BAB II

TINJAUAN PUSTAKA

2.1 Algoritma

Defenisi Algoritma, *"Algoritma adalah urutan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis"*. Sedangkan menurut Kamus Besar Bahasa Indonesia, terbitan Balai Pustaka, 1988. "*Algoritma adalah urutan logis pengambilan putusan untuk pemecahan masalah"*.

2.2 Kriptografi (Cryptography)

Kriptografi berasal dari Bahasa Yunani : "*cryptos*" artinya "*secret*" (rahasia), sedangkan "*graphein"* artinya "*writing"* (tulisan). Jadi, kriptografi berarti "*secret writing"* (tulisan rahasia). Definisi yang dipakai di dalam buku-buku yang lama (sebelum 1980-an) menyatakan bahwa kriptografi adalah ilmu dan seni untuk menjaga kerahasiaan pesan dengan cara menyandikanya ke dalam bentuk yang tidak dapat dimengerti lagi maknanya. Saat ini kriptografi lebih dari sekedar *privacy*, tetapi juga untuk tujuan *confidelity, data integrity*, *autentication* dan *non-repudation.*

1. *Confidelity* (kerahasiaan) yaitu layanan agar isi pesan yang dikirimkan tetap rahasia dan tidak diketahui oleh pihak lain (kecuali pihak pengirim, pihak penerima / pihak-pihak memiliki ijin). Umumnya hal ini dilakukan dengan cara membuat suatu algoritma matematis yang mampu mengubah data hingga menjadi sulit untuk dibaca dan dipahami.

2. *Data integrity* (keutuhan data) yaitu layanan yang mampu mengenali/mendeteksi adanya manipulasi (penghapusan, pengubahan atau penambahan) data yang tidak sah (oleh pihak lain).
3. *Authentication* (keotentikan) yaitu layanan yang berhubungan dengan identifikasi. Baik otentikasi pihak-pihak yang terlibat dalam pengiriman data maupun otentikasi keaslian data/informasi.
4. *Non-repudiation* (anti-penyangkalan) yaitu layanan yang dapat mencegah suatu pihak untuk menyangkal aksi yang dilakukan sebelumnya (menyangkal bahwa pesan tersebut berasal darinya).

Dalam penelitian ini definisi yang digunakan sebagai pengertian kriptografi :

Kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan (*cryptography is the art and science of keeping message secure* (SCH96))

Kata "seni" di dalam definisi di atas berasal dari fakta sejarah bahwa pada masa-masa awal sejarah kriptografi, setiap orang mungkin mempunyai cara yang unik untuk merahasiakan pesan. Cara-cara unik tersebut mungkin berbeda-beda pada setiap pelaku kriptografi sehingga setiap cara menulis pesan rahasia pesan mempunai nilai estetika tersendiri sehingga kriptografi berkembang menjadi sebuah seni merahasiakan pesan(kata "*graphy*" di dalam "*cryptography"* itu sendiri sudah menyiratkan sebuah seni).

2.3 Algoritma kriptografi

Algoritma kriptografi atau sering disebut dengan *cipher* adalah suatu fungsi matematis yang digunakan untuk melakukan enkripsi dan dekripsi (Schneier, 1996).

Ada dua macam algoritma kriptografi, yaitu *algoritma simetris* (*symmetric algorithms*) dan *algoritma asimetris* (*asymmetric algorithms*).

Konsep kriptografi sendiri telah lama digunakan oleh manusia misalnya pada peradaban Mesir dan Romawi walau masih sangat sederhana. Prinsip-prinsip yang mendasari kriptografi yakni:

Berbeda dengan kriptografi klasik yang menitikberatkan kekuatan pada kerahasiaan algoritma yang digunakan (yang artinya apabila algoritma yang digunakan telah diketahui maka pesan sudah jelas "bocor" dan dapat diketahui isinya oleh siapa saja yang mengetahui algoritma tersebut), kriptografi modern lebih menitikberatkan pada kerahasiaan kunci yang digunakan pada algoritma tersebut (oleh pemakainya) sehingga algoritma tersebut dapat saja disebarkan ke kalangan masyarakat tanpa takut kehilangan kerahasiaan bagi para pemakainya.

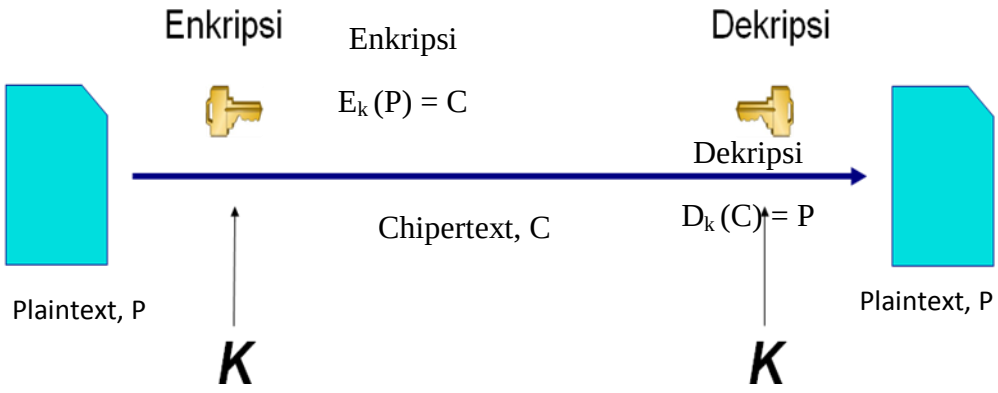
Berikut adalah terminologi yang digunakan dalam bidang kriptografi :

1. **Plaintext** adalah pesan yang hendak dikirimkan (berisi data asli).
2. **Ciphertext** adalah pesan ter-enkrip (tersandi) yang merupakan hasil enkripsi.
3. **Enkripsi** adalah proses pengubahan plaintext menjadi ciphertext.
4. **Dekripsi** adalah kebalikan dari enkripsi yakni mengubah ciphertext menjadi plaintext, sehingga berupa data awal/asli.
5. **Kunci** adalah suatu bilangan yang dirahasiakan yang digunakan dalam proses enkripsi dan dekripsi.

Dalam kriptografi terdapat dua konsep utama yakni enkripsi dan dekripsi. Enkripsi adalah proses dimana informasi/data yang hendak dikirim diubah menjadi bentuk yang hampir tidak dikenali sebagai informasi awalnya dengan

menggunakan algoritma tertentu. Dekripsi adalah kebalikan dari enkripsi yaitu mengubah kembali bentuk tersamar tersebut menjadi informasi awal.

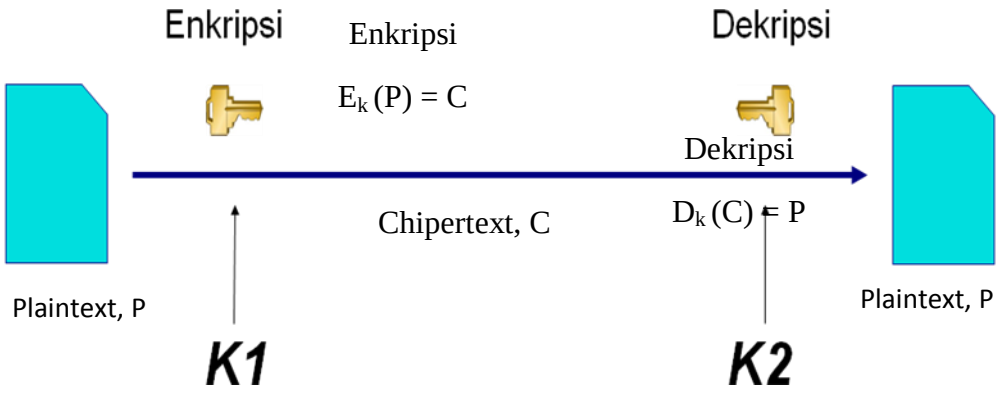
Algoritma kriptografi berdasarkan jenis kunci yang digunakan dapat dibedakan menjadi dua jenis yaitu :



Gambar 2.1 Skema kriptografi Simetri

a. Algoritma simetri

Dimana kunci yang digunakan untuk proses enkripsi dan dekripsi adalah kunci yang sama(**K**).



