BAB 2
LANDASAN TEORI

2.1 Sejarah Kriptografi

Kriptografi mempunyai sejarah panjang dan sangat menarik. Kriptografi sudah digunakan 4000 tahun yang lalu yang diperkenalkan oleh orang-orang Mesir untuk mengirim pesan ke pasukan militer yang berada di lapangan dan supaya pesan tersebut tidak bisa dibaca oleh pihak musuh apabila pasukannya tertangkap maka pesan tersebut disandikan.

Pada zaman romawi kuno dikisahkan pada suatu saat ketika Julius Caesar ingin mengirimkan satu pesan rahasia kepada seorang Jenderal di medan perang, pesan tersebut harus dikirimkan melalui seorang kurir. Tetapi karena pesan tersebut bersifat rahasia, supaya pesan tersebut terbuka ditengah jalan baik oleh musuh maupun oleh kurirnya sendiri maka Julius Caesar memikirkan cara bagaimana mengatasinya dengan cara mengacak pesan tersebut menjadi suatu pesan yang tidak dapat dipahami oleh siapapun kecuali hanya oleh Jenderalnya saja. Sebelumnya sang Jenderal tersebut telah diberi tahu bagaimana cara membaca pesan yang telah diacak tersebut karena telah mengetahui kuncinya.

Pada perang dunia kedua, negara Jerman menggunakan enigma atau juga disebut dengan mesin rotor yang digunakan Hitler untuk mengirim pesan ke tentaranya. Jerman sangat percaya pesan yang dikirim melalui enigma tidak terpecahkan kode-kode enkripsinya. Tapi anggapan itu keliru, setelah bertahuntahun sekutu dapat memecahkan kode-kode tersebut setelah mempelajarinya. Setelah Jerman mengetahui kode-kode tersebut dapat terpecahkan, maka enigma yang digunakan pada perang dunia kedua, beberapa kali mengalami perubahan.

Enigama yang digunakan Jerman bisa mengenkripsikan satu pesan mempunyai 15 milyaran kemungkinan hasil dekripsinya. Selama bertahun-tahun kritografi menjadi bidang khusus yang dipelajari oleh pihak militer seperti agen keamanan nasional Amerika, Uni Soviet, Inggris, Perancis, Israel, dan

negaranegara lainya yang telah membelanjakan milyaran dollar untuk mengamankan komunikasi mereka agar tidak dapat diketahui oleh negara lain tetapi disisi lain mereka juga mempelajari kode-kode rahasia negara lain, dengan adanya persaingan ini maka kriptografi terus berkembang sesuai dengan perkembangan zaman.

Namun pada 30 tahun terakhir ini, kriptografi tidak hanya dimonopoli oleh pihak militer saja, hal yang sama juga dilakukan oleh individu-individu yang menginginkan pesan dan komunikasi mereka tidak diketahui oleh pihak lain. Apalagi pada zaman sekarang ini persaingan yang begitu tinggi, mereka rela mengeluarkan milyaran dollar hanya untuk menjaga *privacy* mereka.

2.1.1 Kriptografi

Kriptografi berasal dari bahasa yunani, menurut bahasa dibagi menjadi dua *kripto* dan *graphia*, *kripto* berarti s*ecret* (rahasia) dan *graphia* berarti *writting* (tulisan). Menurut terminologinya kriptografi adalah ilmu sekaligus seni untuk menjaga keamanan pesan. Para pelaku atau praktisi kriptografi disebut cryptographers. Sebuah algoritma kriptografik (*cryptographic algorithm*), disebut cipher, merupakan persamaan matematik yang digunakan untuk proses enkripsi dan dekripsi. Biasanya kedua persamaan matematik (untuk enkripsi dan dekripsi) tersebut memiliki hubungan matematis yang cukup erat. Keamanan pesan diperoleh dengan menyandikannya menjadi pesan yang tidak memiliki makna. Zaman sekarang ini kerahasiaan informasi menjadi sesuatu yang penting. Informasi yang rahasia perlu disembunyikan agar tidak diketahui oleh pihak yang tidak berhak. Kriptografi atau yang sering dikenal dengan sebutan ilmu pendian data, merupakan suatu bidang ilmu dan seni (*art and science*) yang bertujuan untuk menjaga kerahasiaan suatu pesan dari pihak-pihak yang tidak berhak untuk yang berupaya mengakses pesan tersebut.

