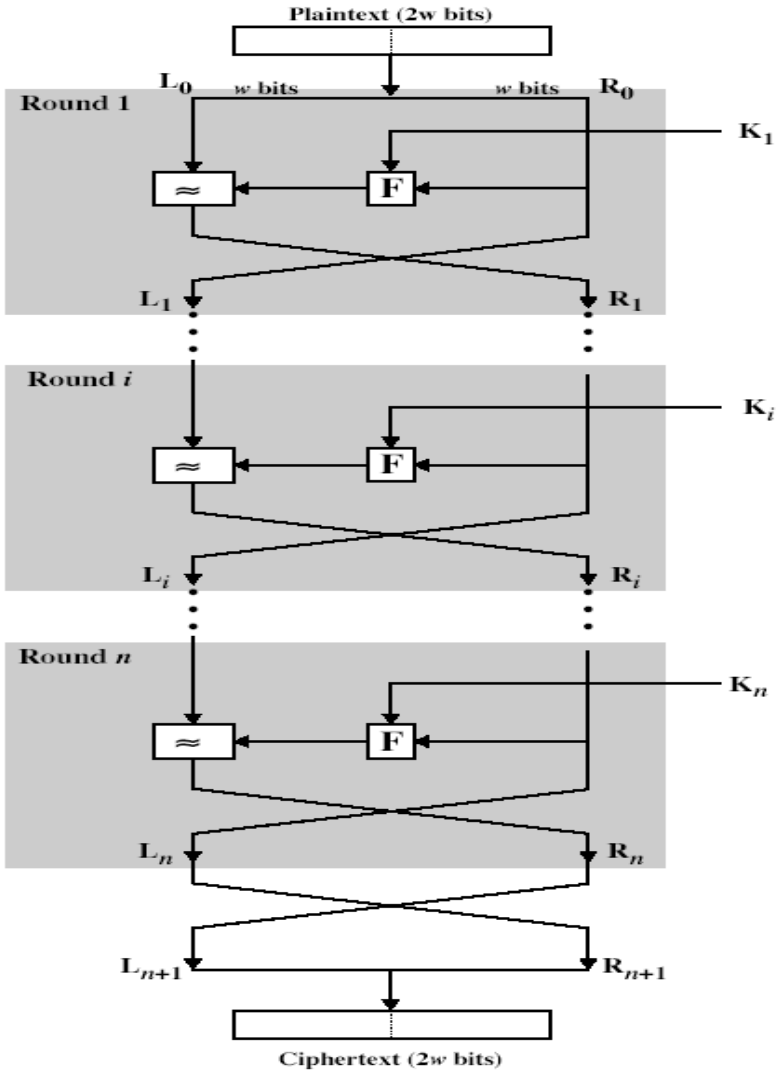
3. *transformasi linear,* dan

4. *operasi aritmatik modular.*

Sandi Feistel merupakan sandi teriterasi yang mentransformasikan blok plainteks $[L_0, R_0]$ berukuran $2w$ bit dengan masing-masing L_0 dan R_0 berukuran w bit ke blok siferteks $[L_n, R_n]$ melalui proses sebanyak n-ronde dimana $n \geq 1$. Untuk $1 \leq i \leq n$, ronde ke-i memetakan $[L_{i-1}, R_{i-1}] \xrightarrow{K_i} [L_i, R_i]$ sebagai

$$L_i = R_{i-1} \text{ dan } R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

dengan melibatkan *subkunci* K_i yang diturunkan dari kunci sandi K (Perhatikan Gambar 3.1).



Gambar 3.1: Jaringan Feitel

3.1.1 Algoritme Derivasi Sub-kunci

Kunci asli 56 bit diperluas menjadi 64 bit dengan menambahkan bit paritas pada posisi ke: 8, 16, 24, 32, 40, 48, 56, 64. Hasilnya akan menjadi intput algoritme berikut untuk mendapatkan output 16 subkunci masing-masing berukuran 48 bit.

Algoritme 3.1 ***(Derivasi Sub-kunci DES)***

INPUT: Kunci 64 *bit (termasuk bit paritas):* $K = k_1 k_2 \cdots k_{64}$. *(Lihat Tabel 3.1).*

OUTPUT: 16 *Sub-kunci masing-masing berukuran* 48 *bit:* $K_1, K_2, \cdots, K_{16}$

1	2	3	4	5	6	7	**8**
9	10	11	12	13	14	15	**16**
17	18	19	20	21	22	23	**24**
25	26	27	28	29	30	31	**32**
33	34	35	36	37	38	39	**40**
41	42	43	44	45	46	47	**48**
49	50	51	52	53	54	55	**56**
57	58	59	60	61	62	63	**64**

Tabel 3.1: *Input kunci.*

1. *Definisikan* v_i, *untuk* $1 \leq i \leq 16$, *sebagai:* $v_i = 1$ *jika* $i \in \{1, 2, 9, 16\}$ *dan* $v_i = 2$ *jika yang lainnya.*

2. $T \longleftarrow PC1(K)$. *Nyatakan* T *sebagai belahan* 28 *bit* (C_0, D_0) *dengan menggunakan Tabel 3.2 untuk Permutasi PC1, diperoleh* $C_0 = k_{57} k_{49} \cdots k_{36}$ *dan* $D_0 = k_{63} k_{55} \cdots k_4$.

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Tabel 3.2: *Pemilihan Permutasi Satu (PC-1)*

3. *Untuk* $i = 1$ *s.d.* 16, *hitung* K_i *sebagai:*

   (a) $C_i \longleftarrow (C_{i-1} \hookleftarrow v_i)$. *Disisni* " $\hookleftarrow$ " *menotasikan "geser kiri putar melingkar".*

   (b) $D_i \longleftarrow (D_{i-1} \hookleftarrow v_i)$

   (c) $K_i \longleftarrow PC2(C_i, D_i)$. *Gunakan Tabel 3.3 untuk Permutasi PC2.*

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	32

Tabel 3.3: *Pemilihan Permutasi Dua (PC-2)*

3.1.2 Algoritme Enkripsi

Algoritme 3.2 ***(Enkripsi DES)***

INPUT: Plainteks 64 *bit:* $\mathbf{m} = m_1 m_2 \cdots m_{64}$ *dan sub-kunci* 48 *bit* K_i *untuk* $1 \leq i \leq 16$.

OUTPUT: Siferteks 64 *bit* $\mathbf{c} = c_1 c_2 \cdots c_{64}$

1. $(L_0, R_0) \longleftarrow IP(m_1 m_2 \cdots m_{64})$. *Gunakan Tabel 3.4. untuk Permutasi IP, diperoleh* $L_0 = m_{58} m_{50} \cdots m_8$ *dan* $R_0 = m_{57} m_{49} \cdots m_7$.

$IP =$

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

dan $IP^{-1} =$

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Tabel ***3.4:*** *Permutasi Inisial* (IP) *dan Inversnya* (IP^{-1})

2. ***Penggunaan Sandi Fiestel*** 16 *ronde dengan input* (L_0, R_0), *output* (L_{16}, R_{16}), *dan melibatkan subkunci* K_i, $1 \leq i \leq 16$, *melalui fungsi*

$$F(R_{i-1}, K_i) = P[S\{E(R_{i-1}) \oplus K_i\}]$$

Untuk $i = 1$ *s.d.* 16, *hitung* L_i *dan* R_i *sebagai berikut:*

(a) $T \longleftarrow E(R_{i-1})$. *Ekspansi* $R_i = r_1 r_2 \cdots r_{32}$, *gunakan Tabel 3.5. untuk transformasi Ekspansi E, diperoleh* $T = r_{32} r_1 r_2 \cdots r_{32} r_1$ *berukuran* 48 *bit.*

32	1	2	3	4	**5**
4	5	6	7	8	**9**
8	9	10	11	12	**13**
12	13	14	15	16	**17**
16	17	18	19	20	**21**
20	21	22	23	24	**25**
24	25	26	27	28	**29**
28	29	30	31	32	**1**

Tabel 3.5: *Permutasi Ekspansi (E)*

(b) $T' \longleftarrow T \oplus K_i$. *Nyatakan* T' *sebagai* 8 *bitstring masing-masing berukuran* 6 *bit:* $T' = (B_1, B_2, \cdots, B_8)$.

(c) $T'' \longleftarrow (S_1(B_1), S_2(B_2), \cdots, S_8(B_8))$. *Untuk* $i = 1$ *s.d.* 8, S_i *memetakan* $B_i = b_1b_2b_3b_4b_5b_6$ *menjadi string* 4 *bit* $A_i = a_1a_2a_3a_4$ *dengan proses:*

i. Hitung b *sebagai representasi desimal dari string* b_1b_6.

ii. Hitung k *sebagai representasi desimal dari string* $b_2b_3b_4b_5$.

iii. Tentukan s *sebagai isian (entry) baris ke-*b *dan kolom ke-*k *dari* S_i *pada Tabel 3.6 (Definisi S-Boxes DES).*

iv. $A_i = a_1a_2a_3a_4$ *adalah representasi biner dari* s.

(d) $T''' \longleftarrow P(T'')$. *Gunakan Tabel 3.7 untuk Permutasi P, mengubah* $T'' = t_1t_2 \cdots t_{32}$ *menjadi* $T''' = t_{16}t_7 \cdots t_{25}$. *Sampai pada langkah ini,* T''' *adalah* $F(R_{i-1}, K_i)$ *dalam struktur Feistel.*

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Tabel 3.7: *Fungsi Permutasi (P)*

(e) $(L_i, R_i) \longleftarrow (R_{i-1}, L_{i-1} \oplus T''')$.

3. $h_1h_2 \cdots h_{64} \longleftarrow (R_{16}, L_{16})$. *Perhatikan bahwa* L_{16} *dan* R_{16} *tukar posisi.*

4. $\mathbf{c} = c_1c_2 \cdots c_{64} \longleftarrow IP^{-1}(h_1h_2 \cdots h_{64})$. *Gunakan Tabel 3.4 untuk Permutasi* IP^{-1}, *diperoleh* $\mathbf{c} = h_{40}h_8 \cdots h_{25}$.

3.1.3 Algoritme Dekripsi

Algoritme 3.3 ***(Dekripsi DES)***

INPUT: Siferteks 64 *bit:* $\mathbf{c} = c_1c_2 \cdots c_{64}$; *sub-kunci* 48 *bit* K_i *untuk* $1 \leq i \leq 16$.

OUTPUT: Plainteks 64 *bit* $\mathbf{m} = m_1m_2 \cdots m_{64}$.

1. ***Urutan subkunci dibalik:*** *Untuk* $i = 1$ *s.d.* 16*:*

   $K_i \longleftarrow K_{16-i+1}$

2. ***ImplementasikanAlgorime Enkripsi*** *dengan*

   INPUT: Siferteks 64 *bit:* $\mathbf{c} = c_1c_2 \cdots c_{64}$; *dan subkunci* 48 *bit* K_i *untuk* $1 \leq i \leq 16$.

Penjelasan Algoritme Dekripsi

Algoritme Dekripsi DES merupakan Algoritme Enkripsi DES yang inputnya $\mathbf{c} = c_1c_2\cdots c_{64}$ dan mengubah urutan subkuncinya dalam keadaan terbalik. Perhatikan langkah ke-2 Algoritme Dekripsi yang pada dasarnya adalah langkah-langkah enkripsi untuk mengubah $\mathbf{c}$ menjadi $\mathbf{m}$:

$$\begin{aligned}(L_0, R_0) &\longleftarrow IP(c_1c_2\cdots c_{64}) \Leftrightarrow\\ (L_0, R_0) &\longleftarrow IP(IP^{-1}(h_1h_2\cdots h_{64})) \Leftrightarrow\\ (L_0, R_0) &\longleftarrow (h_1h_2\cdots h_{64}) = (R_{16}, L_{16})\end{aligned}$$

Ini berarti, (R_{16}, L_{16}) akan menjadi input sandi Feistel. Pada ronde pertama, (R_{16}, L_{16}) diubah menjadi (R_{15}, L_{15}) dengan subkunci K_{16} melalui penjelasan hitungan sebagai berikut:

$$\begin{aligned}& (R_{16}, L_{16}) \xrightarrow{K_{16}} (L_{16}, R_{16} \oplus F(L_{16}, K_{16})), \text{ dimana}\\ (L_{16}, R_{16} \oplus F(L_{16}, K_{16})) &= (R_{15}, [L_{15} \oplus F(R_{15}, K_{16})] \oplus F(R_{15}, K_{16}))\\ &= (R_{15}, L_{15} \oplus [F(R_{15}, K_{16}) \oplus F(R_{15}, K_{16})])\\ &= (R_{15}, L_{15} \oplus \mathbf{0})\\ &= (R_{15}, L_{15}).\end{aligned}$$

Dengan penjelasan yang sama, diperoleh

$$(R_{16}, L_{16}) \xrightarrow{K_{16}} (R_{15}, L_{15}) \xrightarrow{K_{15}} (R_{14}, L_{14}) \xrightarrow{K_{14}} \cdots \xrightarrow{K_1} (R_0, L_0).$$

Selanjutnya setelah keluar dari sandi Feistel, (R_0, L_0) ditukar posisinya menjadi (L_0, R_0). Akhirnya,

$$\begin{aligned}IP^{-1}((L_0, R_0)) &= IP^{-1}(IP(m_1m_2\cdots m_{64}))\\ &= (m_1m_2\cdots m_{64}).\end{aligned}$$

3.2 Keamanan DES

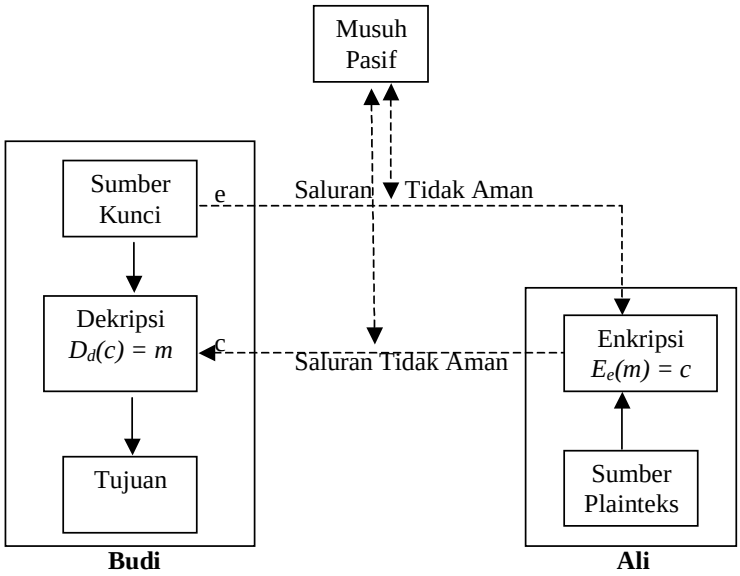
Ongoing Work!!!