Gambar 2.2 Skema kriptografi asimetri

b. Algoritma *asimetris*

Dimana kunci yang digunakan untuk proses enkripsi (**K1**) atau sering disebut dengan *public key* dan dekripsi (**K2**) atau sering disebut dengan *private key* menggunakan kunci yang berbeda.

Sedangkan berdasarkan besar data yang diolah dalam satu kali proses, maka algoritma kriptografi dapat dibedakan menjadi dua jenis yaitu :

a. Algoritma *block cipher*

Informasi/data yang hendak dikirim dalam bentuk blok-blok besar (misal 64-bit) dimana blok-blok ini dioperasikan dengan fungsi enkripsi yang sama dan akan menghasilkan informasi rahasia dalam blok-blok yang berukuran sama.

b. Algoritma *stream cipher*

Informasi/data yang hendak dikirim dioperasikan dalam bentuk blok-blok yang lebih kecil (*byte* atau *bit*), biasanya satu karakter persatuan waktu proses, menggunakan tranformasi enkripsi yang berubah setiap waktu.

2.4 Serangan terhadap kriptografi

2.4.1 Kriptanalisis

Kriptanalisis bertujuan untuk memecahkan chiperteks menjadi plainteks semula tanpa memiliki akses ke kunci yang digunakan. Kriptanalisis berusaha menemukan kelemahan dari sistem kriptografi yang pada akhirnya mengarah untuk menemukan kunci dan mengungkap plainteks.

Dalam membuat serangan terhadap kriptografi, kita selalu mengasumsikan kriptanalisis mengetahui algoritma kriptografi yang digunakan, sehingga satu-satunya keamanan sistem kriptografi terletak sepenuhnya pada kunci. Hal ini didasarkan pada *prinsip kerckhoff(1883)* yang berbunyi :

"*semua algoritma kriptografi harus publik;hanya kunci yang rahasia"*

2.4.2 Keamanan algoritma kriptografi

Lars Knudsen mengelompokan hasil kriptanalisis ke dalam beberapa kategori berdasarkan jumlah dan kualitas informasi yang berhasil ditemukan.

1. Pemecahan total *(total berak)*. Kriptanalis menemukan kunci K sedemikian sehingga dekripsi D_k (C) = P.
2. Deduksi (penarikan kesimpulan) global *(global deduction)* kriptanalis menemukan algoritma alternatif, A, yang ekivalen dengan D_k(C) tetapi tidak mengetahui kunci K.
3. Deduksi lokal *(instace/local deduction)*. Kriptanalis menemukan plainteks yang disadap.
4. Deduksi informasi *(information deduction)*. Kriptanalis menemukan bebarapa inforamsi perihal kunci atau plainteks. Misalnya kriptanalis mengetahui beberapa bit kunci, kriptanalis mengetahui bahasa yang digunakan untuk menulis plainteks, kriptanalis mengetahui format plainteks dan sebagainya.

Sebuah algoritma kriptografi dikatakan aman mutlak tanpa syarat *(uncoditionally secure)* bila cipherteks yang dihasilkan oleh algoritma tersebut tidak mengandung cukup informasi untuk menentukan plainteksnya.

Sebaliknya, kriptografi dikatakan aman secara komputasi *(computationally secure)* bila memenuhi dua kriteria sebagai berikut :

1. Biaya untuk memecahkan cipherteks melampaui nilai inforamsi yang terkandung di dalam cipherteks tersebut.
2. Waktu yang diperlukan untuk memecahkan cipherteks melampaui lamanya waktu inforamsi tersebut harus dijaga kerahasiaanya.

2.4.3 Kompleksitas serangan

Kompleksitas serangan dapat diukur dengan beberapa cara, antara lain kompleksitas data, kompleksitas waktu dan kompleksitas memori.

2.4.3.1 Kompleksitas data *(data complexity)*

Jumlah data (plainteks dan cipherteks) yang dibutuhkan sebagai masukan untuk serangan. Semakin banyak data yang dibutuhkan untuk melakukan serangan, semakin kompleks serangan tersebut yang berarti semakin bagus sistem kriptografi tersebut.

2.4.3.2 Kompleksitas waktu *(time complexity)*

Waktu yang dibutuhkan untuk melakukan serangan. Semakin lama waktu yang dibutuhkan untuk melakukan serangan, berarti semakin bagus sistem kriptografi tersebut.

2.4.3.3 Kompleksitas ruang memori *(space/storage complexity)*

Jumlah memori yang dibutuhkan untuk melakukan serangan. Semakin banyak memori yang dibutuhkan untuk melakukan serangan, berarti semakin bagus sistem kriptografi tersebut.

2.4.4 Jenis-jenis serangan

Serangan (atau "serangan kriptanalsisi") terhadap sistem kriptografi dapat dikelompokan dengan beberapa cara.

2.4.4.1 **Berdasarkan keterlibatan penyerang dalam komunikasi**,

serangan dapat dibagi atas dua macam, yaitu serangan pasif dan serangan aktif :

1. **Serangan pasif *(passive attack)***

Pada jenis ini, penyerang tidak terlibat dalam komunikasiantara pengirim dan penerima, namun penyerang menyadap semua pertukaran pesan anteara kedua entitas tersebut. Tujuanya adalah untuk mendapatkan sebanyak mungkin inforamsi yang digunakan untuk kriptanalisis. Beberapa metode penyadapan data antara lain :

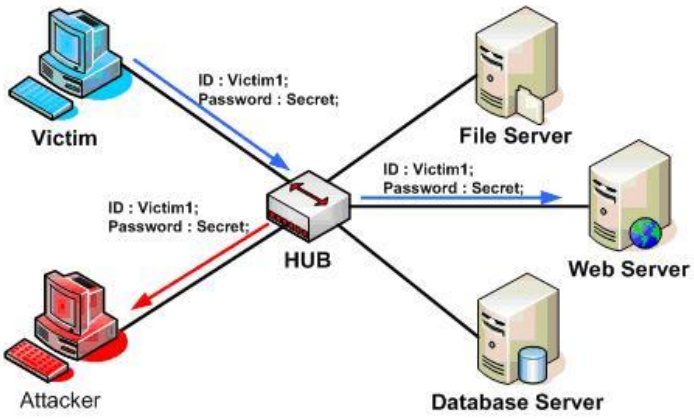
1.1 ***wiretrapping*** *:* penyadap mencegat data yang ditansmisikan pada saluran kabel komunikasi dengan menggunakan sambungan perangkat keras,

1.2 ***electromagnetic eavesdripping*** *:* penyadap mencegat data yang ditransmisikan melalui saluran *wireless*, misalnya radio dan *microwave,*

1.3 ***acoustic eavesdropping*** *:* menangkap gelombang suarayang dihasilkan oleh suara manusia.