Kriptografi pada awalnya dijabarkan sebagai ilmu yang mempelajari bagaimana cara menyembunyikan pesan. Namun pada pengertian modern kriptografi adalah ilmu yang berdasarkan pada teknik matematika yang erat kaitannya dengan keamanan informasi seperti kerahasiaan, keaslian data,

keutuhan data dan anti penyankalan. Jadi pengertian kriptografi modern adalah bukan hanya penyembunyian pesan namun lebih pada sekumpulan teknik yang menyediakan keamanan informasi.

Pesan asli sebelum dirahasiakan disebut *plainteks* (*plaintext* artinya teks asli yang dapat dimengerti), sedangkan pesan hasil penyandian disebut dengan *ciphertext* (artinya teks tersandi). Pesan yang telah tersandikan dapat dikembalikan lagi ke pesan aslinya hanya oleh orang yang berhak (orang yang berhak adalah orang yang mengetahui metode penyandian atau memiliki kunci penyandian).

Menurut ISO 7498-2, terminologi yang lebih tepat digunakan adalah "*encipher*". Proses sebaliknya, untuk mengubah *ciphertext* menjadi *plaintext*, disebut dekripsi (*decryption*). Menurut ISO 7498-2, terminologi yang lebih tepat untuk proses ini adalah "*decipher*".

Cryptanalysis adalah seni dan ilmu untuk memecahkan *ciphertext* tanpa bantuan kunci. *Cryptanalyst* adalah pelaku atau praktisi yang menjalankan cryptanalysis.



Gambar 2.1 Proses Kriptografi

Kriptografi menggunakan metode ilmiah untuk melindungi informasi, sehingga kriptografi adalah sebuah cara yang efektif untuk melindungi informasi berharga untuk disimpan pada sebuah media penyimpanan atau ditransmisikan melalui jaringan komunikasi yang tidak aman.

Ada empat tujuan mendasar dari ilmu kriptografi ini yang juga merupakan aspek keamanan informasi yaitu:

- Kerahasiaan, adalah layanan yang digunakan untuk menjaga isi dari informasi dari siapapun kecuali yang memiliki otoritas atau kunci rahasia untuk membuka informasi yang telah disandi.
- Integritas data

  Berhubungan dengan penjagaan dari perubahan data secara tidak sah. Untuk menjaga integritas data, sistem harus memiliki kemampuan untuk mendeteksi manipulasi data oleh pihak-pihak yang tidak berhak, antara lain penyisipan, penghapusan, dan perubahan data lain kedalam data yang sebenarnya.

  Dalam skenario banyak, pengirim dan penerima pesan mungkin memiliki kebutuhan untuk *Key*akinan bahwa pesan belum diubah selama transmisi. Meskipun menyembunyikan enkripsi isi pesan, dimungkinkan untuk mengubah sebuah pesan terenkripsi tanpa memahaminya. (Algoritma enkripsi Beberapa, yang dikenal sebagai *nonmalleable* yang, mencegah hal ini, tetapi yang lain tidak.) Namun, jika pesan secara digital ditandatangani, setiap perubahan dalam pesan setelah tanda tangan akan membatalkan tanda tangannya. Selain itu, tidak ada cara yang efisien untuk memodifikasi pesan dan tanda tangan untuk menghasilkan pesan baru dengan tanda tangan yang sah, karena ini masih dianggap layak oleh sebagian besar komputasi fungsi hash kriptografi.
- Autentikasi

  Berhubungan dengan identifikasi/pengenalan, baik secara kesatuan sistem maupun informasi itu sendiri. Dua pihak yang saling berkomunikasi harus saling memperkenalkan diri. Informasi yang dikirimkan melalui kanal harus diautentikasi keaslian, isi datanya, waktu pengiriman, dan lain-lain.