Chapter 4

Enkripsi Kunci Publik

4.1 Konsep Umum Enkripsi Kunci Publik

Misalkan $\{E_e/e \in \mathcal{K}\}$ adalah himpunan semua transformasi enkripsi, dan $\{D_d/d \in \mathcal{K}\}$ adalah himpunan semua transformasi dekripsi yang terkait, dimana $\mathcal{K}$ adalah ruang kunci. Pasangan transformasi (E_e, D_d) mempunyai sifat bahwa diketahui E_e dan sembarang $c \in \mathcal{C}$, secara perhitungan tak-layak dapat menentukan $m \in \mathcal{M}$ sehingga $E_e(m) = c$. Sifat ini mengakibatkan bahwa secara perhitungan tak-layak menentukan kunci dekripsi d apabila diketahui kunci enkripsi e. Dalam bahasan enkripsi kunci publik, diasumsikan bahwa transformasi enkripsi E_e adalah *fungsi satu-arah pintu jebakan,* sehingga kunci dekripsi d dapat dihitung dari e yang ditambahkan informasi ekstra.



Gambar 13: Teknik enkripsi menggunakan kunci-publik.

Sebagai ilustrasi bagaimana teknik enkripsi kunci publik bekerja, perhatikan komunikasi antara Ali dan Budi yang diskemakan pada Gambar 13. Budi memilih pasangan kunci (e, d). Budi mengirimkan kunci enkripsi e (dalam hal ini e disebut *kunci publik*) kepada Ali melalui sembarang saluran (secara umum diasumsikan saluran tak-aman), tetapi Budi tetap merahasiakan dan mengamankan kunci dekripsi d (dalam hal ini d disebut *kunci-pribadi*). Giliran tugas Ali mengenkripsi pesan yang akan dikirim m menjadi c dengan menggunakan kunci e yang diperoleh dari Budi, berarti $c = E_e(m)$. Kemudian Ali mengirim c ke Budi, dan Budi mendapatkan m dengan menggunakan kunci d miliknya, artinya

$$m = D_d(c) = D_d(E_e(m)).$$

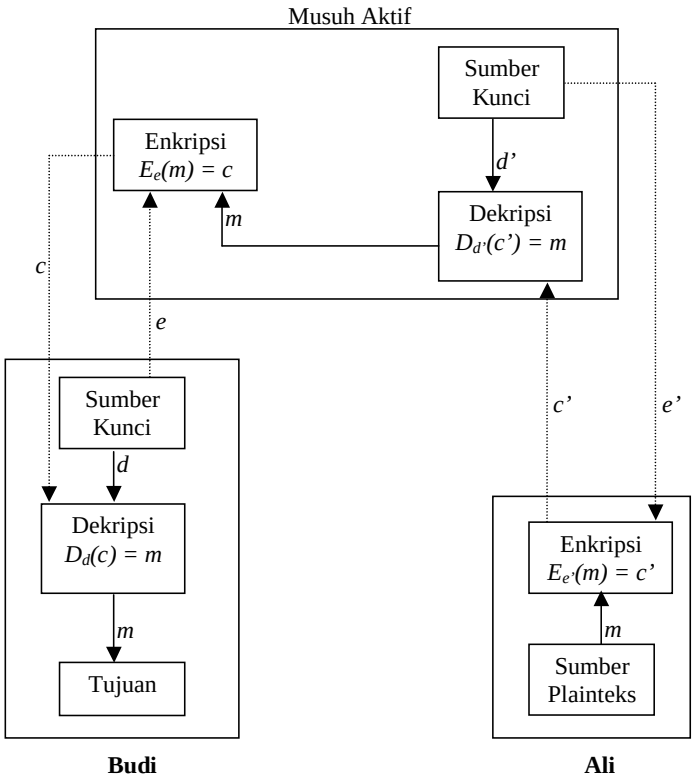
Secara formal ilustrasi tersebut diungkapkan dalam definisi berikut.

Definisi 4.1 *Diberikan skema enkripsi terdiri dari himpunan transformasi enkripsi dan dekripsi* $\{E_e/e \in \mathcal{K}\}$ *dan* $\{D_d/d \in \mathcal{K}\}$. *Metode enkripsi disebut dengan* ***skema enkripsi kunci publik*** *jika untuk setiap pasangan kunci* (e, d), *satu kunci* e *dibuat tersedia untuk umum (publik) dan kunci pasangannya* d *dibuat untuk pribadi dan dijaga kerahasiaannya. Skema tersebut dikatakan* ***aman****, jika secara perhitungan tak-layak menentukan* d *dari* e.

Beberapa contoh skema enkripsi kunci publik diantaranya: RSA, Rabin, ElGamal, generalisasi ElGamal, McEliece, Merkle-Hellman Knapsack, Chor-Rivest knapsack, Goldwasser-Micali probabilistic, dan Blum-Goldwasser probabilistic.

4.1.1 Sistem Kunci Publik Membutuhkan Autentikasi

Tampaknya kriptografi kunci publik merupakan sistem yang ideal, dalam arti tidak memerlukan saluran aman dalam pengiriman kunci pribadi. Keadaan ini mengakibatkan dua entitas bisa berkomunikasi melalui saluran tak-aman tanpa harus bertemu untuk saling menukar kunci. Ternyata apabila dicermati sistem ideal yang dimaksud tidak menjadi kenyataan. Hal ini diilustrasikan pada Gambar 14 yang menujukkan bahwa Musuh Aktif dapat mengalahkan sistem yang bersangkutan.



Gambar 14: Serangan peniruan pada komunikasi dua partai.

Sesuai dengan skenario Budi mengirim kunci publik e ke Ali melalui saluran tak-aman. Musuh Aktif menahan kunci e dan membuat pasangan kunci tiruan (e', d'), kemudian mengganti e dengan e' untuk diteruskan ke Ali. Ali mengenkripsi plainteks m dengan kunci e' menjadi siferteks c' dan dikirim ke Budi sesuai skenario melalui saluran tak-aman. Musuh Aktif mendapatkan c' dan didekripsi dengan kunci pribadi d' menjadi m. Dari sini Musuh Aktif telah mengetahui m dan selanjutnya dienkripsi dengan kunci publik e menjadi siferteks c untuk dikirim ke Budi. Akhirnya, Budi mendekripsi c dengan kunci pribadi d menjadi m sesuai skenario, sehingga Budi tidak menyadari telah terjadinya peniruan pada saat transmisi.

4.2 Skema Kunci Publik

Nama RSA diambil dari nama penemunya: R. Rivest, A. Shamir, dan Adleman. Algoritme ini memberikan tujuan kerahasiaan dan penandaan dijitel. Keamanannya bertumpu kepada kompleksitas problem *faktorisasi intejer*. Untuk itu sebelum pendeskripsian algortme RSA, berikut ini diberikan landasan teori bilangan.

4.2.1 Dasar-dasar Teori Bilangan

Himpunan semua bingan bulat dinotasikan dengan $\mathbb{Z}$. Fakta bahwa walaupun $\mathbb{Z}$ tidak tertutup terhadap pembagian, namun ada beberapa intejer yang dapat dibagi oleh intejer yang lain.

$$\sqrt{2}$$

: 1. 414 2

Definisi 4.2 *Misalkan $a, b \in \mathbb{Z}$, dan $b \neq 0$. Kita sebut b **membagi** a, ditulis $b \mid a$, jika ada intejer n sehingga $a = bn$. Dalam hal ini b disebut juga **pembagi/faktor** dari a atau a disebut **kelipatan** dari b. Dalam hal a tidak membagi b dinotasikan dengan $a \nmid b$.*

Teorema 4.1 *(Sifat-sifat pembagian) Untuk semua $a, b, c \in \mathbb{Z}$.*

a) $1 \mid a$ *dan* $a \mid 0$.

b) $[(a \mid b) \wedge (b \mid a)] \Rightarrow a = \pm b$.

c) $[(a \mid b) \wedge (b \mid c)] \Rightarrow a \mid c$.

d) $[(a \mid b) \wedge (a \mid c)] \Rightarrow [(\forall x, y \in \mathbb{Z}), a \mid (bx + cy)]$.

e) $(x = y + z) \wedge ((a \mid x) \wedge (a \mid y)) \Rightarrow a \mid y$.
f) $(\forall x, y \in \mathbb{Z}^+)\ (a \mid b) \Rightarrow a \leq b$.

Teorema 4.2 *(algoritme Pembagian) Jika $a, b \in \mathbb{Z}$ dengan $b > 0$, maka ada tepat satu $q, r \in \mathbb{Z}$ sehingga*

$$a = qb + r, \quad 0 \leq r < b.$$

*Dalam hal ini a disebut **yang dibagi**, b adalah **pembagi**, q adalah **hasil bagi**, dan r adalah **sisa pembagian**. Selanjutnya sisa pembagian dinotasikan dengan* **a *mod* b** *dan hasil bagi dinotasikan* **a *div* b**.

Fakta bahwa misalkan $a, b \in \mathbb{Z}$, $b \neq 0$, maka

$$\begin{aligned} a \text{ div } b &= \lfloor \frac{a}{b} \rfloor, \text{ dan} \\ a \bmod b &= a - b.\lfloor \frac{a}{b} \rfloor, \end{aligned}$$

notasi $\lfloor x \rfloor$ mengartikan bilangan bulat terbesar yang $\leq x$.

Definisi 4.3 *Untuk $a, b \in \mathbb{Z}$, suatu intejer positif c dikatakan **pembagi bersama** dari a dan b jika $c \mid a$ dan $c \mid b$.*

Definisi 4.4 *Misalkan $a, b \in \mathbb{Z}$, dengan a dan b tidak semuanya nol. Intejer $c \in \mathbb{Z}^+$ disebut **pembagi bersama terbesar** dari a dan b jika*

a) *c adalah pembagi bersama dari a dan b, dan*

b) *untuk setiap pembagi bersama d dari a dan b, maka $d \mid c$.*

Pembagi bersama dari a dan b dinitasikan dengan $\gcd(a, b)$.

$$\gcd(24, 18)$$

: 6

Teorema 4.3 *Untuk setiap $a, b \in \mathbb{Z}^+$, ada tepat satu $c \in \mathbb{Z}^+$ sehingga $c = \gcd(a, b)$. Selanjutnya ada $x, y \in \mathbb{Z}$ sehingga $c = xa + yb$ (c adalah suatu kombinasi linear dari a dan b).*

Sifat-sifat dasar dari pembagi bersama terbersar dapat dirinci sebagai berikut. Misalnya $c = \gcd(a, b)$, maka:

1. c adalah intejer positif terkecil dari himpunan $\{xa + yb/x, y \in \mathbb{Z}\}$.
2. Jika $d = sa + tb$ untuk suatu $x, y \in \mathbb{Z}$, maka $c \mid d$.
3. $\gcd(a, b) = \gcd(-a, b) = \gcd(a, -b) = \gcd(-a, -b) = \gcd(b, a)$.
4. $\gcd(a, 0) = |a|$ dan $\gcd(0, 0)$ tak terdefinisikan.
5. $c = \gcd(a, b) \Rightarrow \gcd\left(\frac{a}{c}, \frac{b}{c}\right) = 1$.

Intejer a dan b disebut **prima relatif** jika $\gcd(a, b) = 1$, berdasarkan Teorema 4.3 ada $x, y \in \mathbb{Z}$ sehingga $xa + yb = 1$.

Contoh 4.1 *Karena* $\gcd(42, 70) = 14$, *maka ada* $x, y \in \mathbb{Z}$, *sehingga*

$$42x + 70y = 14 \Leftrightarrow 3x + 5y = 1.$$

Mudah diperiksa bahwa $x = 2$ *dan* $y = -1$ *adalah solusinya. Kemudian untuk* $k \in \mathbb{Z}$,

$$3(2 - 5k) + 5(-1 + 3k) = 1,$$

juga

$$42(2 - 5k) + 70(-1 + 3k) = 14.$$

Jadi solusi untuk x *dan* y *tidak tunggal.*

Teorema 4.4 *(algoritme Euclid) Jika* $a, b \in \mathbb{Z}^+$, *maka dengan algoritme pembagian berlaku langkah-langkah berikut ini:*

Langkah ke-1	$a = q_1 b + r_1$	$0 < r_1 < b$
Langkah ke-2	$b = q_2 r_1 + r_2$	$0 < r_2 < r_1$
Langkah ke-3	$r_1 = q_3 r_2 + r_3$	$0 < r_3 < r_2$
$\vdots$	$\vdots$	$\vdots$
Langkah ke-(i+2)	$r_i = q_{i+2} r_{i+1} + r_{i+2}$	$0 < r_{i+2} < r_{i+1}$
$\vdots$	$\vdots$	$\vdots$
Langkah ke-k	$r_{k-2} = q_k r_{k-1} + r_k$	$0 < r_k < r_{k-1}$
Langkah ke-(k+2)	$r_{k-1} = q_{k+1} r_k$.	

Maka r_k,*sisa tak-nol terakhir, adalah* $\gcd(a, b)$.

$$\gcd(24, 18)$$

Contoh 4.2 *Carilah* $\gcd(250, 111)$, *nyatakan hasilnya sebagai kombinasi linear dari* 250 *dan* 111.

Langkah ke-1	$250 = 2(111) + 28$	$0 < 28 < 111$
Langkah ke-2	$111 = 3\,(28) + 27$	$0 < 27 < 28$
Langkah ke-3	$28 = 1(27) + 1$	$0 < 1 < 27$
Langkah ke-4	$27 = 27(1) + 0$	

Maka $\gcd(250, 111) = 1$. *Selanjutnya, untuk mendapatkan kombinasi linearnya kita lakukan langkah balik. Perhatikan pada Langkah ke-3:*

$$\begin{aligned} 1 &= 28 - 1\,(27) \\ &= 28 - 1\,(111 - 3\,(28)) \\ &= (-1)\,(111) + (4)\,(28) \\ &= (-1)\,(111) + (4)\,(250 - 2\,(111)) \\ &= (4)\,250 + (-9)\,(111) \end{aligned}$$

.Secara umum, untuk $k \in \mathbb{Z}$,

$$1 = (4 - 111k)\,250 + (-9 + 250k)\,111.$$

Algoritme Euclid dapat diperluas sehingga tidak hanya mengasilkan pembagi bersama terbesar dari dua intejer a *dan* b, *tetapi juga menghasilkan intejer* x *dan* y *yang memenuhi* $ax + by = d$; *diberikan dalam Prosedur 7.*

Prosedur berikut ini disebut *Algoritme Euclides yang diperluas.*

Prosedur 1

procedure *gcd*$(a, b$: intejer positif, positif, $a \geq b)$
begin
  if $b = 0$ **then**
    begin
      $d := a$, $x := 1$, $y := 0$
      return(d, x, y)
    end
  $x_2 := 1$, $x_1 := 0$, $y_2 := 0$, $y_1 := 1$
  while $b > 0$ **do**
    begin
      $q := \lfloor \frac{a}{b} \rfloor$, $r := a - qb$, $x := x_2 - qx_1$, $y := y_2 - qy_1$
      $a := b$, $b := r$, $x_2 := x_1$, $x_1 := x$, $y_2 := y_1$, $y_1 := y$
    end
  $d := a$, $x := x_2$, $y := y_2$
  return(d, x, y)
end

Contoh 4.3 *Tabel berikut menunjukkan langkah-langkah Prosedur 1 dengan input* $a = 4864$ *dan* $b = 3458$, *diperoleh* $\gcd(4864, 3458) = 38$ *dan* $(4864)(32) + (3458)(-45) = 38$.

q	r	x	y	a	b	x_2	x_1	y_2	y_1
–	–	–	–	4864	3458	1	0	0	1
1	1406	1	−1	3458	1406	0	1	1	−1
2	646	−2	3	1406	646	1	−2	−1	3
2	114	5	−7	646	114	−2	5	3	−7
5	76	−27	38	114	76	5	−27	−7	38
1	38	32	−45	76	38	−27	32	38	−45
2	0	−91	128	38	0	32	−91	−45	128

Definisi 4.5 *Suatu intejer* $p \geq 2$ *disebut* ***prima*** *(prime) jika pembagi positifnya hanyalah* 1 *dan* p. *Intejer yang bukan prima disebut* ***komposit*** *(composite).*

Lemma 4.1 *Jika* $n \in \mathbb{Z}^+$ *adalah komposit, maka ada prima* p *sehingga* $p \mid n$.