2. **Serangan akrif*(active attack)***

Pada jenis serangan ini, penyerang mengintervensi komunikasi dan ikut mempengaruhi sistem untuk keuntungan dirinya. Misalnya penyerang mengubah aliran pesan seperti menghapus sebagian cipherteks, mengubah cipherteks menyisipkan potongan cipherteks palsu, me-*replay* pesan lama, mengubah inforamsi yang tersimpan dan sebagainya.



Gambar 2.3 Man-in-the-middle attack

Serangan yang termasuk serangan aktif adalah *man-in-the-middle attack*. Pada serangan ini, penyerang mengintersepsi komunikasi antara dua pihak yang berkomunikasi kemudian "menyerupai" salah satu pihak dengan cara bersikap seolah-olah ia adalah salah satu pihak yang berkomunikasi (pihak lainya tidak menyadari kalau ia berkomunikasi dengan pihak yang salah). Tujuan dari serangan ini adalah untuk mendapatkan inforamasi berharga seperti kunci atau nilai rahasia lainya. Caranya, penyerang memutus komunikasi antara dua pihak lalu menempatkan dirinya di antara keduanya.

2.4.4.2 Berdasarkan banyaknya inforamsi yang diketahui oleh kriptanalis, maka serangan dapat dikelompokan menjadi 5 jenis : *chipertext-only attack, known-plaintext attack, chosen-plaintext attack, chosen-chipertext attack dan chosen-text attack.*

1. *Chipertext-only attack*

Kriptanalisis menemukan plainteks sebanyak mungkin dari cipherteks atau menemukan kunci yang digunakan untuk mendekripsi. Secara formal hal ini diformulasikan sebagai berikut :

Diberikan : $C_1 = E_k(P_1, C_2 = E_k(P_2),...,C_i = E_k(P_i)$

Deduksi : $P_1, P_2,....,P_i$ atau k untuk mendapatkan P_{i+1} dari C_{i+1}

Untuk mendeduksi plainteks dari cipherteks, kriptanalis mungkin menggunakan beberapa cara, seperti mencoba semua kemungkinan kunci secara *exhaustive search*, menggunakan teknik analisis frekuensi membuat terkaan berdasarkan informasi yang diketahui.

2. Known-Plaintext attack

Jenis serangan dimana kriptanalisis memiliki pasangan plainteks dan cipherteks yang berkoresponden. Secara formal hal ini diformulasikan sebagai berikut :

Diberikan : $P_1, C_1, = E_k(P_1);P_2,C_2 = E_k(P_2);..., P_i, C_i = E_k(P_i)$

Deduksi : k untuk mendapatkan P_{i+1} dari $C_{i+1} = E_k(P_{i+1})$.

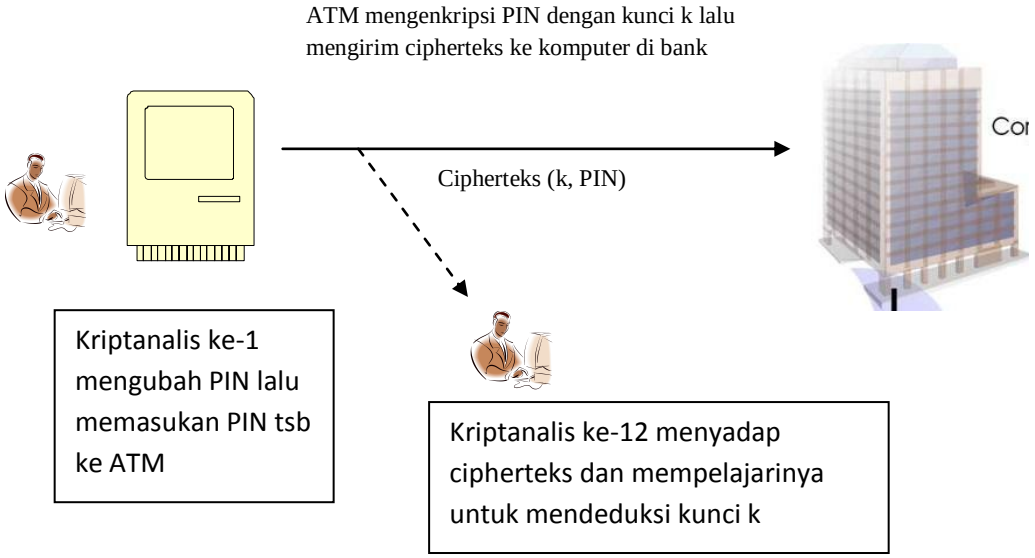
Beberapa pesan yang formatnya terstruktur membuka peluang kepada kriptanalis untuk menerka plainteks dari cipherteks yang bersesuaian.

3. Chosen-plaintext attack

Kriptanalis dapat memilih plainteks yang dimilikinya untuk dienkripsikan, yaitu plainteks-plainteks yang lebih mengarahkan penemuan kunci. Secara formal hal ini diformulasikan sebagai berikut:

Deberikan : $P_1, C_1 = E_k(P_1),P_2, C_2 = E_k(P_2),...,P_i,C_i = E_k(P_i)$ dimana kriptanalis dapat memilih diantara P1, P2,P3,...,Pi

Deduksi : k untuk mendapatkan P_{i+1} dari $C_{i+1} = Ek(P_{i+1})$



Gambar 2.4 Chosen-plaintext attack

Gambar 2.4 menggambarkan serangan jenis ini dengan memilih plainteks yang dienkripsi pada mesin ATM. Untuk diketahui, setiap kali nasabah memakukan PIN, mesin ATM mengenkripsi PIN tersebut lalu mengirim cipherteks tersebut ke komputer *server* di bank untuk diotentiksi. Seorang kriptanalis mengubah PIN yang dimilikinya, kemudian memakukan PIN tersebut ke mesin ATM. Rekan kriptanalis yang lain menyadap pengiriman cipherteks "di tengah jalan". Serangan ini diulang beberapa kali dengan beberapa nilai PIN sehingga kriptanalis memeperoleh sejumlah plainteks dan cipherteks yang berkoresponden. Kriptanalis mempelajari cipherteks tersebut untuk mendeduksi kunci penyandian.

4. Chosen-chiptertext attack

Kriptanalis memilih cipherteks untuk dienkripsikan dan memiliki akses ke plainteks hasil dekripsi (misalnya terhadap mesin elektronik yang melakukan dekripsi secara otomatis). Jenis serangan ini biasanya dipakai pada sistem kriptografi. Secara formal hal ini diformulasikan sebagai berikut :

Diberikan : C_1, $P_1 = D_k(C_1)$, C_2,$P_2 = D_k(C_2)$,...,C_i,$P_i = D_k(C_i)$

Deduksi : k (yang mungkin diperlukan untuk mendekripsi pesan pada waktu yang akan datang).

5. Chosen-text attack

Merupakan kombinasi *chosen-plaintext attack* dan *chosen-chipertext attack.*

2.4.4.3 Berdasarkan teknik yang digunakan dalam menemukan kunci, serangan dapat dibagi menjadi *exhaustive attack* dan *analilytical attack*

1. *exhaustive attack* atau *brute force attack*

ini adalah serangan untuk mengungkap painteks atau kunci dengan mencoba semua kemungkinan kunci. Diasumsikan kriptanalis mengetahui algoritma kriptografi digunakan oleh pengirim pesan.