  Meskipun pesan seringkali dapat mencakup informasi tentang entitas mengirim pesan, bahwa informasi mungkin tidak akurat. Tanda tangan digital dapat digunakan untuk otentikasi sumber pesan. Ketika kepemilikan kunci rahasia tanda tangan digital terikat kepada pengguna tertentu, tanda tangan yang sah menunjukkan bahwa pesan yang dikirim oleh pengguna tersebut. Pentingnya kepercayaan yang tinggi dalam

otentisitas pengirim ini terutama jelas dalam konteks keuangan. Misalnya, kantor cabang bank mengirimkan instruksi ke kantor pusat meminta perubahan saldo account. Apabila kantor pusat tidak yakin bahwa pesan tersebut benar-benar dikirim dari sumber resmi, bertindak atas permintaan semacam itu bisa menjadi kesalahan besar.

- Non-repudiasi.

  Usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman suatu informasi oleh yang mengirimkan data tersebut.

  Merupakan aspek penting dari tanda tangan digital. Dengan properti ini suatu entitas yang telah menandatangani beberapa informasi tidak dapat di lain waktu menyangkal memiliki menandatanganinya. Demikian pula, akses ke kunci publik hanya tidak memungkinkan pihak penipuan untuk palsu tanda tangan valid.

Pembakuan penulisan pada kriptografi dapat ditulis dalam bahasa matematika. Fungsi-fungsi yang mendasar dalam kriptografi adalah enkripsi dan dekripsi. Enkripsi adalah proses mengubah suatu pesan asli (*plaintext*) menjadi suatu pesan dalam bahasa sandi (*ciphertext*), C = E (M). Sedangkan dekripsi adalah proses mengubah pesan dalam suatu bahasa sandi menjadi pesan asli kembali, yaitu M = D (C), dimana:

C = Pesan dalam bahasa sandi (untuk ringkasnya disebut sandi).

D = Proses dekripsi.

E = Proses enkripsi.

M = Pesan asli.

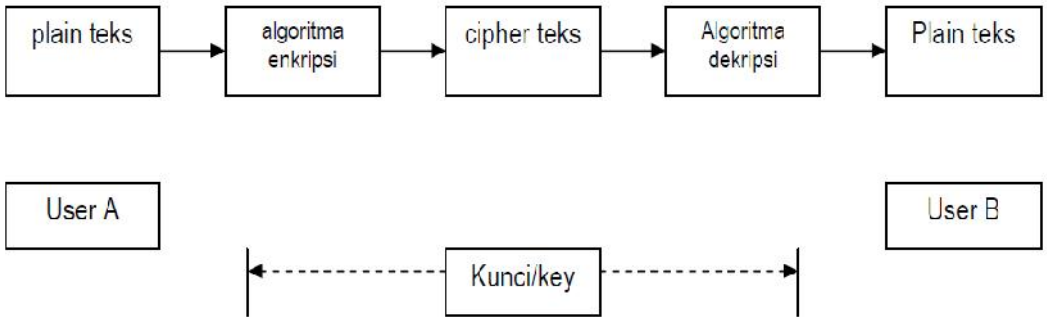
Umumnya, selain menggunakan fungsi tertentu dalam melakukan enkripsi dan dekripsi, seringkali fungsi itu diberi parameter tambahan yang disebut dengan istilah kunci yang dibagi menjadi kunci simetris dan asimetris [1].

Pada pembahasan selanjutnya yang digunakan adalah kunci asimetris atau disebut kriptografi asimetris/ kriptografi kunci publik.