Teorema 4.5 *Jika* p *prima dan* $p|ab$, *maka* $p|a$ *atau* $p|b$.

Teorema 4.6 *(Teorema Dasar Aritmatika) Setiap intejer* $n \geq 2$ *dapat difaktorisasikan secara tunggal sebagai produk kuasa prima:*

$$n = p_1^{e_1} p_2^{e_2} ... p_k^{e_k},$$

dimana p_i *prima berbeda dan* e^i *intejer positif.*

Beberapa contoh dari teorema di atas: $63 = 3^2.7$, $100 = 2^2.5^2$, $4864 = 2^8.19$, $3458 = 2.7.13.19$.

Aljabar Intejer Modulo n

Misalkan n adalah intejer positif.

$$n = 11 \times 13$$

: 143

$$100^{-1} \bmod 143$$

: 133

Definisi 4.6 *Misalkan a dan b adalah intejer, maka a disebut* ***kongruen ke*** *b modulo n, ditulis $a \equiv b \pmod{n}$, apabila n membagi $(a - b)$. Intejer n disebut* ***modulus*** *dari kongruensi tersebut.*

Contoh 4.4 $24 \equiv 9 \pmod{5}$ *karena* $24 - 9 = 3.5$, *dan* $-11 \equiv 17 \pmod{7}$ *karena* $-11 - 17 = (-4)(7)$.

Teorema 4.7 *(Sifat-sifat kongruensi) Untuk semua $a, a_1, b, b_1, c \in \mathbb{Z}$, berlaku berikut ini.*

1. *$a \equiv b \pmod{n} \Leftrightarrow a$ dan b mempunyai sisa yang sama apabila dibagi n.*
2. *Refleksif.: $a \equiv a \pmod{n}$.*
3. *Simetrik: jika $a \equiv b \pmod{n}$, maka $b \equiv a \pmod{n}$.*
4. *Transitif: jika $a \equiv b \pmod{n}$ dan $b \equiv c \pmod{n}$, maka $a \equiv c \pmod{n}$.*
5. *Jika $a \equiv a_1 \pmod{n}$ dan $b \equiv b_1 \pmod{n}$, maka $a + b \equiv a_1 + b_1 \pmod{n}$ dan $ab \equiv a_1 b_1 \pmod{n}$.*

Definisi 4.7 ***Intejer modulo*** **n**, *dinotasikan $\mathbb{Z}_n$, adalah himpunan (kelas ekuavalensi) intejer $\{0, 1, 2, ..., n - 1\}$ yang dikenai operasi: jumlah, kurang, dan kali diperlakukan dalam modulo n.*

Contoh 4.5 $\mathbb{Z}_{10} = \{0, 1, 2, ..., 9\}$. *Di dalam* $\mathbb{Z}_{10}$,

$$\begin{aligned} 6 + 7 &= 3 \\ 4 \times 8 &= 2 \\ 3 - 9 &= 3 + 1 = 4. \end{aligned}$$

Definisi 4.8 *Misalkan $a \in \mathbb{Z}_n$,* ***Invers multiplikatif dari*** *a* ***modulo*** *n adalah suatu intejer $x \in \mathbb{Z}_n$ sehingga $ax \equiv 1 \pmod{n}$. Faktanya tidak semua anggota $\mathbb{Z}_n$ mempunyai invers (x belum tentu ada). Dalam hal x yang bersangkutan ada, maka a disebut* ***invertibel*** *dan x disebut* ***invers*** *dari a, dinotasikan $x = a^{-1}$. Selanjutnya,* **a** ***dibagi***

Definisi 4.9 **b** ***modulo n*** *diartikan sebagai a* ***kali*** *b^{-1} modulo n.*

Teorema 4.8 *Misalkan $a \in \mathbb{Z}_n$, a adalah invertible jika dan hanya jika* $\gcd(a, n) = 1$.

Contoh 4.6 *Di dalam* $\mathbb{Z}_9$, *unsur-unsur yang invertibel adalah* 1, 2, 4, 5, 7, *dan* 8. *Dalam hal ini,* $7^{-1} = 4$ *karena* $7.4 \equiv 1$ *(mod* 9).

$$\left(33^{-1}\right) \bmod 100$$

: 87

Catatan 4.1 *Berdasarkan Teorema 4.3,* $\gcd(a, n) = 1$ *jika dan hanya jika ada intejer* x *dan* y *sehingga*

$$ax + ny = 1 \Leftrightarrow ax - 1 = -ny \Leftrightarrow ax \equiv 1 \ (mod\ n).$$

Ini berarti x *adalah invers dari* a *modulo* n *dan untuk menghitung* x *dapat digunakan Prosedur 1, dengan input* a *dan* n.

Definisi 4.10 ***Grup multiplikatif*** *dari* $\mathbb{Z}_n$ *adalah himpunan*

$$\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n / \gcd(a, n) = 1\}.$$

Khususnya untuk n ***prima,*** $\mathbb{Z}_n^* = \{1, 2, ..., n-1\}$

Contoh: $\mathbb{Z}_{10}^* = \{1, 3, 7, 9\}$, $\mathbb{Z}_{15}^* = \{1, 2, 4, 7, 8, 11, 13, 14\}$, dan $\mathbb{Z}_5^* = \{1, 2, 3, 4\}$. Kardinalitas dari $\mathbb{Z}_n^*$, yaitu $|\mathbb{Z}_n^*|$, merupakan bilangan phi Euler, notasi $\phi(n) = |\mathbb{Z}_n^*|$.

Teorema 4.9 *(Teorema Fermat) Misalkan* p *adalah prima. Jika* $\gcd(a, p) = 1$, *maka*

$$a^{p-1} \equiv 1 \pmod{p}.$$

Khususnya, untuk sembarang intejer

$$a^p \equiv a \pmod{p}$$

Teorema 4.10 *(Teorema Euler) Jika* $a \in \mathbb{Z}_n^*$, *maka*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

Teorema 4.11 *Jika* p *dan* q *adalah dua intejer positif dengan* $\gcd(p, q) = 1$, *maka*

$$\phi(pq) = \phi(p).\phi(q).$$

Khususnya, jika p *dan* q *keduanya prima, maka*

$$\phi(pq) = (p-1)(q-1)$$

$$37^{-1} \bmod 120$$

$$8^{-1} \bmod 11$$

: 7

4.2.2 Algoritme RSA

Algoritme 4.1 ***(Pembangkitan Kunci RSA)***
RINGKASAN: Dalam skema enkripsi kunci publik, entitas B membuat kunci publik dan kunci pribadi dengan langkah-langkah sebagai berikut:

1. *Bangkitkan dua prima berbeda p dan q yang memenuhi syarat keamanan*
2. *Hitung $n = pq$ dan $\Phi = (p-1)(q-1)$.*
3. *Pilih suatu intejer $e \in \mathbb{Z}_\Phi^*$.*
4. *Gunakan algoritme Euclid (Prosedur 1) untuk menghitung intejer $d = e^{-1}$ dalam $\mathbb{Z}_\Phi^*$*
5. *Kunci publik: pasangan e; kunci pribadi: d, dan n adalah parameter publik.*

Definisi 4.11 *Intejer e dalam pembangkitan kunci RSA disebut* ***eksponen enkripsi****, dan d disebut* ***eksponen dekripsi****.*

Algoritme 4.2 ***(Enkripsi dan Dekripsi RSA)***
RINGKASAN: Entitas A mengenkripsi pesan m menjadi siferteks c untuk entitas B, dan B mendekripsi c menjadi m.

1. ***Enkripsi.*** *A melakukan langkah-langkah berikut:*
   (a) *Representasikan pesan sebagai intejer m, yaitu $m \in \mathbb{Z}_n$.*
   (b) *Gunakan Algoritme Eksponen untuk menghitung $c = m^e$ dalam aritmetik $\mathbb{Z}_n$*
   (c) *Kirim c ke B.*
2. ***Dekripsi.*** *B melakukan langkah-langkah berikut:*
   (a) *Gunakan kunci pribadi d untuk mendapatkan $m = c^d$ dalam aritmetik $\mathbb{Z}_n$.*

Bukti bahwa algoritme enkripsi dan dekripsi adalah saling invers.

Karena e dan d berada di dalam grup multiplikasi $\mathbb{Z}_\Phi^*$, maka $ed = 1$ dalam $\mathbb{Z}_\Phi^*$, sehingga ada $k \in \mathbb{Z}$ yang memenuhi $ed = k\Phi + 1$. Karena p prima,

berdasarkan teorema Fermat, untuk m bukan kelipantan dari p ($\gcd(m,p) = 1$), berlaku

$$\begin{aligned} m^{p-1} &\equiv 1(\mathrm{mod}\, p) \Leftrightarrow \\ \left(m^{p-1}\right)^{q-1} &\equiv (1)^{q-1}\,(\mathrm{mod}\, p) \Leftrightarrow \\ m^{(p-1)(q-1)}.m &\equiv 1.m(\mathrm{mod}\, p) \Leftrightarrow \\ m^{(p-1)(q-1)+1} &\equiv m(\mathrm{mod}\, p) \end{aligned}$$

Di lain pihak untuk $\gcd(m,p) = p$, jelas berlalaku $m^{(p-1)(q-1)+1} \equiv m(\mathrm{mod}\, p)$. Dengan demikian, secara umum untuk sembarang $m \in \mathbb{Z}_n$ berlaku

$$\begin{aligned} m^{(p-1)(q-1)+1} &\equiv m(\mathrm{mod}\, p) \Leftrightarrow \\ m^{ed} &\equiv m(\mathrm{mod}\, p) \end{aligned}$$

Dengan argumen yang sama, karena q prima, untuk sembarang $m \in \mathbb{Z}_n$ berlaku

$$m^{ed} \equiv m(\mathrm{mod}\, q)$$

Akhirnya, karena p dan q dua berbeda, maka juga berlaku

$$m^{ed} \equiv m(\mathrm{mod}\, pq) \Leftrightarrow m^{ed} \equiv m(\mathrm{mod}\, n)$$

Algoritme 4.3 ***(Eksponen dalam $\mathbb{Z}_n$)***
INPUT: $a \in \mathbb{Z}_n$ dan intejer: $0 \leq k < n$ dengan representasi biner

$$k = \sum_{i=0}^{t} k_i 2^i.$$

OUTPUT: $a^k \bmod n$.

1. $b \leftarrow 1$. *Jika* $k = 0$, *maka* **return**(b)
2. $A \leftarrow a$.
3. *Jika* $k_0 = 1$ *maka* $b \leftarrow a$.
4. *Untuk* $i = 1$ *s.d.* t *hitung:*
   (a) $A \leftarrow A^2 \bmod n$.
   (b) *Jika* $k_i = 1$ *maka* $b \leftarrow A.b \bmod n$.

5. **return**(b).

$$100^{13} \bmod 143$$

: 100

Contoh 4.7 *(Ilustrasi kalkulasi algoritme RSA)*

1. ***Pembangkitan kunci.*** *Misalkan dipilih dua bialngan prima* 16 *bit:* $p =$

4.2.3 Keamanan Algoritme RSA

Tujuan musuh pasif di dalam skema kunci publik yang menggunakan algoritme RSA adalah ingin mendapatkan plainteks **m** dari siferteks **c** dengan bermodalkan pengetahuan tentang kunci publik (n, e). Di dalam teori bilangan, masalah ini disebut dengan problem RSA. Salah satu pendekatan yang mungkin dilakukan dalam memecahkan problem RSA adalah pertama kali memfaktorkan $n = pq$, kemudian menghitung $\phi = (p-1)(q-1)$, dan selanjutnya menghitung d sebagai invers dari e yang pada akhirnya didapatkan $\mathbf{m} = \mathbf{c}^d$. Dengan pendekatan ini bisa dikatakan bahwa problem RSA ekuivalen dengan problem faktorisasi n. Sejauh ini belum ada algoritme yang efisien untuk memecahkan problem tersebut. Beberapa algoritme tentang faktorisasi intejer: *Trial Division*, *Pollard's rho*, *Pollard's* $p-1$, *Elliptic Curve*, *Quadratic Sieve*, *Random Square Factoring Methods*, dan *Number Field Sieve Factoring.*

Kemajuan faktorisasi intejer dalam dasawarsa terakhir mengakibatkan modulus n dengan panjang 512 bit hanya memberikan keamanan marjinal. Sejak tahun 1996, untuk menghindari algoritme *Quadratic Sieve* dan *Number Field Sieve,* penggunaan modulus n dengan panjang minimal 768 bit dianjurkan, dan 1024 bit atau lebih besar merupakan pilihan yang bijak demi keamanan yang lebih lama.

4.3 Kriptografi Kunci Simetrik lawan Kunci Publik

Skema enkripsi kunci simetrik dan kunci publik mempunyai berbagai kelebihan dan kekurangan, beberapa diantaranya keduanya mempunyai kesamaan.

4.3.1 Kelebihan Kriptografi Kunci Simetrik

1. Sandi kunci simetrik dapat didisain untuk mempunyai laju pemrosesan data yang tinggi. Beberapa implementasi *hardware* mampu mencapai laju enkripsi sampai ratusan megabyte per detik, sementara implementasi *software* bisa mencapai laju pemrosesan dalam satuan megabyte per detik.