Jika hanya cipherteks yang tersedia *(chipertext only)*, cipherteks tersebut didekripsi dengan setiap kemungkinan kunci, kemudian plainteks hasil dekripsi diperiksa apakah mengandung arti.

Jika cipherteks dan plainteks yang berkoresponden tersedia *(known plaintext)* maka *exhaustive attack* menjadi lebih mudah. Plainteks tersebut

dienkripsi dengan setiap kemungkinan kuncilalu hasilnya dibandingkan dengan cipherteks yang berkoresponden.

2. Analitycal attack

Jenis serangan ini, kriptanalis tidak mencoba-coba semua kemungkinan kunci tetapi menganalisis kelemahan algoritma kriptografi untuk mengurangi kemungkinan kunci yang tidak mungkin ada. Diasumsikan kriptanalis mengetahui algoritma kriptografi yang digunakan oleh pengirim pesan. Analisis dapat menggunakan pendekatan matematik dan statistik dalam rangka menemukan kunci. Secara statistik misalnya dengan menggunakan analisis frekuensi, sedangkan secara matematik dengan memecahkan persamaan-persamaan matematika yang mengandung peubah-peubah yang merepresentasikan plainteks atau kunci.

Untuk menghadapi serangan ini, kriptografer harus membuat algoritma kriptografi yang kompleks sedemikian sehingga plainteks merupakan fungsi matematika dari cipherteks dan kunci yang cukup kompleks, dan tiap kunci merupakan fungsi matematika dari cipherteks dan plainteks yang cukup kompleks.

2.5. Kriptografi kunci-publik

Diffie dan Hellman memecahkan masalah kriptografi asimetri yang memungkinkan pengguna berkomunikasi secara aman tanpa perlu berbagi kunci rahasia. Nama lainya adalah kriptografi kunci-publik (*public-key cryptography*), sebab kunci untuk enkripsi diumumkan kepada publik sehingga dapat diketahui oleh siapapun, sementara kunci dekripsi hanya diketahui oleh penerima pesan.

2.6. Konsep kriptografi kunci-publik

Kriptografi kunci-publik sederhana dan elegan, tetapi mempunyai konsekuensi penggunaan yang hebat. Seperti telah dijelaskan pada bagian awal, pada kriptografi kunci-publik, setiap pengguna memiliki sepasang kunci, satu kunci untuk enkripsi dan satu kunci untuk dekripsi (**Gambar2.5**). Kunci untuk enkripsi diumumkan kepada publik – oleh karena itu tidak rahasia – sehingga dinamakan **kunci publik** (*public key*), disimbolkan dengan *e*. Kunci untuk dekripsi bersifat rahasia – sehingga dinamakan **kunci privat** (*private key*), disimbolkan dengan *d*. Karena kunci enkripsi ≠ kunci dekripsi maka kriptografi kunci-publik disebut juga kriptografi asimetri.

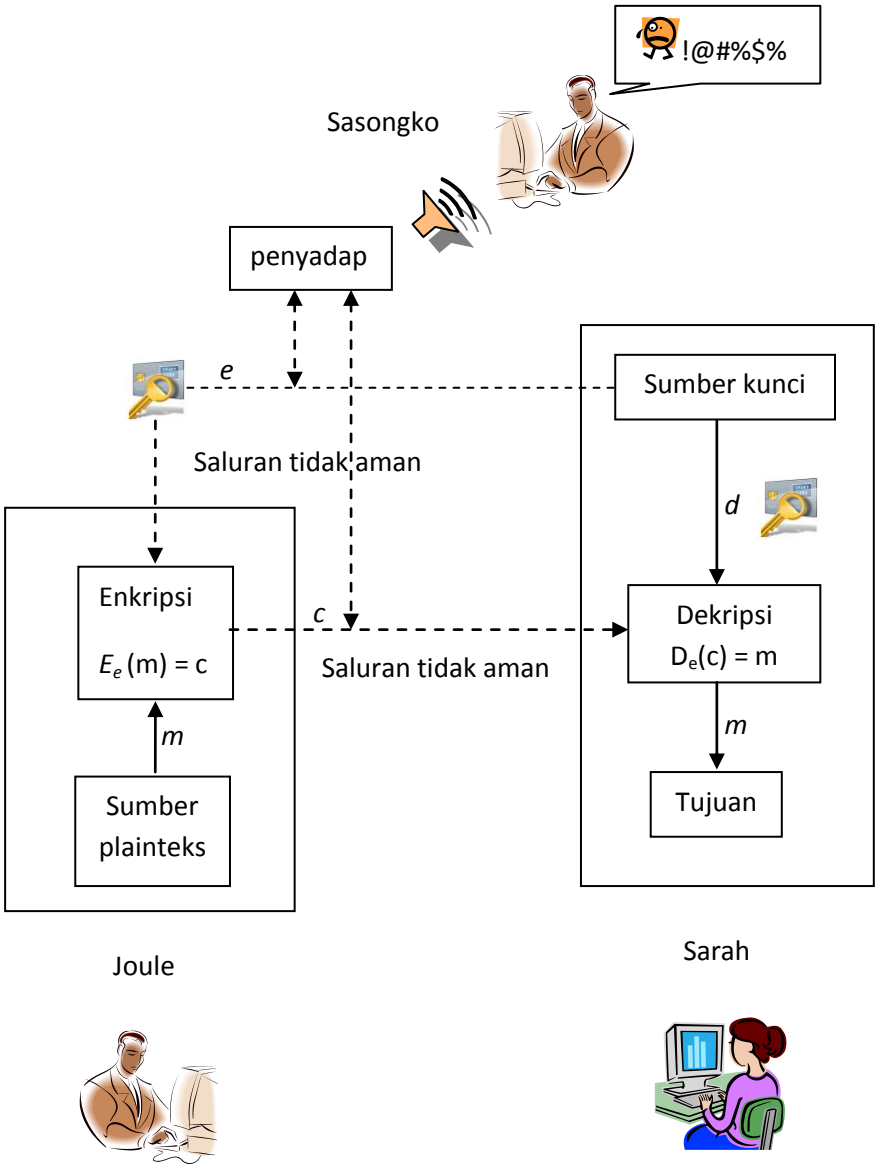
Misalkan *E* adalah fungsi enkripsi dan *D* adalah fungsi dekripsi. Misalkan (*e*,*d*) adalah pasangan kunci untuk enkripsi dan dekripsi sedemikian sehingga :

$$E_d(m) = c \text{ dan } D_d(c) = m$$

Untuk suatu plainteks *m* dan cipherteks *c*. Kedua persamaan ini menyiratkan bahwa dengan mengetahui *e* dan *c*, maka secara komputasi hampir tidak mungkin menemukan *m*. Asumsi lainya, dengan mengetahui *e*, secara hampir tidak mungkin menurunkan *d*. E_e digambarkan seabgai fungsi pintu-kolong(*trapdoor*) satu-arah dengan *d* adalah inforamsi *trapdoor* yang diperlukan untuk menghitung fungsi inversnya, *D*, yang dalam hal ini membuat proses dekripsi dapat dilakukan.

Misalkan Joule berkomunikasi dengan Sarah. Sarah memilih pasangan kunci (*e*,*d*). Sarah mengirimkan kunci enkripsi *e* (kunci publik) kepada Joule melalui sembarang saluran tetapi tetap menjaga kerahasiaan kunci dekripsinya, *d* (kunci privat). Kemudian, Joule ingin mengirim pesan *m* kepada Sarah. Joule

mengenkripsikan pesan *m* dengan menggunakan kunci Publik Sarah, untuk mendapatkan $c = E_e(m)$, lalu mengirimkan *c* melalui saluran komunikasi (yang tidak perlu aman). Sarah mendekripsi cipherteks *c* dengan menggunakan kunci privatnya untuk memperoleh $m = D_d(c)$.



Gambar 2.5 Enkripsi/dekripsi dengan kriptografi kunci-publik

Gambar ini memperlihatkan perbedaan mendasar sistem asimetri dengan sistem simetri. Sarah mengirim kunci publik, *e*, untuk enkripsi kepada Joule

melalui saluran yang tak perlu aman (*secure channel*). Saluran yang tidak perlu aman ini mungkin sama dengan saluran yang digunakan untuk mengirim cipherteks. Sasongko melakukan intersepsi komunikasi mungkin berhasil mendapatkan kunci enkripsi *e* dan cipherteks *c*, tetapi karena ia tidak mengetahui kunci dekripsi *d*, maka ia tidak dapat melakukan dekripsi. Hanya Sarah yang mengetahui kunci privatnya sendiri, *d*, sehingga ia mendekripsi pesan dari Joule dengan kunci privat tersebut.

Sistem kriptografi kunci-publik yang aman memiliki dua karakteristik sebagai berikut:

1. Komputasi untuk enkripsi/dekripsi pesan mudah dilakukan.
2. Secara komputasi hampir tidak mungkin (*infeasible*) menurunkan kunci privat, *d*, bila diketahui kunci publik, *e*, pasanganya.

Kedua karakteristik di atas dapat dianalogikan dengan dua masalah matematika berikut :

1. Perkalian *vs* pemfaktoran. Mengalikan dua buah bilangan prima,

   $a \times b = n$, mudah,tetapi memfaktorkan *n* menjadi faktor-faktor primanya lebih sulit.

   Contoh : 31 x 47 = 1457 (perkalian, mudah)

   1457 = ? x ? (pemfaktoran, sulit)

2. Perpangkatan *vs* logaritmik diskrit. Melakukan perpangkatan modulo, $b = a^2 \mod n$, mudah, tetapi menemukan *x* dari $a^x \equiv b \pmod n$ lebih sulit.

   Contoh : $126^2 \mod 1125 = 126$ (perpangkatan modulo, mudah)

   Carilah *x* dari $12^x \equiv 234 \pmod{1125}$ (logaritmik disrit ,sulit)

Dua masalah matematika di atas sering dijadikan dasar pembangkitan sepasang kunci pada kriptografi kunci-publik, yaitu

1. Pemfaktoran

Diberikan bilangan bulat *n*. Faktorkan *n* menjadi faktor primanya

Contoh : 10 = 2 x 5, 60 = 2 x 2 x 3 x 5, 2^{213} = 2 x 2 x 2 x 2 x ....(dst).

Semakin besar *n*, semakin sulit memfaktorkan (butuh waktu lama).

2. Logaritma diskrit

Temukan x sedemikian sehingga $a^x \equiv b \pmod{n}$ sulit dihitung.

Contoh : jika diketahui $a = 3$, $b = 15$, $n = 17$,

maka dengan rumus $a^x \equiv b \pmod{n}$ menghasilkan $3^x \equiv 15 \pmod{17}$

$\Leftrightarrow 3^x \bmod 17 = 15$, cari x | $3^x \bmod 17 = 15$. Ditemukan bahwa x = 6.

Semakin besar *a, b* dan *n* semakin sulit memfaktorkan.

Persoalan logaritma diskrit adalah kebalikan dari persoalan perpangkatan modular :

$$a^x \bmod n$$

2.6.1 Kelebihan kriptografi kunci-publik (asimetri)

1. Hanya kunci private yang perlu dijaga kerahasiaanya oleh setiap entitas yang berkomunikasi (tetapi, otentikasi kunci publik tetap harus terjamin). Tidak ada kebutuhan mengirim kunci kunci privat sebagaimana pada sistem simetri.
2. Pasangan kunci publik/ kunci privat tidak perlu diubah, bahkan dalam periode waktu yang panjang.
3. Dapat digunakan untuk mengamankan pengiriman kunci simetri.

4. Beberapa algoritma kunci-publik dapat digunakan untuk memberi tanda tangan digital pada pesan.

2.6.2 Kekurangan kriptografi kunci-publik (asimetri)

1. Ekripsi dan dekripsi data umumnya lebih lambat daripada sistem simetri, karena enkripsi dan dekripsi menggunakan bilangan yang besar dan melibatkan operasi perpangkatan yang besar.
2. Ukuran cipherteks lebih besar daripada plainteks (2 – 4 kali lebih besar)
3. Ukuran kunci relatif lebih besar dari pada kunci simetri
4. Karena kunci publik diketahui secara luas dan dapat digunakan setuap orang, maka cipherteks tidak memberikan inforamsi mengenai otentikasi pengirim.

2.7. Kriptografi ElGamal

Algoritma ElGamal dibuat oleh Taher ElGamal pada tahun 1984. Algoritma ini pada mulanya digunakan untuk tanda tangan digital atau *digital signature.* Namun kemudian dimodifikasi sehingga juga bisa digunakan untuk enkripsi dan dekripsi. ElGaMal digunakan dalam perangkat lunak yang dikembangkan oleh GNU, program PGP, dan pada program keamanan jaringan lainnya. Keamanan algoritma ini terletak pada sulitnya menghitung algoritma diskrit.

Algoritma ElGamal banyak dibahas pada buku-buku kriptografi, tetapi masih sedikit yang membahas secara mendetail tentang konsep-konsep matematisnya. Stinson (1995) telah menjelaskan secara umum tentang algoritma ElGamal beserta sistem pendukungnya. Buchmann (2000) secara khusus menitikberatkan pada pemahaman konsep dasar matematis dari algoritma

ElGamal, seperti teori bilangan bulat, persamaan kongruen, dan struktur aljabar abstrak yang meliputi grup, homomorfisma dan gelanggang. Pembahasan aljabar abstrak yang lebih terperinci diberikan oleh Fraleigh (2000), namun tidak ada pembahasan yang mengaitkan secara langsung dengan algoritma ElGamal. Sedangkan implementasi algoritma ElGamal diberikan oleh Menezes, Oorschot dan Vanstone (1996), termasuk penjelasan beberapa algoritma yang dapat digunakan untuk membuat program komputer.

Besaran-besaran yang digunakan di dalam algoritma ElGamal :

1. Bilangan prima, p (tidak rahasia)
2. Bilangan acak, α ($\alpha < p$) (tidak rahasia)
3. Bilangan acak, a ($a < p$) (rahasia, kunci privat)
4. m (plainteks) (rahasia)
5. a dan b (cipherteks) (tidak rahasia)

2.8. Proses pembentukan kunci

Proses pertama adalah pembentukan kunci yang terdiri dari kunci rahasia dan kunci publik. Pada proses ini dibutuhkan sebuah bilangan prima p yang digunakan untuk membentuk grup Zp*, elemen primitif α dan sebarang $a \in \{1,..., p - 2\}$. Kunci publik algoritma ElGamal berupa pasangan 3 bilangan, yaitu (p, α, β), dengan

$$\beta = \alpha^{a} \bmod p$$

Sedangkan kunci rahasianya adalah bilangan a tersebut.