2. Kunci untuk sandi kunci simetrik relatif pendek.

3. Sandi kunci simetrik dapat diterapkan sebagai perangkat dasar untuk mengkonstruksi berbagai mekanisme kriptografik termasuk pembangkitan bilangan pseudorandom.

4. Sandi kunci simetrik dapat dikomposisikan untuk memproduksi sandi yang lebih kuat.

4.3.2 Kekurangan Kriptografi Kunci Simetrik

1. Dalam dua partai yang berkomunikasi, kunci harus dirahasiakan oleh keduanya sampai komunikasi berakhir.

2. Dalam jaringan yang besar, ada banyak pasangan yang harus dikelola. Akibatnya, manajemen kunci yang efektif memerlukan penggunaan TTP[1] terpercaya secara kondisional.

3. Dalam dua partai yang berkomunikasi antara entitas A dan B, praktik kriptografik yang sehat memerintahkan kuncinya harus diubah sesering mungkin.

4. Mekanisme penandaan dijitel yang bersumber pada enkripsi kunci simetrik memerlukan kunci yang besar untuk fungsi verifikasi publik atau menggunakan TTP.

4.3.3 Kelebihan Kriptografi Kunci Publik

1. Hanya kunci pribadi yang harus dijaga kerahasiaannya (akan tetapi kunci publik harus dijaga keautentikannya).

2. Administrasi kunci pada suatu jaringan memerlukan hanya TTP terpercaya secara fungsional.

[1] Akan dibahas pada subbab berikutnya.

3. Ketergantungan pada mode penggunaan, pasangan kunci pribadi/publik bisa dipertahankan untuk suatu periode waktu tertentu.

4. Banyak skema kunci publik yang menghasilkan mekanisme penandaan dijitel yang relatif efisien.

4.3.4 Kekurangan Kriptografi Kunci Publik

1. Laju pemrosesan data untuk metode enkripsi kunci publik umumnya beberapa tingkat lebih rendah dibandingkan skema kunci simetrik terbaik.

2. Ukuran kunci yang diperlukan jauh lebih besar dibandingkan ukuran kunci untuk enkripsi kunci simetrik.

3. Tidak ada skema kunci publik yang benar-benar terjamin keamanannya. Ini karena kebanyakan enkripsi kunci publik yang efektif menggatungkan keamanannya pada pemecahan problem teori bilangan.

Chapter 5

Fungsi Hash

Salah satu primitif kriptografi modern adalah fungsi hash kriptografik, sering kali disebut *fungsi hash satu-arah.* Berikut diberikan definisi sederhana fungsi hash (tidak harus satu-arah).

Definisi 5.1 ***Fungsi hash*** *adalah fungsi yang secara komputasi efisien memetakan bitsring dengan panjang sembarang ke bitstring dengan panjang tetap yang disebut* ***nilai-hash*** *(hash-value).*

Untuk fungsi hash dengan output nilai-hash n-bit, probabilitas pemilihan string secara random yang dipetakan ke nilai-hash n-bit adalah 2^{-n}. Berdasarkan definisi di atas, ide dasar dari fungsi hash adalah membuat string input menjadi teratur rapat dengan panjang seragam. Terkait dengan kegunaan kriptografi, fungsi hash h dipilih sedemikian sehingga secara komputasi tak-layak menentukan input berbeda $\mathbf{x}$ dan $\mathbf{y}$ sehingga $h(\mathbf{x}) = h(\mathbf{y})$.

5.1 Klasifikasi dan Definisi

5.1.1 Klasifikasi Umum

Secara umum fungsi hash dibagi menjadi dua kelas: *fungsi hash tak-berkunci* (unkeyed hash function) dan fungsi *hash berkunci* (keyed hash function). Fungsi hash tak-berkunci mempunyai spesifikasi mengatur parameter input tunggal (pesan). Fungsi hash berkunci mempunyai spesifikasi mengatur parameter dua input berbeda (pesan dan kunci). Terkait dengan klasifikasi tersebut, berikut didefinisikan fungsi hash yang lebih ditekankan pada sifat-sifatnya.

Definisi 5.2 ***Fungsi hash*** *adalah fungsi h yang mempunyai minimal dua sifat berikut:*

1. ***kompresi*** *(compression): h memetakan input $\mathbf{x}$ dengan sembarang panjang bit yang berhingga, ke output $h(\mathbf{x})$ dengan panjang bit tetap n.*

2. ***kemudahan komputasi*** *(ease of computation): diketahui h dan suatu input $\mathbf{x}$, $h(\mathbf{x})$ mudah dihitung.*

Berdasarkan kegunaanya, fungsi hash dibedakan atas dua tipe:

1. **MDC**

   MDC disebut juga dengan *manipulation detection code* (MDC) atau *message integrity code* (MIC). MDC merupakan subkelas dari fungsi hash tak-berkunci. Tujuan dari MDC, dikaitkan dengan suatu mekanisme, adalah untuk memberikan jaminan integritas data sebagaimana diperlukan dalam suatu aplikasi khusus. Dua bahasan yang termasuk dalam MDC:

   (a) *fungsi hash satu-arah* (one-way hash function - OWHF); fungsi hash ini mempunyai sifat bahwa diketahui suatu nilai-hash untuk menentukan sembarang inputnya adalah tak-layak.

   (b) *fungsi hash tahan tumbukan* (collision resistant hash function - CRHF); fungsi hash ini mempunyai sifat bahwa diketahui suatu nilai-hash untuk menentukan sembarang dua inputnya adalah tak-layak.

2. **MAC**

   MAC merupakan subkelas dari fungsi hash berkunci. MAC mempunyai dua parameter berbeda, yaitu: input pesan dan kunci rahasia. Sesuai dengan namanya, tujuan MAC (tanpa terkait dengan mekanisme yang lain) adalah untuk menjamin integritas pesan dan asal pesan.

5.1.2 Sifat Dasar dan Definisi

Menyambung dua sifat pada Definisi 5.2, berikut ini diberikan tiga sifat penting untuk fungsi hash tak-berkunci h dengan input $\mathbf{x}, \mathbf{x}'$ dan output $\mathbf{y}, \mathbf{y}'$:

1. *ketahanan preimej* (preimage resistance): jika diketahui sembarang nilai-hash $\mathbf{y}$, maka secara perhitungan tak-layak mencari $\mathbf{x}'$ sehingga $h(\mathbf{x}') = \mathbf{y}$. Istilah lain untuk ketahanan preimej adalah *satu-arah* (one-way).

2. *ketahanan preimej kedua* (2nd-preimage resistance): jika diketahui $\mathbf{x}$, maka secara perhitungan tak-layak mencari $\mathbf{x}' \neq \mathbf{x}$ sehingga $h(\mathbf{x}') = h(\mathbf{x})$. Istilah lain untuk ketahanan preimej kedua adalah *ketahanan tumbukan lemah* (weak collision resistance).

3. *ketahanan tumbukan* (collision resistance): secara perhitungan tak-layak mencari dua input berbeda $\mathbf{x}, \mathbf{x}'$ sehingga $h(\mathbf{x}') = h(\mathbf{x})$. Istilah lain untuk ketahanan tumbukan adalah *ketahanan tumbukan kuat* (strong collision resistance).

Definisi 5.3 *OWHF adalah fungsi hash dengan dua sifat pada Definisi 5.2 ditambah sifat ketahanan preimej dan ketahanan preimej kedua. Istilah lain untuk OWHF adalah* ***fungsi hash satu-arah lemah****.*

Definisi 5.4 *CRHF adalah fungsi hash dengan dua sifat pada Definisi 5.2 ditambah sifat ketahanan preimej kedua dan ketahanan tumbukan. Istilah lain untuk CRHF adalah* ***fungsi hash satu-arah kuat****.*

Definisi 5.5 *Algoritme MAC adalah keluarga fungsi h_k dengan parameter kunci rahasia k mempunyai sifat:*

1. ***kemudahan komputasi*** *(ease of computation): diketahui fungsi h_k, diberikan nilai k dan input $\mathbf{x}$, maka $h_k(\mathbf{x})$ mudah dihitung. Hasil fungsi ini disebut* ***nilai-MAC*** *atau* ***MAC*** *saja.*

2. ***kompresi*** *(compression): h_k memetakan input x dengan sembarang panjang bit berhingga, ke output $h_k(\mathbf{x})$ dengan panjang bit tetap n.*

   Selanjutnya, diberikan suatu deskripsi dari keluarga fungsi h, untuk setiap nilai tetap yang dibolehkan k (musuh tidak boleh mengetahui), sifat berikut dipenuhi:

3. ***ketahanan komputasi*** *(computation resistance): diberikan nol atau lebih pasangan teks-MAC $(\mathbf{x}_i, h_k(\mathbf{x}_i))$, secara komputasi tak-layak menghitung sembarang pasangan $(\mathbf{x}, h_k(\mathbf{x}))$ untuk sembarang input baru $\mathbf{x} \neq \mathbf{x}_i$ (termasuk mungkin $h_k(\mathbf{x}) = h_k(\mathbf{x}_i)$ untuk suatu i) tanpa mengetahui kunci k.*

Jika ketahanan komputasi tidak dipenuhi, algoritme MAC disebut *pemalsuan-MAC*. Ketahanan komputasi mengakibatkan secara perhitungan tak-layak menentukan kunci k jika diketahui satu atau lebih pasangan teks-MAC $(\mathbf{x}_i, h_k(\mathbf{x}_i))$. Sebaliknya, ketahanan kunci tidak harus mengakibatkan ketahanan komputasi.

Serangan terhadap MDC: *(Ongoing Work)*

Serangan terhadap MAC: *(Ongoing Work)*

5.2 Konstruksi Umum Fungsi Hash

Kebanyakan fungsi hash h didisain melalui proses iteratif yang menginputkan bitstring dengan panjang sembarang dan mengoutputkan bitstring dengan panjang tetap (Gambar 9.2, handbook, Hal. 332). Misalkan $\mathbf{x}$ adalah bitstring dengan panjang sembarang $|\mathbf{x}| = b$, nilai hash $h(\mathbf{x})$ adalah bitstring dengan panjang tetap $|h(\mathbf{x})| = n$ dihitung melaui tahapan proses berikut ini.

1. **Proses Penambahan Bit Ekstra** (padding). Input $\mathbf{x}$ dibagi menjadi t blok,
$$\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3...\mathbf{x}_t,$$
yang setiap blok $\mathbf{x}_i$ mempunyai panjang $|\mathbf{x}_i| = r$. Dalam hal b bukan kelipatan dari r, dilakukan proses penambahan bit ektra (padding) agar blok terakhir mempunyai panjang r. Sering juga ditambahkan blok ekstra $\mathbf{x}_{t+1}$ sebagai representasi biner rata-kanan dari b. Blok ini menyimpan informasi panjang $\mathbf{x}$ yang asli.

2. **Proses Kompresi**. Setiap blok $\mathbf{x}_i$ akan bertindak sebagai input dari *fungsi kompresi* f yang menghitung secara iteratif dengan rumusan
$$\mathbf{H}_0 = VI; \quad \mathbf{H}_i = f(\mathbf{H}_{i-1}, \mathbf{x}_i); \quad 1 \leq i \leq t.$$
$\mathbf{H}_0$ adalah vektor inisial yang nilainya didefinisikan sebelumnya, $\mathbf{H}_i$ adalah nilai berrantai yang disebut *variabel berrantai* (channing variable) merupakan bitstring dengan panjang $|\mathbf{H}_i| = n$. Output dari proses ini adalah $\mathbf{H}_t$.

3. **Proses Finalisasi**. Proses ini sifatnya opsional, yaitu mentransformasikan $\mathbf{H}_t$ oleh fungsi g ke bitsring dengan panjang m,
$$h(\mathbf{x}) = g(\mathbf{H}_t); \quad |h(\mathbf{x})| = n.$$
Jika proses ini tidak dilakukan berarti g adalah fungsi identitas, yaitu $g(\mathbf{H}_t) = \mathbf{H}_t$.

Tentunya di dalam pendefinisian komponen-komponen proses di atas, kita harus mempertimbangkan sifat keamanan dari h, yaitu sebagai fungsi satu-arah dan tahan tumbukan. Misalnya, agar tahan terhadap *serangan harilahir* (birthday attack), dianjurkan m cukup besar. Hal ini bisa dilakukan dengan mendefinisikan $h(\mathbf{x}) = \mathbf{H}_{t-1}||\mathbf{H}_t$.

Suatu fakta menyatakan bahwa fungsi kompresi yang tahan tumbukan dapat digunakan untuk mengkonstruksi fungsi hash tahan tumbukan. Fakta ini dirinci dalam algoritme berikut.

Algoritme 5.1 ***(Merkle meta-method for hashing)***
INPUT: Fungsi kompresi tahan tumbukan f yang memetakan string $(n+r)$-bit ke string n-bit
OUTPUT: Fungsi hash h yang tahan tumbukan.

1. *Dipecah $\mathbf{x}$ menjadi $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3...\mathbf{x}_t$ dengan $|\mathbf{x}_i| = r$. Jika $r \nmid b$, lakukan padding pada $\mathbf{x}_t$ dengan bit-0 agar $|\mathbf{x}_t| = r$*
2. *Definisikan blok ekstra $\mathbf{x}_{t+1} = (b)_2$ sebagai representasi biner rata-kanan (anggap sebelumnya bahwa $b < 2^r$)*
3. *Definisikan nilai hash n-bit sebagai*

$$h(\mathbf{x}) = \mathbf{H}_{t+1} = f(\mathbf{H}_t||\mathbf{x}_{t+1})$$

dihitung dari

$$\mathbf{H}_0 = 0^n; \quad \mathbf{H}_i = f(\mathbf{H}_{i-1}||\mathbf{x}_i); \quad 1 \leq i \leq t+1.$$

Catatan bahwa 0^n adalah string n-bit yang semua simbolnya 0.

Metode padding diberikan dalam dua algoritme berikut.

Algoritme 5.2 ***(Metode Padding 1)***
INPUT: Bitstring $\mathbf{x}$ dengan panjang bebas.
OUTPUT: Bitstring $\mathbf{x}'$ dengan panjang kelipatan dari r.

1. *Rangkaikan pada $\mathbf{x}$ dengan string bit-0 sehingga menjadi bitsring $\mathbf{x}'$ dengan panjang kelipatan dari r.*

Algoritme 5.3 ***(Metode Padding 2)***
INPUT: Bitstring $\mathbf{x}$ dengan panjang bebas.
OUTPUT: Bitstring $\mathbf{x}'$ dengan panjang kelipatan dari r.

1. *Rangkaikan pada* $\mathbf{x}$ *dengan satu bit*-1.
2. *Kemudian, rangkaikan dengan string bit*-0 *sehingga menjadi bitsring* $\mathbf{x}'$ *dengan panjang kelipatan dari* r.