Karena pada algoritma ElGamal menggunakan bilangan bulat dalam proses perhitungannya, maka pesan harus dikonversi ke dalam suatu bilangan bulat. Untuk mengubah pesan menjadi bilangan bulat, digunakan kode ASCII (*American Standard for Information Interchange*). Kode ASCII merupakan representasi numerik dari karakter-karakter yang digunakan pada komputer, serta mempunyai nilai minimal 0 dan maksimal 255. Oleh karena itu, berdasarkan sistem kriptografi ElGamal di atas maka harus digunakan bilangan prima yang lebih besar dari 255. Kode ASCII berkorespondensi 1-1 dengan karakter pesan.

Berikut ini diberikan suatu algoritma yang dapat digunakan untuk melakukan pembentukan kunci.

Algoritma Membangkitkan Pasangan Kunci :

1. Pilih sembarang bilangan prima p(>255).
2. Pilih dua buah bilangan acak, α dan a, dengan syarat $\alpha < p$ dan $1 \leq a \leq p-2$.
3. Hitung $\beta = \alpha^a \bmod p$.
4. Publikasikan nilai p, α dan β , serta rahasiakan a.

Hasil dari algortma ini :

Kunci Publik : triple (β,α,p)

Kunci privat : pasangan (a,p)

Pihak yang membuat kunci publik dan kunci rahasia adalah penerima, sedangkan pihak pengirim hanya mengetahui kunci publik yang diberikan oleh penerima, dan kunci publik tersebut digunakan untuk mengenkripsi pesan. Jadi,

kentungan menggunakan algoritma kriptografi kunci publik adalah tidak ada permasalahan pada distribusi kunci apabila jumlah pengirim sangat banyak serta tidak ada kepastian keamanan jalur yang digunakan.

2.9. Enkripsi

Pada proses ini pesan dienkripsi menggunakan kunci publik (β,α,p) dan sebarang bilangan acak rahasia k ϵ {1, ..., p-2}. Misalkan m adalah pesan yang akan dikirim. Selanjutnya, m diubah ke dalam blok-blok karakter dan setiap karakter dikonversikan ke dalam kode ASCII, sehingga diperoleh plainteks m_1, m_2, ..., m_n dengan m_i ϵ {1, 2,..., p-1}, i = 1, 2,..., n . Untuk nilai ASCII pemisah ";" digunakan untuk menandai akhir dari suatu teks.

Proses enkripsi pada algoritma ElGamal dilakukan dengan menghitung

$$\gamma = \alpha^{k} \, mod \, p$$

dan

$$\delta = \beta^{k}.m \, mod \, p$$

dengan rahasia k ϵ {1, ..., p-2} acak. Diperoleh cipherteks (γ,δ).

Bilangan acak k ditentukan oleh pihak pengirim dan harus dirahasiakan, jadi hanya pengirim saja yang mengetahuinya, tetapi nilai k hanya digunakan saat melakukan enkripsi saja dan tidak perlu disimpan. Berikut adalah algoritma enkripsi.

2.9.1 Algoritma enkripsi :

1. Plainteks disusun menjadi blok-blok m_1, m_2, …, sedemikian sehingga setiap blok merepresentasikan nilai di dalam rentang 0 sampai $p - 1$ (dengan mengubah nilai m menjadi kode ASCII).
2. Pilih bilangan acak k, yang dalam hal ini $0 < k < p - 1$, sedemikian sehingga k relatif prima dengan $p - 1$.
3. Setiap blok m dienkripsi dengan rumus :

$$\gamma = \alpha^{k} \bmod p$$

$$\delta = \beta^{k} m \bmod p$$

Pasangan γ dan δ adalah cipherteks untuk blok pesan m. Jadi, ukuran cipherteks dua kali ukuran plainteksnya.

Salah satu kelebihan algoritma ElGamal adalah bahwa suatu plainteks yang sama akan dienkripsi menjadi cipherteks yang berbeda-beda. Hal ini dikarenakan pemilihan bilangan k yang acak. Akan tetapi, walaupun cipherteks yang diperoleh berbeda-beda, tetapi pada proses dekripsi akan diperoleh plainteks yang sama.

2.10. Dekripsi

Setelah menerima cipherteks (γ,δ), proses selanjutnya adalah mendekripsi cipherteks menggunakan kunci publik p dan kunci rahasia a. Dapat ditunjukkan bahwa plainteks m dapat diperoleh dari cipherteks menggunakan kunci rahasia a.

Diberikan (p, α, β) sebagai kunci publik dan a sebagai kunci rahasia pada algoritma ElGamal. Jika diberikan cipherteks (γ,δ), maka untuk mendekripsi γ

dan β digunakan kunci rahasia, *a*, dan plainteks *m* diperoleh kembali dengan persamaan :

$$m = \delta.(\gamma^{\alpha})^{-1} \, mod \, p$$

dengan *m* adalah plainteks.

$$m = \delta / \gamma^{a} \bmod p$$

Catatlah bahwa karena

$$\gamma^{a} \equiv \alpha^{k a} \pmod{p}$$

maka

$$\delta / \gamma^{\alpha} \equiv \beta^{k} m / \gamma^{\alpha}$$
$$\equiv \alpha^{a k} m / \alpha^{ak}$$
$$\equiv m \pmod{p}$$

Dengana demikian terbukti bahwa m = $\delta.(\gamma^{a})^{-1}$ mod *p*, yang berarti bahwa plainteks dapat ditemukan kembali dari pasangan cipherteks γ dan δ.

Karena Zp* merupakan grup siklik yang mempunyai order *p* -1 dan *a* ϵ {1,..., *p* - 2}, maka $(\gamma^{a})^{-1} = \gamma^{-a} = \gamma^{p-1-a} \, mod \, p$.

2.10.1 Algoritma dekripsi

Diketahui cipherteks (γ,δ), *i* =1, 2,..., *n*, kunci publik *p* dan kunci rahasia *a*.

Langkah :

1. Untuk *i* dari 1 sampai *n* kerjakan :

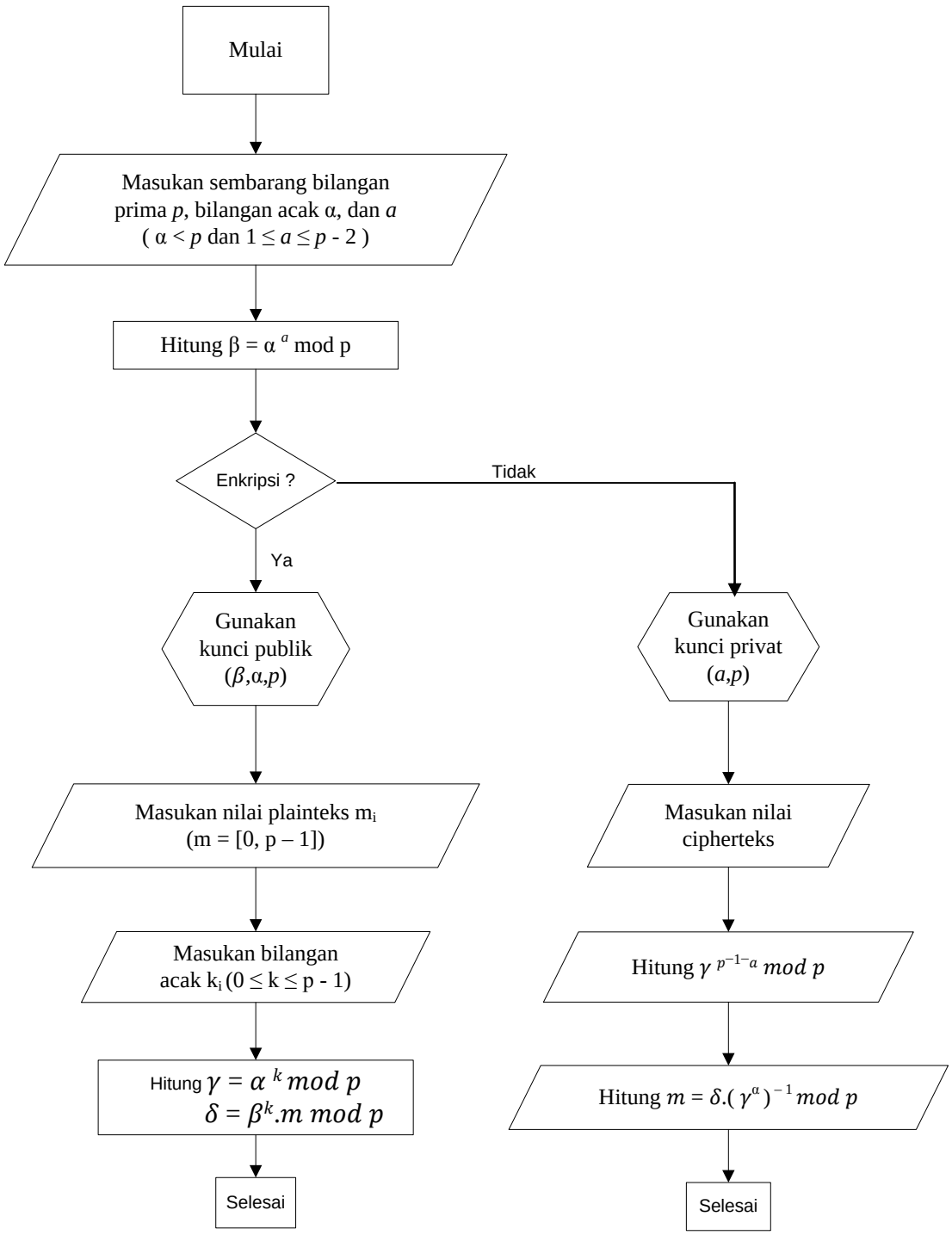
   1.2 Hitung $\gamma_i^{p-1-a} \bmod p$

   1.3 Hitung $m_i = \delta_i . \gamma_i^{p-1-a} \bmod p$

2. Diperoleh plainteks m_1 m_2 , ,..., m_n.

Konversikan masing-masing bilangan m_1 , m_2,..., m_n ke dalam karakter sesuai dengan kode ASCII-nya, kemudian hasilnya digabungkan kembali.

2.11. Flowchart pembentukan kunci, enkripsi dan dekripsi



Gambar 2.6 Flowchart proses enkripsi dan dekripsi

2.12. Metode *Fast Exponentiation*

Diberikan sebarang grup G, $g \epsilon G$ dan bilangan bulat positif z. Untuk menghitung g^z dilakukan langkah berikut ini.

Dibentuk ekspansi biner dari bilangan bulat z, yaitu

$$z = \sum_{i=0}^{k} a_i . 2^i$$

Karena z ditulis dalam expansi biner, maka $a_i \epsilon \{0,1\}$. Sehingga

$$g^z = g^{\sum_{i=0}^{k} a_i . 2^i} = \prod_{i=0}^{k} \left(g^{2^i} \right)^{a_i} = \prod_{1 \le i \le k, a_i = 1} g^{2^i} .$$

Dengan hasil ini diperoleh cara yang cepat untuk menghitung g^z yang disebut dengan metode *Fast Exponentiation*, yaitu

1. Hitung nilai g^{2^i} , $0 \le i \le k$
2. Nilai g^z merupakan hasil perkalian nilai-nilai g^{2^i} , dengan a_i diperoleh bahwa

$$g^{2^{i+1}} = \left(g^{2^i} \right)^2 .$$

2.12.1 Contoh perhitungan pangkat dengan *Fast Exponentiation*

Berikut contoh perhitungan pangkat bilangan bulat modulo yang besar menggunakan metode *Fast Exponentiation*

Dihitung 2^{380} mod 1234

Tabel 2.1 Perhitungan 2^{380} mod 1234

i	0	1	2	3	4	5	6	7	8
k_i	1	0	1	1	1	1	1	0	1
A	2	4	16	256	134	680	884	334	496
b	2	2	16	394	968	518	98	98	482

Tabel 2.1 Dapat dilihat bahwa 2^{380} mod 1234 = 482

2.12.2 Algoritma *Fast Exponentiation*

Diketahui $a < m$, m = bilangan bulat positif dan k bilangan bulat, $0 \leq k \leq m$ dengan representasi biner dari $k = \sum_{i=0}^{t} k_i . 2^i$.

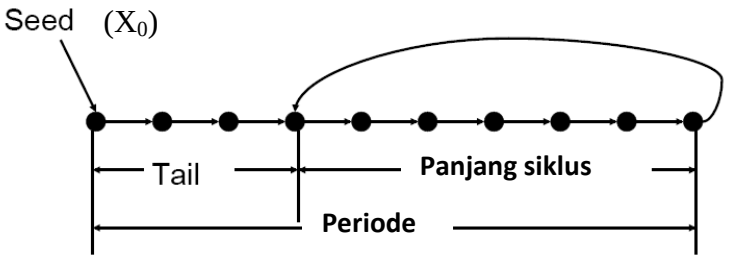
Langkah :

1. *Set* b ← 1. Jika $k = 0$, maka *output(b).*
2. *Set* A ← *a*.
3. Jika $k_0 = 1$, maka *set b* ← *a.*
4. Untuk *i* dari 1 sampai *t* kerjakan :
   4.1 *Set A* ← A^2 mod *n.*
   4.2 Jika $k_i = 1$, maka *set b* ← *A.b* mod *n.*
5. *Output(b).*

2.13. Bilangan acak semu *PRNG* (*pseudorandom number generator*)

Untuk membangkitkan pasangan kunci pada kriptografi dan validasi pada Luhn dibutuhkan sekelompok bilangan acak. Dalam implementasinya bilangan acak dapat dibuat dengan fungsi acak dalam bahasa pemrograman tertentu, namun bagaimana jika kemunculan bilangan acak dapat diprediksi.

Generator bilangan acak semu (*PRNG*) dapat digambarkan seperti di bawah ini :



Gambar 2.7 Siklus generator bilangan acak semu

Jika panjang siklus sudah memenuhi suatu periode tertentu, maka siklus bilangan acak akan kembali ke awal. Dengan demikian kemunculan bilangan acak berikutnya dapat diprediksi. Jika *k* adalah bilangan acak, dan (*k* + 1) adalah bilangan acak selanjutnya, maka setelah *k* ke-*n* dalam periode tertentu akan kembali ke (*k* + 1). Sehingga bilangan acak selanjutnya dapat ditemukan dengan perhitungan statistik.

Pembangkit bilangan acak yang dapat menghasilkan bilangan yang tidak dapat diprediksi oleh pihak lawan, cocok untuk kriptografi, pembangkit tersebut dinamakan *cryptographycally secure pseudorandom generator* (*CSPRNG*).

Persyaratan *CSPRNG* pertama adalah :

1. Secara statistik mempunyai sifat-sifat yang bagus (lolos uji statistik)

2. Tahan terhadap serangan (*attack*) yang serius. Serangan ini bertujuan untuk memprediksi bilangan acak yang dihasilkan.