Metode padding 1 menimbulkan kerancuan karena batasan antara bitstring sebelum dan sesudah padding tidak bisa dibedakan kecuali kalau bitstring sebelum padding diketahui panjangnnya. Hal ini tidak terjadi pada metode 2, bahkan ketika panjang dari $\mathbf{x}$ sudah merupakan kelipatan dari r, padding pada $\mathbf{x}$ menciptakan blok ekstra

5.3 Algoritme MDC

Berbekal pada struktur konstruksi umum MDC, algoritme MDC bisa dikonstruksi berdasarkan tiga kategori:

1. MDC berbasis pada sandi blok.
2. MDC untuk tujuan khusus (customized).
3. MDC berbasis pada aritmatik modular.

5.3.1 MDC Berbasis Pada Sandi Blok

Motivasi praktis konstruksi MDC berbasis sandi blok adalah apabila implementasi efisien dari sandi blok sudah tersedia di dalam suatu sistem, dan kemudian menggunakannya sebagai komponen sentral dalam pembentukan fungsi hash dengan biaya tambahan yang cukup kecil. Tentu saja keseluruhan konstruksi harus mempertimbangkan aspek keamanan.

Definisi 5.6 *Misalkan* h *adalah fungsi hash teriterasi yang dinkontruksi dari suatu sandi blok, dengan fungsi kompresi* f *yang mengerjakan* s *enkripsi blok untuk memproses secara berrantai blok pesan* r*-bit. Dalam hal ini,* h *dikatakan mempunyai* laju $= \frac{1}{s}$.

Tiga algoritme berikut merupakan dasar dari konstruksi MDC berbasis sandi blok.

Algoritme 5.4 ***(Matyas-Meyer-Oseas hash**)*
INPUT: Bitstring $\mathbf{x}$ *dengan panjang bebas.*
OUTPUT: Nilai hash n*-bit dari* $\mathbf{x}$.

1. *Dipecah* $\mathbf{x}$ *menjadi* $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3...\mathbf{x}_t$ *dengan* $|\mathbf{x}_i| = n$, *lakukan padding pada* $\mathbf{x}$ *jika diperlukan, dan definisikan vektor inisial* VI *berukuran* n*-bit.*

2. *Outputnya adalah* $\mathbf{H}_t$ *yang dihitung dari*

$$\mathbf{H}_0 = VI; \quad \mathbf{H}_i = E_{g(\mathbf{H}_{i-1})}(\mathbf{x}_i) \oplus \mathbf{x}_i; \quad 1 \leq i \leq t,$$

dimana E_K *adalah algoritme enkripsi dengan kunci* K, *dan* g *adalah fungsi yang memetakan input* n*-bit ke* K.

Algoritme 5.5 ***(Davies-Meyer hash)***
INPUT: Bitstring $\mathbf{x}$ *dengan panjang bebas.*
OUTPUT: Nilai hash r*-bit dari* $\mathbf{x}$.

1. *Dipecah* $\mathbf{x}$ *menjadi* $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3...\mathbf{x}_t$ *dengan* $|\mathbf{x}_i| = k$ *(*k *adalah panjang kunci enkripsi* K*), lakukan padding pada* $\mathbf{x}$ *jika diperlukan, dan definisikan vektor inisial* n*-bit* VI.

2. *Outputnya adalah* $\mathbf{H}_t$ *yang dihitung dari*

$$\mathbf{H}_0 = VI; \quad \mathbf{H}_i = E_{\mathbf{x}_i}(\mathbf{H}_{i-1}) \oplus \mathbf{H}_{i-1}; \quad 1 \leq i \leq t,$$

dimana E_K *adalah algoritme enkripsi dengan kunci* K.

Algoritme 5.6 ***(Miyaguchi-Preneel hash)***
INPUT: Bitstring $\mathbf{x}$ *dengan panjang bebas.*
OUTPUT: Nilai hash n*-bit dari* $\mathbf{x}$.

1. *Dipecah* $\mathbf{x}$ *menjadi* $\mathbf{x} = \mathbf{x}_1\mathbf{x}_2\mathbf{x}_3...\mathbf{x}_t$ *dengan* $|\mathbf{x}_i| = n$, *lakukan padding pada* $\mathbf{x}$ *jika diperlukan, dan definisikan vektor inisial* n*-bit* VI.

2. *Outputnya adalah* $\mathbf{H}_t$ *yang dihitung dari*

$$\mathbf{H}_0 = VI; \quad \mathbf{H}_i = E_{g(\mathbf{H}_{i-1})}(\mathbf{x}_i) \oplus \mathbf{x}_i \oplus \mathbf{H}_{i-1}; \quad 1 \leq i \leq t,$$

dimana E_K *adalah algoritme enkripsi dengan kunci* K, *dan* g *adalah fungsi yang memetakan input* n*-bit ke* K.

MDC-2 dan MDC-4 merupakan kode deteksi manipulasi yang memerlukan bertuturut-turut 2 dan 4 operasi sandi blok untuk setiap blok input pesan. Keduanya menggunakan skema Matyas-Meyer-Oseas untuk menghasilkan nilai hash dengan panjang ganda. Apabila sandi blok yang digunakan adalah DES, mereka akan menghasilkan nilai hash 128-bit. Spesifikasi komponen-komponen sebelumnya menggunakan:

1. $E_K(\mathbf{m})$ adalah algoritme DES dengan input $\mathbf{m}$ dan kunci K, dimana $|\mathbf{m}| = 64$ dan $|K| = 56$,
2. dua fungsi g dan $\widetilde{g}$ memetakan string 64-bit ke string 56-bit.

Chapter 6

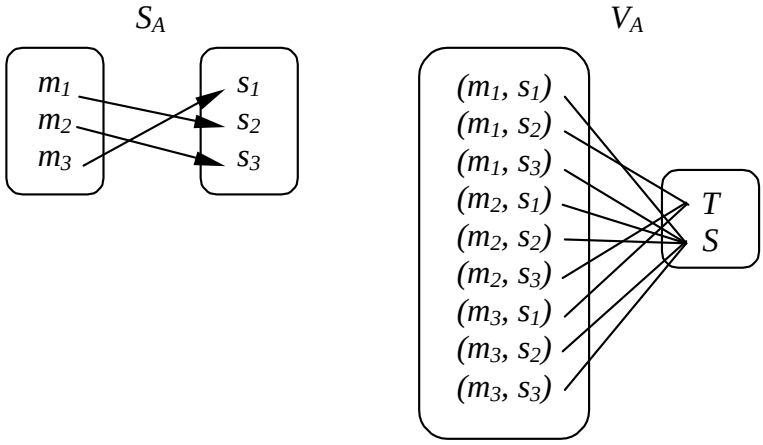
Penandaan Dijitel

Primitif kriptografi yang menjadi landasan di dalam autentikasi, autorisasi, dan non-repudiasi adalah *penandaan dijitel* (digital signature). Tujuan dari penandaan dijitel adalah memberikan suatu alat yang digunakan oleh entitas untuk mengikat identitasnya menjadi satu bagian dari informasi. Proses pemberian *tanda* (signature) pada informasi rahasia yang akan dikirim disebut *penandaan* (signing). Berikut diberikan dekripsi umum dari penandaan dijitel:

- $\mathcal{M}$ adalah himpunan pesan yang anggotanya akan diberi tanda.
- $\mathcal{S}$ adalah himpunan yang anggotanya disebut *tanda*, dan tanda biasanya berupa bitstring dengan panjang tetap.
- S_A adalah suatu transformasi dari $\mathcal{M}$ ke $\mathcal{S}$ yang disebut *transformasi penandaan* oleh entitas A. Transformasi S_A dijaga kerhasiaannya oleh A dan digunakan untuk memberi tanda pada pesan-pesan dalam $\mathcal{M}$.
- V_A adalah suatu transformasi dari himpunan $\mathcal{M} \times \mathcal{S}$ ke himpunan $\{T$ (benar), F (salah)$\}$. V_A disebut *transformasi verifikasi* untuk semua tanda milik A, diketahui oleh publik, dan transformasi ini digunakan oleh entitas lainnya untuk memeriksa kebenaran tanda dibuat oleh A.

Definisi 6.1 *S_A dan V_A memberikan **skema** (mekanisme) **penandaan dijitel**.*

Contoh 6.1 *Misalkan $\mathcal{M} = \{m_1, m_2, m_3\}$ dan $\mathcal{S} = \{s_1, s_2, s_3\}$. Gambar berikut mencontohkan suatu transformasi S_A dan V_A.*



Prosedur Penandaan

Entitas A (Penanda) menciptakan suatu tanda pada pesan $m \in \mathcal{M}$ dengan melakukan:

1. Hitung $s = S_A(m)$,
2. Kirim (m, s), s disebut tanda untuk pesan m.

Prosedur Verifikasi

Untuk memeriksa bahwa tanda s adalah benar diciptakan oleh A, maka entitas B (pemeriksa) melakukan:

1. Mencari fungsi verifikasi V_A dari A.
2. Hitung $u = V_A(m, s)$
3. Diterima apabila $u = T$ dan ditolak apabila $u = F$.

Dua sifat yang harus dipenuhi untuk tranformasi penandaan dan verifikasi:

(a) s adalah tanda yang sah dari A jika dan hanya jika $V_A(m, s) = T$.

(b) Secara perhitungan tak-layak untuk entitas selain A mendapatkan $s \in \mathcal{S}$, untuk sembarang $m \in \mathcal{M}$, sehingga $V_A(m, s) = T$.

Terkait dengan kegunaan penandaan dijitel secara praktis, berikut diberikan penotasian dan pendefinisian dengan pengertian yang lebih luas untuk skema penandaan dijitel.

1. *Ruang pesan*, dinotasikan $\mathcal{M}$, adalah himpunan yang anggota-anggotanya dapat dibubuhi *tanda dijitel* oleh penanda.

2. *Ruang penandaan*, dinotasikan $\mathcal{M}_S$, adalah himpunan yang anggota-anggotanya diterapkan transformasi penandaan. Catatan bahwa transformasi penandaan tidak harus secara langsung diterapkan pada himpunan $\mathcal{M}$.

3. *Ruang tanda*, dinotasikan $\mathcal{S}$, adalah himpunan yang anggota-anggotanya dapat dibubuhkan pada anggota-anggota $\mathcal{M}$. Dengan kata lain anggota $\mathcal{S}$ mewakili identitas penanda yang melekat pada pesan.

4. *Fungsi ridandensi* (redundancy function), dinotasikan R, adalah pemetaan $1-1$ dari $\mathcal{M}$ ke $\mathcal{M}_S$.

5. *Imej dari* R, dinotasikan $\mathcal{M}_R$.

6. *Himpunan indeks untuk penadaan* (indexing set for signing), dinotasikan $\mathcal{R}$, adalah himpunan yang digunakan untuk mengidentifikasi transformasi penandaan khusus.

7. Dinotasikan h, adalah fungsi satu-arah dengan domain $\mathcal{M}$.

8. *Ruang nilai hash*, dinotasikan $\mathcal{M}_h$, adalah imej dari h; $\mathcal{M}_h \subseteq \mathcal{M}_S$.

Skema penandaan dijitel diklasifikasikan menjadi dua kelas besar, yaitu:

1. *Skema penandaan dijitel dengan apendiks*, memerlukan pesan asli sebagai input untuk algoritme verifikasi.

2. *Skema penandaan dijitel dengan pemulihan pesan*, tidak memerlukan pesan asli sebagai input untuk algoritme verifikasi. Dalam hal ini, pesan asli dipulihkan dari tanda itu sendiri.

Kedua kelas tersebut selanjutnya dibagi lagi atas dasar apakah $|\mathcal{R}| = 1$ atau $|\mathcal{R}| > 1$.

Definisi 6.2 *Skema penandaan dijitel (baik dengan apendiks atau pemu- lihan pesan) disebut* ***deterministik*** *(deterministic) jika* $|\mathcal{R}| = 1$, *dan disebut* ***randomized*** *jika* $|\mathcal{R}| > 1$.

6.1 Skema Penandaan Dijitel dengan Apendiks

Skema penandaan dijitel dengan apendiks adalah yang paling umum digunakan dalam praktik. Contoh beberapa skema yang menggunakan model ini diantaranya: skema penandaan DSA, ElGamal, dan Schnorr. Prosedur skema tersebut diberikan dalam algoritme berikut:

Algoritme 6.1 ***(Pembangkitan kunci)***
RINGKASAN: Setiap entitas membuat kunci pribadi digunakan untuk menandai pesan, dan terkait dengan kunci publik yang digunakan entitas lainnya untuk memeriksa tanda.

1. *Setiap entitas A memilih kunci pribadi yang mendefinisikan suatu himpunan transformasi*
$$\mathcal{S}_A = \{\mathcal{S}_{A,k}/k \in \mathcal{R}\}.$$
Masing-masing $\mathcal{S}_{A,k}$ adalah pemetaan $1-1$ dari $\mathcal{M}_h$ ke $\mathcal{S}$ dan disebut ***transformasi penandaan****.*

2. *$\mathcal{S}_A$ mendefinisikan suatu pemetaan terkait V_A dari $\mathcal{M}_h \times \mathcal{S}$ ke $\{T, F\}$ sedemikian sehingga*
$$V_A(\widetilde{m}, s^*) = \begin{cases} T, & \text{jika } \mathcal{S}_{A,k}(\widetilde{m}) = s^*, \\ F, & \text{jika selainnya,} \end{cases}$$
untuk semua $\widetilde{m} \in \mathcal{M}_h$, $s^* \in \mathcal{S}$; disini $\widetilde{m} = h(m)$ untuk semua $m \in \mathcal{M}$. V_A disebut ***transformasi verifikasi*** *dan dikonstruksi sedemikian sehingga ia bisa dihitung tanpa mengetahui kunci pribadi penanda.*

3. *Kunci publik milik A adalah V_A; kunci pribadi milik A adalah himpunan $\mathcal{S}_A$.*

Algoritme 6.2 ***(Pembangkitan tanda dan verifikasi)***
RINGKASAN: Entitas A menciptakan tanda $s \in \mathcal{S}$ untuk pesan $m \in \mathcal{M}$, yang kemudian diverifikasi oleh siapapun entitas B.