Persyaratan kedua adalah :

1. Setiap *CSPRNG* seharusnya memenuhi uji-bit berikutnya(*next-bit test*) sebagai berikut : diberikan *k* buah bit barisan acak, maka tidak ada algoritma dalam waktu polinomial yang dapat memprediksi bit ke -(*k*+1) dengan peluang keberhasilan lebih besar dari ½.

2. Setiap *CSPRNG* dapat menahan "perluasan status", yaitu jika sebagian atau semua statusnya dapat diungkap (atau diterka dengan benar), maka tidak mungkin merekonstruksi aliran bilangan acak.

2.13.1 Perancangan *CSPRNG* berbasis teori bilangan

CSPRNG berbasis teori bilangan dapat dirancang berdasarkan persoalan matematika, seperti pemfaktoran bilangan menjadi faktor prima, logaritma diskrit, dsb.

2.13.2 Algoritma *CSPRNG* berbasis RSA

CSPRNG yang akan digunakan adalah modifikasi dari *RSA*. Algoritma pembangkit bilangan acak berbasis *RSA* adalah sebagai berikut :

1. Pilih dua buah bilangan prima rahasia, *p* dan *q*, kalikan keduanya menjadi $n = pq$ dan $\phi = (p-1)(q-1)$. Pilih bilangan bulat acak *e*, $1 \leq e \leq \phi$ dimana gcd $(e, \phi) = 1$.

2. Pilih bilangan bulat acak lain, x_0 , $x_0 \in [1, n-1]$.

3. Untuk $1 \leq i \leq l$ lakukan :

   3.1 $x_i \leftarrow x_{i-1}^{e} \pmod{n}$

   3.2 $z_i \leftarrow$ *least significant bit* dari x_i

4. Barisan bilangan acak $z_1,z_2,z_3,...$

Keamanan pembangkit bilangan acak modifikasi *RSA* ini terletak pada sulitnya memfaktorkan *n*.

2.14. Pembagi persekutuan terbesar

Berikut ini dijelaskan pengertian dan sifat-sifat suatu bilangan yang disebut dengan pembagi persekutuan terbesar. Pembagi persekutuan dari bilangan bulat $a_1,a_2,...a_k$ adalah suatu bilangan bulat yang membagi $a_1,a_2,...a_k$. Diberikan $a_1,a_2,...a_k \in$ bilangan bulat. Suatu bilangan bulat non-negatif *d* disebut pembagi persekutuan terbesar (*greatest common divisor*) dari $a_1,a_2,...a_k$ jika :

1. Bilangan bulat *d* merupakan pembagi persekutan dari $a_1,a_2,...a_k$, yaitu *d* membagi $a_1,a_2,...a_k$,
2. Untuk sembarang bilangan bulat *c*, jika *c* membagi $a_1,a_2,...a_k$, maka *c* membagi *d.*
3. Bilangan bulat *d* tersebut dinotasikan dengan $d = \gcd(a_1,a_2,...a_k)$

Dengan kata lain, pembagi persekutuan terbesar adalah nilai maksimum dari semua pembagi persekutuan, yaitu

$\gcd(a_1,a_2,...a_k) = \max \{n \in$ bilangan bulat positif :$n|a_1$ dan $n|a_2$ dan ... dan $n|a_k\}$.

2.14.1 Algoritmna Euclide

Berikut ini diberikan sebuah algoritma yang dapat digunakan untuk menghitung nilai pembagi persekutuan terbesar dari dua bilangan bulat dengan sangat mangkus. Algoritma ini didasarkan pada teorema di bawah ini.

Diberikan $a,b \epsilon$ bilangan bulat positif

1. *while* $b \neq 0$ *do* :

   1.1 *Set* $r \leftarrow a \bmod b$, $a \leftarrow b$, $b \leftarrow r$

2.15. Luhn

Algoritma Luhn atau formula Luhn yang biasa dikenal sebagai “modulus 10” atau algoritma “mod 10”, adalah formula sederhana untuk memvalidasi nomor identifikasi, seperti nomor kartu kredit, nomor IMEI, NPIN dan sebagainya. Algoritma Luhn dikembangkan oleh peneliti dari IBM Hans Peter Luhn dipatenkan dalam undang-undang Amerika Serikat No.2,950,048, disahkan pada tanggal 6 januari 1954, dan dipublikasikan pada 23 agustus ’60.

Algoritma ini menjadi domain publik dan digunakan secara luas saat ini. Hal ini ditetapkan dalam ISO/7812-1 IEC. Algoritma ini tidak dimaksudkan untuk menjadi fungsi hash yang aman dalam kriptografi, tetapi algoritma ini dirancang untuk melindungi terhadap kesalahan yang tidak disengaja, bukan dari serangan yang serius. Kebanyakan kartu kredit dan nomor identifikasi pemerintah menggunakan algoritma ini sebagai metode sederhana untuk membedakan nomor valid dan kumpulan bilangan acak

2.15.1 Analisis algoritma Luhn

Algoritma Luhn merupakan suatu algoritma yang digunakan untuk mengecek validitas suatu angka berdasarkan digit-digit yang membentuknya.

Algoritma Luhn dapat mendeteksi kesalahan setiap digit angka yang dibangun, termasuk semua transposisi dari angka yang saling berdampingan seperti 09 – 90.

Pada pembangunan digit angka yang akan diterapkan pada *smart card* mengadopsi beberapa kontribusi yang ditetapkan *card issuer* yang akan berpengaruh terhadap total perhitungan dari digit yang ada, dalam hal ini tergantung dari posisi digit tersebut pada *smart card*. Jika terdapat kesalahan pada salah satu digit, maka hasilnya pun akan tidak sesuai dengan ketentuan cek digit Luhn.

2.15.2 Langkah pembuatan digit Luhn :

Untuk menentukan nilai Luhn, ada beberapa langkah yang perlu dilakukan, antara lain:

1. Buat bilangan acak sebanyak *t* (*t* diambil dari jumlah digit NIM + MCUID + CSPRNG_RSA).
2. Mengalikan semua bilangan yang berada pada digit ganjil dengan bilangan 2. Bila nilainya lebih dari 9, maka dikurangi dengan 9.
3. Menjumlahkan semua hasil perkalian pada digit ganjil di atas.
4. Menjumlahkan semua bilangan yang berada pada digit genap.
5. Jumlahkan nilai pada langkah 3 dan 4.
6. Nilai Luhn adalah 10 – (nilai akhir modulus 10)

2.15.3 Langkah pengecekan digit Luhn

Metode cek ini membutuhkan 3 langkah untuk membuktikan apakah suatu kartu memenuhi cek digit Luhn, caranya sebagai berikut :

1. Untuk setiap digit pada posisi ganjil (perhitungan digit mulai dari kiri dengan yang paling kiri adalah adalah digit ke-1, kalikan nilainya dengan 2, jika hasilnya lebih dari 9, maka dikurangi dengan 9. Jumlahkan semua angka dengan yang telah didapat itu.

2. Untuk setiap digit pada posisi genap jumlahkan semua nilainya dan tambahkan hasilnya dengan hasil pada langkah 1.

3. Jika hasil pada langkah 2 hasilnya habis dibagi 10, *smart card* tersebut maka sah.

Metode algoritma ini dapat dimanfaatkan sebagai cara agar penggandaan *smart card* tidak efektif.