1. ***Pembangkitan tanda****. Entitas A seharusnya melakukan hal berikut:*
   (a) *Memilih suatu unsur $k \in \mathcal{R}$.*
   (b) *Menghitung $\widetilde{m} = h(m)$ dan $s^* = \mathcal{S}_{A,k}(\widetilde{m})$.*
   (c) *Tanda milik A untuk m adalah s^*. Keduanya disiapkan bagi entitas siapapun yang ingin memeriksanya.*

2. ***Verifikasi***. *Entitas B seharusnya melakukan hal berikut:*

   (a) *Mencari kunci publik autentik V_A milik A.*

   (b) *Menghitung $\widetilde{m} = h(m)$ dan $u = V_A(\widetilde{m}, s^*)$.*

   (c) *Tanda diterima jika dan hanya jika $u = T$.*

Sifat-sifat yang diperlukan oleh transformasi penandaan dan verifikasi adalah:

(i) untuk setiap $k \in \mathcal{R}$, seharusnya efisien untuk menghitung $\mathcal{S}_{A,k}$;

(ii) untuk menghitung V_A seharusnya efisien; dan

(iii) untuk siapapun entitas selain A, seharusnya secara perhitungan tak-layak mendapatkan $m \in \mathcal{M}$ dan $s^* \in \mathcal{S}$ sehingga $V_A(\widetilde{m}, s^*) = T$, dimana $\widetilde{m} = h(m)$.

Paling sering skema penandaan dijitel dengan pemulihan pesan diterapkan pada pesan-pesan yang panjangnya tetap, sementara skema penandaan dijitel dengan apendiks diterapkan pada pesan-pesan panjangnya sembarang. Dengan demikian, fungsi satu-arah h pada Algoritme 6.2 biasanya dipilih dari jenis *fungsi hash tahan tumbukan*[1]. Alternatifnya, pesan dipecah dalam blok-blok dengan panjang tetap dan kemudian digunakan skema penandaan dijitel dengan pemulihan pesan. Akan tetapi alternatif ini akan mengakibatkan implementasi yang relatif lambat dan berrisiko mengurangi keamanan, jadi metode hash masih lebih baik.

6.2 Skema Penandaan Dijitel dengan Pemulihan Pesan

Skema penandaan dijitel dengan pemulihan pesan mempunyai keutamaan bahwa pesan bertanda dapat dipulihkan dari tanda itu sendiri. Beberapa contoh dari skema ini diantaranya: skema pendandaan kunci publik RSA, Rabin, dan Nyberg-Rueppel.

Algoritme 6.3 ***(Pembangkitan kunci)***

RINGKASAN: Setiap entitas membuat kunci pribadi digunakan untuk menandai pesan, dan terkait dengan kunci publik yang digunakan entitas lainnya untuk memeriksa tanda.

[1]Fungsi ini akan dibahas pada subbab berikutnya.

1. *Setiap entitas A seharusnya memilih kunci pribadi yang mendefinisikan suatu himpunan transformasi*

$$\mathcal{S}_A = \{\mathcal{S}_{A,k}/k \in \mathcal{R}\}.$$

*Masing-masing $\mathcal{S}_{A,k}$ merupakan pemetaan $1-1$ dari $\mathcal{M}_S$ ke $\mathcal{S}$ dan disebut **transformasi penandaan**.*

2. *$\mathcal{S}_A$ mendefinisikan suatu pemetaan terkait V_A dengan sifat $V_A \circ \mathcal{S}_{A,k}$ adalah pemetaan identitas pada $\mathcal{M}_S$ untuk semua $k \in \mathcal{R}$. V_A disebut **transformasi verifikasi** dan dikonstruksi sedemikian sehingga ia bisa dihitung tanpa mengetahui kunci pribadi penanda.*

3. *Kunci publik milik A adalah V_A; kunci pribadi milik A adalah himpunan $\mathcal{S}_A$.*

Algoritme 6.4 ***(Pembangkitan tanda dan verifikasi)***
RINGKASAN: Entitas A menciptakan tanda $s \in \mathcal{S}$ untuk pesan $m \in \mathcal{M}$, yang kemudian diverifikasi oleh siapapun entitas B. Pesan m dipulihkan dari s.

1. ***Pembangkitan tanda**. Entitas A seharusnya melakukan hal berikut:*

   (a) *Memilih suatu unsur $k \in \mathcal{R}$.*
   (b) *Menghitung $\widetilde{m} = R(m)$ dan $s^* = \mathcal{S}_{A,k}(\widetilde{m})$.*
   (c) *Tanda milik A adalah s^* dan ini disiapkan untuk entitas siapapun yang ingin memeriksanya dan memulihkan m darinya.*

2. ***Verifikasi**. Entitas B seharusnya melakukan hal berikut:*

   (a) *Mencari kunci publik autentik V_A milik A.*
   (b) *Menghitung $\widetilde{m} = V_A(s^*)$.*
   (c) *Memeriksa apakah $\widetilde{m} \in \mathcal{M}_R$. Jika $\widetilde{m} \notin \mathcal{M}_R$, tanda ditolak.*
   (d) *Memulihkan m dari $\widetilde{m}$ dengan menghitung $R^{-1}(\widetilde{m})$.*

Sifat-sifat yang diperlukan oleh transformasi penandaan dan verifikasi adalah:

(i) untuk setiap $k \in \mathcal{R}$, seharusnya efisien untuk menghitung $\mathcal{S}_{A,k}$;

(ii) untuk menghitung V_A seharusnya efisien; dan

(iii) untuk siapapun entitas selain A, seharusnya secara perhitungan tak-layak mendapatkan $s^* \in \mathcal{S}$ sehingga $V_A(s^*) \in \mathcal{M}_R$.

Fungsi ridandensi R dan inversnya R^{-1} diketahui oleh publik. Memilih R yang sesuai merupakan keputusan yang sangat penting terkait dengan keamanan sistem. Sebagai ilustrasi, misalkan $\mathcal{M}_R = \mathcal{M}_S$. Andaikan R adalah bijeksi dari $\mathcal{M}$ ke $\mathcal{M}_R$ dan $\mathcal{S}_{A,k}$ adalah bijeksi dari $\mathcal{M}_S$ ke $\mathcal{S}$, maka $|\mathcal{M}| = |\mathcal{S}|$. Akibatnya, seandainya entitas selain A (Penyerang) memilih $s^* \in \mathcal{S}$ dan dipersilahkan untuk diverifikasi, maka Pemeriksa akan menghitung $\widetilde{m} = V_A(s^*)$ yang dipastikan berada di dalam $\mathcal{M}_R$ sehingga dengan mudah didapatkan pesan m. Selanjutnya. s^* disimpulkan oleh Pemeriksa sebagai tanda yang sah untuk pesan m diperoleh tanpa mengetahui himpunan transformasi penadaan $\mathcal{S}_A$. Berikut ini suatu contoh sederhana bagaimana mendefinisikan $\mathcal{M}_R$ dan $\mathcal{M}_S$ yang cukup aman untuk menanggulangi masalah tersebut.

Contoh 6.2 *Misalkan* $\mathcal{M} = \{\mathbf{m} \ / \ \mathbf{m} \in \{0,1\}^n\}$ *untuk suatu intejer positif tetap* n, *dan* $\mathcal{M}_S = \{\mathbf{t} \ / \ \mathbf{t} \in \{0,1\}^{2n}\}$. *Definisikan* $R : \mathcal{M} \to \mathcal{M}_S$ *dengan* $R(\mathbf{m}) = \mathbf{m}||\mathbf{m}$, *dimana* $||$ *menotasikan konketinasi (concatination), sehingga* $\mathcal{M}_R = \{\mathbf{m}||\mathbf{m} \ / \ \mathbf{m} \in \mathcal{M}\} \subseteq \mathcal{M}_S$. *Untuk nilai* n *yang besar, kuantitas* $\frac{|\mathcal{M}_R|}{|\mathcal{M}s|} = (\frac{1}{2})^n$ *bernilai sangat kecil yang bisa diabaikan. Penentuan fungsi ridandensi ini sangat sesuai karena pemilihan sembarang* s^* *oleh musuh akan menghasilkan probabilitas bahwa* $V(s^*) \in \mathcal{M}_R$ *sangat kecil yang bisa diabaikan.*

6.3 Enkripsi Kunci-publik Riversibel untuk Penandaan Dijitel

Misalkan E_e adalah transformasi enkripsi kunci publik dengan ruang pesan $\mathcal{M}$ dan ruang siferteks $\mathcal{C}$. Misalkan juga bahwa $\mathcal{M} = \mathcal{C}$. Jika D_d adalah transformasi dekripsi pasangan dari E_e, maka karena E_e dan D_d adalah permutasi, kita peroleh

$$D_d(E_e(m)) = E_e(D_d(m)) = m, \quad \forall m \in \mathcal{M}.$$

Skema enkripsi kunci publik dengan tipe seperti ini disebut *riversibel* (reversible). *Catatan* bahwa syarat $\mathcal{M} = \mathcal{C}$ adalah suatu keharusan karena, jika tidak, $D_d(m)$ menjadi *tidak mempunyai arti* karena $m \notin \mathcal{C}$.

Konstruksi untuk skema penandaan dijitel:

1. Misalkan $\mathcal{M}$ adalah ruang pesan untuk skema penandaan.

2. Misalkan $\mathcal{C} = \mathcal{M}$ adalah ruang tanda.

3. Misalkan (e, d) adalah pasangan kunci untuk skema enkripsi kunci publik.

4. Definisikan fungsi penandaan S_A sebagai D_d. Ini berarti tanda untuk pesan $m \in \mathcal{M}$ adalah $s = D_d(m)$.

5. Definisikan fungsi verifikasi V_A dengan

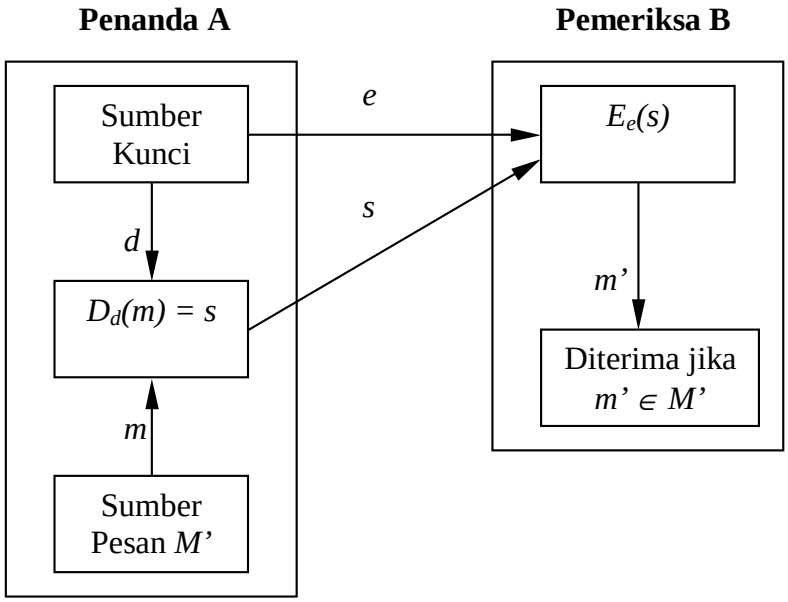
$$V_A(m, s) = \begin{cases} T, & \text{jika } E_e(s) = m, \\ F, & \text{untuk selainnya.} \end{cases}$$

Skema penandaan dapat disederhanakan lagi jika A hanya menandai pesan yang mempunyai *struktur khusus*, dan struktur ini boleh diketahui publik. Misalkan $\mathcal{M}' \subseteq \mathcal{M}$, dimana semua anggota $\mathcal{M}'$ adalah anggota-anggota $\mathcal{M}$ yang terdefinisikan dengan baik dengan struktur khusus. Contohnya, $\mathcal{M}$ adalah himpunan semua bitstring dengan panjang $2t$ untuk suatu intejer positif t. Misalkan $\mathcal{M}' \subseteq \mathcal{M}$ beranggota semua bitstring yang t bit pertama diulang dalam t posisi terakhir (ilustrasi: 101101 adalah anggota $\mathcal{M}'$ untuk $t = 3$). Jika A hanya menandai pesan-pesan dalam $\mathcal{M}'$, maka pemeriksa dengan mudah dapat mengenalinya.

Pendefinisian fungsi verifikasi menjadi

$$V_A(s) = \begin{cases} T, & \text{jika } E_e(s) \in \mathcal{M}', \\ F, & \text{untuk selainnya.} \end{cases}$$

Dengan skenario baru ini A hanya perlu mengirim tanda s karena pesan $m = E_e(s)$ dapat dipulihkan dengan menerapkan fungsi verifikasi. Skema yang demikian termasuk jenis *skema penandaan dijitel dengan pemulihan pesan*. Gambar 15 mengilustrasikan bagaimana fungsi penandaan digunakan. Keutamaan pemilihan pesan dengan struktur khusus diacu sebagai pemilihan pesan dengan *ridandensi* (redudancy).



Gambar 15: Skema penandaan dijitel dengan pemulihan pesan.

Modifikasi di atas lebih dari sekedar penyederhanaan; ini sangat penting jika diinginkan dipenuhinya Sifat (b) pada sifat-sifat penandaan dijitel di Halaman 38. Alasannya, B dapat memilih secara random $s \in \mathcal{S}$ sebagai tanda dan menerapkan transformasi E_e untuk mendapatkan $m = E_e(s)$; ini bisa dilakukan karena $\mathcal{S} = \mathcal{M}$ dan E_e diketahui oleh publik. Kemudian B mengirim bahwa pesan bertanda (m, s). Dengan mudah dilihat bahwa s akan diperiksa sebagai tanda yang diciptakan oleh A, padahal A tidak melakukannya. Dengan demikian B telah memalsu tanda dari A dan ini merupakan suatu contoh yang disebut *pemalsuan eksistensial* (existential forgery).

Chapter 7

Autentikasi

7.1 Pendahuluan

Sebelum masuk ke bahasan utama dalam bab ini, berikut ini diberikan pengertian tentang protokol dan mekanisme yang natinya akan banyak digunakan di dalam bahasan-bahasan berikutnya baik pada bab ini maupun bab selanjutnya.

7.1.1 Protokol dan Mekanisme

Definisi 7.1 ***Protokol kriptografik*** *adalah algoritme terdistribusi yang didefinisikan sebagai barisan langkah-langkah secara tepat menspesifikasikan tindakan–tindakan dari dua entitas atau lebih yang diperlukan untuk mencapai tujuan keamanan tertentu.*

Protokol memegang peranan penting di dalam kriptografi dan utamanya di dalam mempertemukan tujuan-tujuan kriptografi. Skema enkripsi, penandaan dijitel, fungsi hash, dan pembangkitan bilangan random merupakan primitif-primitif yang sering kali digunakan untuk membangun suatu protokol.

Contoh 7.1 *Ali dan Budi telah sepakat untuk menggunakan skema kunci simetrik dalam komunikasi melalui saluran tak-aman. Untuk mengenkripsi informasi mereka memerlukan suatu kunci. Protokol komunikasinya dapat dilakukan langkah-langkah berikut.*

1. *Budi mengkonstruksi suatu skema enkripsi kunci-publik dan mengirimkan kunci publiknya kepada Ali melalui saluran tak-aman yang telah disepakati.*

2. *Ali membuat suatu kunci untuk digunakan dalam skema kunci-simetrik yang telah disepakati.*

3. *Ali mengenkripsi kunci yang telah dibuat dalam Langkah 2 dengan menggunakan kunci publik milik Budi yang telah diterima pada Langkah 1, kemudian hasil enkripsinya dikirimkan ke Budi.*

4. *Dengan kunci pribadi yang dimilikinya, Budi mendekripsi pesan dari Ali pada Langkah 3 untuk mendapatkan kunci simetrik milik Ali.*

5. *Ali dan Budi mulai berkomunikasi secara rahasia dengan menggunakan sistem kunci simetrik sebagai kunci rahasia bersama.*

Protokol pada contoh di atas menggunakan fungsi-fungsi dasar untuk merealisasikan komunikasi rahasia pada saluran tak-aman. Perangkat dasarnya adalah skema enkripsi kunci simetrik dan kunci-publik. Protokol ini mempunyai kelemahan terhadap serangan pemalsuan (perhatikan Subbab 4.1.1), tetapi ide protokol bisa diterima.

Sebagai perbandingan untuk pengertian protokol, suatu *mekanisme* mempunyai pengertian lebih umum yang meliputi makna protokol itu sendiri, algoritme-algoritme yang menspesifikasikan langkah-langkah setiap entitas, dan teknik-teknik keamanan non-kriptografik (misalnya, perlindungan hardware dan kontrol prosedur) untuk mencapai tujuan keamanan tertentu.

Definisi 7.2 ***Kegagalan protokol atau mekanisme*** *terjadi ketika protokol atau mekanisme gagal mencapai tujuan yang dimaksud, dalam suatu cara dimana musuh mendapatkan keuntungan dengan tidak membobol primitif-primitif penyusunnya, tetapi dengan memanipulasi protokol atau mekanisme itu sendiri.*

Contoh 7.2 *Ali dan Budi berkomunikasi menggunakan sandi alir. Pesan yang mereka enkripsi diketahui mempunyai bentuk khusus:* 20 *bit pertama membawa informasi yang merepresentasikan jumlah uang. Musuh aktif dengan mudah dapat menerapkan operasi XOR suatu bitstring yang sesuai terhadap* 20 *bit pertama dari siferteks untuk mengubah jumlah uang. Dalam hal ini musuh tidak mengetahui pesan asli, tetapi ia dapat mengubah pesan. Ini menunjukkan protokol telah gagal karena integritas data tidak terlindungi.*

Contoh 7.3 *Transaksi bank secara elektronik mencatat semua nilai transaksi dalam domain* 32 *bit dienkripsi menggunakan skema kunci publik. Protokol sederhana ini ditujukan untuk memberikan layanan kerahasiaan nilai*

transaksi. Pertanyaannya, apakah tujuan tersebut dapat tercapai? Seorang musuh aktif dapat dengan mudah mengenkripsi semua 2^{32} plainteks dengan kunci publik yang tersedia. Kemudian, ia membandingkan hasilnya (2^{32} sifer-teks) dengan nilai transaksi yang terenkripsi, maka ia akan mendapatkan nilai transaksi yang sebenarnya. Pada kasus ini, musuh tidak membobol kunci pribadi, tetapi menggunakan cara yang lain atas dasar kelemahan protokol.

Beberapa alasan yang menyebabkan terjadinya kegagalan protokol atau mekanisme diataranya dinyatakan berikut ini.

1. Kelemahan-kelemahan dalam suatu primitif kriptografi penyusunnya.
2. Jaminan keamanan diasumsikan terlalu ketat, sehingga tidak bisa dimengerti secara jelas.
3. Kesalahan menerapkan beberapa prinsip ke sejumlah primitif seperti enkripsi.

Dalam mendisain protokol dan mekanisme dua langkah penting yang harus diperhatikan:

1. identifikasi semua asumsi yang digunakan.
2. untuk masing-masing asumsi, tentukan pengaruhnya terhadap tujuan keamanan jika asumsi itu dilanggar.

7.2 Pendefinisian Autentikasi

Autentikasi merupakan suatu alat yang digunakan untuk menjamin bahwa entitas atau informasi yang sah belum dimanipulasi oleh pihak-pihak yang tidak berwenang. Beberapa contoh tujuan keamanan autentikasi secara spesifik meliputi: kontrol akses, autentikasi entitas, autentikasi pesan, integritas data, non repudiasi, dan autentikasi kunci. Autentikasi merupakan salah satu bagian terpenting dari suluruh tujuan keamanan informasi. Secara garis besar autentikasi dikelompokkan menjadi dua, yaitu *autentikasi entitas* (identifikasi) dan *atentikasi asal data* (autentikasi pesan).

Definisi 7.3 *Dalam suatu transakasi yang melibatkan dua partai, teknik* ***identifikasi*** *atau* ***autentikasi entitas*** *menjamin agar pihak kedua meyakini (melalui bukti yang kuat) identitas pihak pertama, sementara itu pihak pertama aktif menciptakan bukti yang diperlukan.*

Contoh 7.4 *Misalkan A menelpon B. Jika mereka mengenali satu sama lain tanpa menyebutkan identitasnya, berarti autentikasi entitas telah terjadi dengan melalui pengenalan suara.*

Contoh 7.5 *Dalam transaksi bank melalui ATM, A memberikan nomor identitas personal (PIN) beserta kartu strip magnetik yang menyimpan informasi tentang A. Dalam hal ini, ATM menggunakan informasi yang ada pada kartu dan PIN untuk memverifikasi identitas A. Jika verifikasi berhasil, A diberikan akses berbagai layanan yang ditawarkan oleh bank.*

Definisi 7.4 *Dalam suatu transakasi yang melibatkan dua partai, teknik* ***autentikasi asal data*** *atau* ***autentikasi pesan*** *menjamin satu pihak yang menerima pesan meyakini (melalui bukti yang kuat) bahwa pesan benar-benar berasal dari identitas pihak yang mengirim pesan.*

Sering kali pesan yang dikirim ditambahkan informasi ekstra, sehingga pihak penerima dapat menentukan identitas dari entitas yang mengirim pesan.

Contoh 7.6 *A mengirim pesan email kepada B. Pesan melewati berbagai sistem komunikasi jaringan, dan disimpan oleh B untuk dibaca kemudian. B memerlukan suatu alat untuk memverifikasi pesan yang diterima bahwa benar-benar dibuat oleh A dan berasal dari A.*

Autentikasi asal data secara implisit memberikan integritas data karena. Alasannya, jika pesan telah dimanipulasi selama transmisi, maka pesan bukan lagi berasal dari A.

Skema umum, suatu identifikasi melibatkan *penuntut* (claimant) A dan *pemeriksa* (verifier) B. Pemerika dihadirkan dengan praduga sebelumnya bahwa penuntut mengakui identitas berasal darinya. Tujuannya adalah untuk membuktikan bahwa identitas benar-benar dari A.

Perbedaan utama antara autentikasi entitas dan autentikasi pesan adalah autentikasi pesan tidak memberikan jaminan rentang waktu diciptanya pesan, sedangkan autentikasi entitas mencakup pembuktian identitas penuntut melalui komunikasi nyata oleh pemeriksa terkait, selama eksekusi protokol di dalam waktu nyata (real time). Perbedaan lainnya adalah bahwa autentikasi pesan melibatkan pesan yang mempunyai makna, sedangkan autenkasi entitas hanya sekedar membuktikan entitas tanpa melibatkan pesan.

Dalam konteks komunikasi jaringan, autentikasi pesan mampu mencegah serangan-serangan berikut:

1. *Penyingkapan* (disclosure): Penyingkapan isi pesan atau kunci kriptografik kepada seseorang.

2. *Analisis lalulintas* (traffic analysis): Pengungkapan pola lalulintas oleh partai-partai yang terlibat dalam transaksi informasi.

3. *Penyamaran* (masquerade): Penyisipan pesan ke dalam suatu jaringan dari suatu sumber yang akan berbuat curang.

4. *Modifikasi isi* (content modification): Manipulasi isi pesan oleh pihak yang tidak berwenang.

5. *Modifikasi urutan* (sequence modification): Mengubah urutan pesan oleh pihak yang tidak berwenang.

6. *Repudiasi sumber* (source repudiation): Pengingkaran transmisi pesan oleh pengirim.

7. *Repudiasi tujuan* (destinartion repudiation): Pengingkaran penerimaan pesan oleh penerima.

7.3 Integritas Data

Definisi 7.5 *Integritas data adalah suatu sifat dimana data belum dimanipulasi (diganti, disisipi, dihapus, diubah urutannya, dll) dengan suatu cara yang tidak sah oleh pihak-pihak yang tidak berwenang, sejak data itu dibuat, dikirim, atau disimpan.*

Verifikasi dari suatu integritas data memerlukan pendefinisian subhimpunan yang beranggotakan satuan-satuan data yang memenuhi kriteria khusus untuk diterima, dan dibedakan dari kriteria data yang ditolak. Suatu contoh cara pengenalan di dalam integritas data adalah dengan melibatkan konsep ridandensi dengan format tertentu. Teknik kriptografi untuk integritas data bertumpu pada informasi rahasia dan saluran autentik.

Penggunaan kunci rahasia bukan suatu yang utama di dalam integritas data. Cara ini bisa digantikan teknik *hashing* (menggunakan primitif fungsi hash), dan perlindungan autentisitas hashnya digunakan saluran autentik (tidak harus rahasia). Dalam hal ini, prosedur untuk pengirim:

1. pengirim menghitung kode hash dari data pesan dengan menggunakan MDC,

2. mengirim data pesan melalui saluran tak-aman, dan

3. mengirim kode hash melalui saluran yang terpisah dari saluran data pesan, dan saluran ini harus autentik untuk memberikan jaminan autentikasi asal data. Saluran autentik bisa berupa telepon publik melalui pengenalan suara, media penyimpan data (seperti *floppy disk*, kertas, dll) di dalam suatu tempat terpercaya (terkunci rapat), atau media publik yang sulit dipalsukan (seperti surat kabar harian, majalah, televisi, dll).

Prosedur untuk penerima:

1. menghitung nilai hash dari data pesan yang diterima,

2. membandingkan nilai hash hasil hitungannya dengan nilai hash yang kode hash yang diterima, dan

3. data pesan diterima sah (mempunyai integritas) apabila kedua nilai hash sama, menolak (tidak mempunyai integritas) apabila tidak sama.

Metode integritas data berbasis pada MDC diskemakan pada Gambar 17.

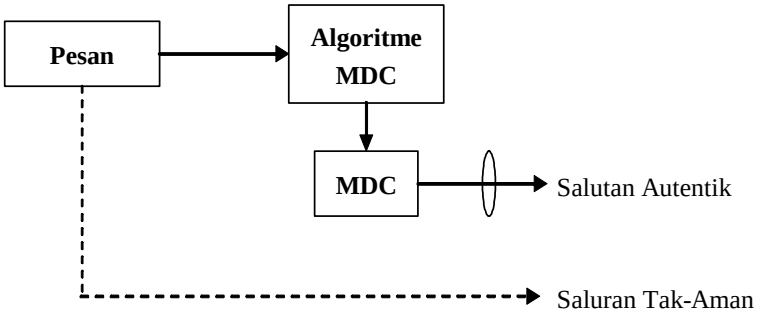
Beberapa contoh penerapan yang terkait dengan integritas data:

1. proteksi software dari virus,

2. distribusi sotfware atau kunci publik melalui jaringan tidak terpercaya,

3. pemeriksaan virus dari sumberdaya komputer atau kode kode, dll.

Suatu contoh umum penggunaan MDC dikombinasikan dengan saluran autentik untuk memberikan integritas data adalah skema penandaan dijitel (seperti RSA).



Gambar 16: MDC dan Saluran Autentik

7.4 Autentikasi Pesan

Teknik autentikasi pesan melibatkan fungsi autentikasi. Fungsi autentikasi memetakan suatu pesan plainteks ke suatu nilai yang disebut *autentikator* (authenticator). Dua kelompok besar yang biasanya digunakan sebagai fungsi autentikasi adalah *fungsi enkripsi* dan *fungsi hash*.

7.4.1 Enkripsi Sebagai Fungsi Autentikasi

Autentikasi dengan Enkripsi Kunci Simetrik

Perhatikan lagi skema enkripsi kunci simetrik yang diberikan pada Gambar 3 di Bab 2, kerahasiaan pasangan kunci (e, d) mengakibatkan Budi sangat yakin bahwa plainteks $\mathbf{m}$ (sebagai hasil transformasi dekripsi $D_d(\mathbf{c}) = \mathbf{m}$) benar-benar milik dan berasal dari Ali. Alasannya, hanya Ali pemegang kunci e dan mampu mengenkripsi $\mathbf{m}$ menjadi $\mathbf{c} = E_e(\mathbf{m})$. Dalam skema ini integritas $\mathbf{m}$ juga dijamin karena pihak ketiga tidak akan mampu mengubah $\mathbf{m}$ tanpa pengetahuan akan pasangan kunci (e, d).

Dari paragraf di atas dapat disimpulkan bahwa enkripsi kunci simetrik memberikan layanan *kerahasiaan* sekaligus *autentikasi*. Walaupun kesimpulan ini sah, namun dari pengertian autentikasi sendiri masih perlu dipertegas agar tidak terjadi kesalahan intepretasi. Perhatikan paragraf berikut ini.

Di dalam komunikasi di atas, E_e bertindak sebagai fungsi autentikasi, sedangkan D_d bertindak sebagai *fungsi verifikasi*. Input dari D_d adalah sembarang anggota dari ruang siferteks $\mathcal{C}$. Jika diberikan sembarang $\mathbf{c}' \in \mathcal{C}$

dan $D_d(\mathbf{c}') = \mathbf{m}'$, masih belum jelas bagaimana Budi memutuskan (memverifikasi) apakah $\mathbf{m}'$ plainteks (sah) yang dikirim oleh Ali atau yang berasal dari pihak ketiga. Untuk mengatasi masalah hal ini diperlukan definisi subhimpunan $\mathcal{M}'$ dari ruang plainteks $\mathcal{M}$ yang mempunyai struktur (pola) tententu, mempunyai makna, dan $|\mathcal{M}'| \ll |\mathcal{M}|$ (ingat kembali bahasan penandaan dijitel dengan pemulihan pesan). Anggota dari $\mathcal{M}'$ disebut *plainteks sah* yang harus dipilih oleh Ali.

Contoh 7.7 *Pada sandi Caesar, $\mathcal{M}'$ didefinisikan sebagai himpunan semua teks yang mempunyai struktur dan makna bahasa Indonesia, dan dipilih kunci $e = 1$. Jika pesan yang diterima Budi:*

$$ZBPHNBIBLVBTB,$$

diverifikasi menjadi

$$YANGMAHAKUASA,$$

*dan disimpulkan sebagai **plainteks sah**. Jika pesan yang diterima:*

$$GKUDFFRTPABNH,$$

diverifikasi menjadi

$$FJTCDDQSNZAMG,$$

*dan disimpulkan sebagai **plainteks tidak sah**.*

Jika plainteks, misalnya file biner atau sinar-X dijitel, untuk mendefinisikan $\mathcal{M}'$ biasanya sangat sulit. Salah satu cara untuk menyelesaikan problem ini adalah dengan membuat plainteks agar mempunyai struktur yang mudah dikenal tetapi, tidak dapat direplikasi tanpa bantuan fungsi enkripsi. Sebagai misal, pada setiap plainteks dibubuhkan *kode deteksi galat* (KDG) atau *checksum* sebelum dienkripsi, diilustrasikan pada Gambar 17, dan dirinci pada prosedur berikut ini.

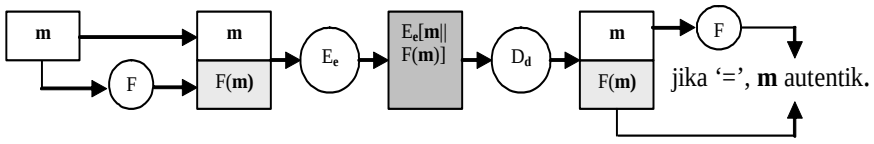
Protokol Autentikasi Dengan Enkripsi Kunci Simetrik:
Prosedur Untuk Ali

1. Ambil $\mathbf{m} \in \mathcal{M}$.
2. Definisikan fungsi checksum $F : \mathbf{m} \mapsto KDG$.
3. Hitung $F(\mathbf{m}) = KDG$.
4. Bubuhkan KDG pada $\mathbf{m}$, yaitu $\mathbf{m} \| KDG$.

5. Enkripsi dengan kunci rahasia e, yaitu $E_e[\mathbf{m}\|KDG] = \mathbf{c}$.
6. Kirim $\mathbf{c}$ ke Budi.

Prosedur Verifikasi oleh Budi

1. Dekripsi $\mathbf{c}$ dengan kunci d, yaitu: $D_d(\mathbf{c}) = \mathbf{m}\|KDG$
2. Hitung $F(\mathbf{m})$
3. Bandingkan $F(\mathbf{m})$ dan KDG.
4. $\mathbf{m}$ adalah plainteks sah apabila $F(\mathbf{m}) = KDG$.



Gambar 17: Skema autentikasi pesan dengan enkripsi kunci simetrik.

Autentikasi Dengan Enkripsi Kunci Publik

Sebagaimana telah dinyatakan pada Bab 4 bahwa skema pada Gambar 13 memberikan kerahasiaan tetapi tidak untuk autentikasi. Dalam hal ini, karena kunci yang digunakan untuk mengenkripsi adalah adalah publik, maka tidak ada jaminan bagi Budi bahwa pesan benar-benar milik dan berasal dari Ali. Dengan kunci publik siapa saja bisa mengirimkan pesan ke Budi. Agar konsep kunci publik memberikan autentikasi prosedur berikut ini bisa digunakan (Gambar 11.1c).

Protokol Autentikasi Dengan Enkripsi Kunci Publik:
Prosedur Untuk Ali

1. Ciptakan kunci pribadi KR_A dan pasangannya kunci publik KU_A.
2. Kirim KU_A ke Budi.
3. Ambil $\mathbf{m} \in \mathcal{M}$.

4. Enkripsi **m** dengan kunci pribadi KR_A, yaitu: $E_{KR_A}(\mathbf{m}) = \mathbf{c}$.
5. Kirim **c** ke Budi.

Prosedur Untuk Budi

1. Dekripsi **c** dengan kunci KU_A, yaitu: $D_{KU_A}(\mathbf{c}) = \mathbf{m}$

Perhatikan bahwa skema di atas memberikan autentikasi tetapi tidak untuk kerahasiaan. Karena hanya Ali seorang yang mempunyai KR_A, maka Budi akan yakin bahwa pesan benar-benar milik dan berasal dari Ali. Kerahasiaan tidak dijamin karena siapapun bisa mendapatkan **m** dari dekripsi dengan kunci publik KU_A. Agar skema ini sekaligus bisa memberikan kerahasiaan, modifikasi bisa kita lakukan seperti pada Gambar 11.1d.

7.4.2 Fungsi Hash Sebagai Fungsi Autentikasi

Autentikasi Dengan MAC

Berikut ini diberikan prosedur autentikasi dengan MAC yang hanya memberikan autentikasi, tidak untuk kerahasiaan (Gambar 11.4a).

Protokol Autentikasi Dengan MAC:
Prosedur Untuk Ali

1. Ciptakan kunci K dan kirim salinannya ke Budi.
2. Ambil $\mathbf{m} \in \mathcal{M}$.
3. Hitung nilai MAC dengan kunci K, yaitu $h_K(\mathbf{m}) = MAC$.
4. Bubuhkan MAC pada **m**, yaitu $\mathbf{m} \| MAC = \mathbf{c}$.
5. Kirim **c** ke Budi.

Prosedur Verifikasi oleh Budi

1. Diterima: $\mathbf{c} = \mathbf{m} \| MAC$ dan kunci K.
2. Hitung $h_K(\mathbf{m})$
3. Bandingkan $h_K(\mathbf{m})$ dan MAC.

4. $\mathbf{m}$ adalah plainteks sah apabila $h_K(\mathbf{m}) = MAC$.

Pada skema ini pesan $\mathbf{c}$ berbentuk $\mathbf{m}\|MAC$. Ini berarti $\mathbf{m}$ diketahui publik sehingga tidak memberikan kerahasiaan. Agar skema ini sekaligus bisa memberikan kerahasiaan, modifikasi bisa kita lakukan seperti pada Gambar 11.4b.

Protokol Autentikasi Dengan MAC dan Enkripsi Simetrik:
Prosedur Untuk Ali

1. Ciptakan kunci MAC K, kunci enkripsi e dan kirim secara rahasia salinannya ke Budi.
2. Ambil $\mathbf{m} \in \mathcal{M}$.
3. Hitung nilai MAC dengan kunci K, yaitu $h_K(\mathbf{m}) = MAC$.
4. Bubuhkan MAC pada $\mathbf{m}$, yaitu $\mathbf{m}\|MAC$
5. Enkripsi $\mathbf{m}\|MAC$ dengan kunci e, yaitu $E_e(\mathbf{m}\|MAC) = \mathbf{c}$
6. Kirim $\mathbf{c}$ ke Budi.

Prosedur Verifikasi oleh Budi

1. Diterima: pesan $\mathbf{c}$, kunci K, dan e.
2. Hitung kunci dekripsi d dari e.
3. Dekripsi $D_d(\mathbf{c}) = \mathbf{m}\|MAC$.
4. Hitung $h_K(\mathbf{m})$.
5. Bandingkan $h_K(\mathbf{m})$ dan MAC.
6. $\mathbf{m}$ adalah plainteks sah apabila $h_K(\mathbf{m}) = MAC$.

Autentikasi Dengan MDC

Sesuai dengan namanya, MDC mempunyai kemampuan deteksi galat: pengubahan bit pada pesan input akan mengubah nilai (output) hash. Ini berarti MDC memberikan integritas pesan. Karena MDC tidak menggunakan kunci, skema autentikasi yang menggunakan MDC biasanya dikombinasikan dengan fungsi enkripsi untuk melindungi nilai MDC atau untuk tujuan kerahasiaan. Berikut diberikan beberapa model skema autentikasi dengan MDC (Gambar 11.5).

Protokol-1: Autentikasi Dengan MDC dan Enkripsi Simetrik:
Prosedur Untuk Ali

1. Ciptakan kunci enkripsi e dan kirim secara rahasia salinannya ke Budi.
2. Ambil $\mathbf{m} \in \mathcal{M}$.
3. Hitung nilai MDC yaitu $h(\mathbf{m}) = MDC$.
4. Bubuhkan MDC pada $\mathbf{m}$, yaitu $\mathbf{m} \| MDC$
5. Enkripsi $\mathbf{m} \| MDC$ dengan kunci e, yaitu $E_e(\mathbf{m} \| MDC) = \mathbf{c}$
6. Kirim $\mathbf{c}$ ke Budi.

Prosedur Verifikasi oleh Budi

1. Diterima: pesan $\mathbf{c}$, dan kunci e.
2. Hitung kunci dekripsi d dari e.
3. Dekripsi $D_d(\mathbf{c}) = \mathbf{m} \| MDC$.
4. Hitung $h(\mathbf{m})$.
5. Bandingkan $h(\mathbf{m})$ dan MDC.
6. $\mathbf{m}$ adalah plainteks sah apabila $h(\mathbf{m}) = MDC$.

Pada skema ini tujuan autentikasi dan kerahasiaan tercapai, dan dua skema berikut ini hanya memberikan autentikasi pada pesan.

Protokol-2: Autentikasi Dengan MDC dan Enkripsi Simetrik:
Prosedur Untuk Ali

1. Ciptakan kunci enkripsi e dan kirim secara rahasia salinannya ke Budi.
2. Ambil $\mathbf{m} \in \mathcal{M}$.
3. Hitung nilai MDC yaitu $h(\mathbf{m}) = MDC$.
4. Enkripsi MDC dengan kunci e, yaitu $E_e(MDC)$
5. Bubuhkan $E_e(MDC)$ pada $\mathbf{m}$, yaitu $\mathbf{m} \| E_e(MDC) = \mathbf{c}$
6. Kirim $\mathbf{c}$ ke Budi.

Prosedur Verifikasi oleh Budi

1. Diterima: pesan $\mathbf{c} = \mathbf{m} \| E_e(MDC)$, dan kunci e.
2. Hitung $h(\mathbf{m})$.
3. Hitung kunci dekripsi d dari e.
4. Dekripsi $D_d[E_e(MDC)] = MDC$.
5. Bandingkan $h(\mathbf{m})$ dan MDC.
6. $\mathbf{m}$ adalah plainteks sah apabila $h(\mathbf{m}) = MDC$.

Protokol-3: Autentikasi Dengan MDC dan Enkripsi Publik:
Prosedur Untuk Ali

1. Ciptakan kunci pribadi KR_A dan kunci publik KU_A.
2. Kirim KU_A ke Budi melalui saluran publik
3. Ambil $\mathbf{m} \in \mathcal{M}$.
4. Hitung nilai MDC yaitu $h(\mathbf{m}) = MDC$.
5. Enkripsi MDC dengan kunci KR_A, yaitu $E_{KR_A}(MDC)$
6. Bubuhkan $E_{KR_A}(MDC)$ pada $\mathbf{m}$, yaitu $\mathbf{m} \| E_{KR_A}(MDC) = \mathbf{c}$
7. Kirim $\mathbf{c}$ ke Budi.

Prosedur Verifikasi oleh Budi

1. Diterima: pesan $\mathbf{c} = \mathbf{m} \| E_{KR_A}(MDC)$, dan kunci KU_A.
2. Hitung $h(\mathbf{m})$.
3. Dekripsi $D_{KU_A}[E_{KR_A}(MDC)] = MDC$.
4. Bandingkan $h(\mathbf{m})$ dan MDC.
5. $\mathbf{m}$ adalah plainteks sah apabila $h(\mathbf{m}) = MDC$.

7.5 Autentikasi Entitas (Identifikasi)

Skema umum, suatu identifikasi melibatkan *penuntut* (claimant) A dan *pemeriksa* (verifier) B. Pemerika dihadirkan dengan praduga sebelumnya bahwa penuntut mengakui identitas berasal darinya. Tujuannya adalah untuk membuktikan bahwa identitas benar-benar dari A.

Perbedaan utama antara autentikasi entitas dan autentikasi pesan adalah autentikasi pesan tidak memberikan jaminan rentang waktu diciptanya pesan, sedangkan autentikasi entitas mencakup pembuktian identitas penuntut melalui komunikasi nyata oleh pemeriksa terkait, selama eksekusi protokol di dalam waktu nyata (real time). Perbedaan lainnya adalah bahwa autentikasi pesan melibatkan pesan yang mempunyai makna, sedangkan autenkasi entitas hanya sekedar membuktikan entitas tanpa melibatkan pesan.

Tujuan Protokol Identifikasi

Dalam pandangan pemeriksa, produk dari protokol identifikasi adalah diterimanya identitas penuntut secara sah atau protokol berakhir dengan menolak identitas penuntut. Rebih rincinya tujuan protokol identifikasi diantaranya meliputi berikut ini:

1. Dalam hal penuntut A dan pemeriksa B saling jujur, A dapat mengotekasi dirinya ke B.
2. B tidak bisa memalsukan identitas A (menggunakan kembali) untuk pihak ketiga C.
3. Sangat kecil peluang bahwa pihak ketiga C berpura-pura sebagai A untuk mengelabui B.

Basis Identifikasi

Basis identifikasi meliputi:

Chapter 8

Establismen, Manajemen, dan Sertifikasi Kunci

Pada bab ini akan dibahas metodologi dasar untuk menjamin keamanan distribusi kunci untuk tujuan kriptografi. $\longrightarrow$Ongoing Work!!!

8.1 Konsep Dasar

Definisi 8.1 ***Establismen kunci*** *adalah suatu proses atau protokol yang berkenaan dengan tersedianya kunci bersama bagi dua entitas atau lebih untuk tujuan kriptografi.*

Definisi 8.2 ***Manajemen kunci*** *adalah serangkaian proses dan mekanisme yang mendukung establismen kunci dan menjaga hubungan penggunaan kunci diantara entitas, termasuk penggatian kunci lama dan kunci baru jika diperlukan.*

Secara garis besar establismen kunci dibagi menjadi dua: *persetujuan kunci* dan *transport kunci.*

Establismen dan manajemen kunci merupakan suatu teknik yang bisa digunakan untuk menangani suatu problem seperti diilustrasikan berikut ini. Misalkan suatu network yang beranggotakan n entitas ingin berkomunikasi dengan menggunakan teknik kunci simetrik, maka banyaknya kunci rahasia yang dibutuhkan agar setiap pasang entias mampu berkomunikasi secara adalah $\binom{n}{2} = \frac{n(n-1)}{n}$. Di dalam pratik, jumlah entitas di dalam suatu network biasanya cukup besar, akibatnya jumlah kunci rahasia yang diperlukan sesuai dengan rumusan di atas menjadi sangat besar. Telah banyak metode man-

ajemen kunci yang bisa digunakan untuk menyelesaikan problem tersebut. Sebagai gambaran, berikut ini diberikan dua metode sederhana.

Bibliography

[1] R. P. Menezes, P. C. van Oorschot, and S. Vanstone, "Handb1ook of Applied Cryptography," *CRC Press, Inc.*, 1997.

[2] W. Stallings, "Cryptography and Network Security," *Prentice Hall.*, 2003, ISBN: 0-13-11502-2.

[3] D. R. Stinson, "Cryptography: Theory and Practice," *CRC Press, Inc.*, 1995, ISBN: 0-8493-8521-0.