2.1.2 Kriptografi Simetri

Algoritma simetris atau disebut juga algoritma kriptografi konvensional adalah algoritma yang menggunakan kunci yang sama untuk proses enkripsi dan proses dekripsi.

Algoritma Kriptografi simetris dibagi menjadi 2 kategori yaitu algoritma aliran (*Stream Ciphers*) dan algoritma blok (*Block Ciphers*). Pada algoritma aliran, proses penyandiannya berorientasi pada satu bit atau satu byte data. Sedang pada algoritma blok, proses penyandiannya berorientasi pada sekumpulan bit atau byte data (per blok). Contoh: algoritma kunci simetris adalah *DES (Data Encryption Standard), blowfish, twofish, MARS, IDEA, 3DES (DES diaplikasikan 3 kali), AES (Advanced Encryption Standard)* yang bernama asli Rijndael.



Gambar 2.2 Proses Kriptografi Simetri

Kriptografi kunci simetri merupakan jenis kriptografi yang paling umum dipergunakan, kunci yang digunakan untuk enkripsi sama dengan kunci untuk dekripsi. Jadi pembuat pesan dan penerimanya harus memiliki kunci yang sama dan harus dirahasiakan. Kelebihan kriptografi kunci simetri adalah lebih cepat apabila dibandingkan dengan seistem kriptografi lain. Kelemahan dari sistem ini bukan hanya masalah pengiriman *chipertext*-nya melainkan pengirim harus mencari cara yang tepat untuk memberitahukan kunci kepada penerima. Selain itu masalah efisiensi kunci dimana apabila terdapan pengguna sebanyak n, maka dibutuhkan kunci sebanyak n (n-1)/2 sehingga untuk jumlah pengguna yang banyak sistem kriptografi kunci publik kurang efisien. Dua kategori

algoritmayang termasuk pada sistem kriptografi kunci simetri ini adalah algoritma *block chiper* dan *stream chiper*.

1. *Block Chiper*

   Algoritma *block chiper* merupakan algoritma yang inputan dan keluarannya berupa satu blok dan satu bloknya terdiri dari banyak bit, misalnya 1 blok terdiri dari 64 bit atau 128 bit. Contoh algoritma *block chiper* adalah *DES (Data Encryption Standard), AES (Advanced Encryption Standard), dan IDEA (International Data Encryption Algoritm)*.

2. *Stream Chiper*

   Sandi stream yang beroperasi pada data stream sehingga operasi penyandian dilakukan per satu bit atau per satu byte pada satu waktu. Stream chiper berasal dari hasil XOR antara setiap bit *plaintext* dengan setiap bit kuncinya. Contohnya adalah algoritma OTP (*One Time Pad*), A5, dan RC4.

2.1.3 Kriptografi Asimetri

Pada pertengahan tahun 70-an Whitfield Diffie dan Martin Hellman menemukan teknik enkripsi asimetris sering disebut kriptografi asimetris atau kriptografi kunci publik yang merevolusi dunia kriptografi. Kunci asimetris adalah pasangan kunci-kunci kriptografi yang salah satunya dipergunakan untuk proses enkripsi dan yang satu lagi untuk dekripsi. Semua orang yang mendapatkan kunci publik dapat menggunakannya untuk mengenkripsikan suatu pesan, sedangkan hanya satu orang saja yang memiliki rahasia tertentu, dalam hal ini kunci privat untuk melakukan pembongkaran terhadap sandi yang dikirim untuknya.

Contoh Alice mengirim pesan untuk Bob dengan cara seperti ini, Alice dapat merasa yakin bahwa pesan tersebut hanya dapat dibaca oleh Bob, karena hanya Bob yang bisa melakukan dekripsi dengan kunci privatnya. Tentunya Alice harus memiliki kunci publik Bob untuk melakukan enkripsi. Alice bisa mendapatkannya dari Bob, ataupun dari pihak ketiga seperti Eve.

Teknik enkripsi asimetris ini jauh lebih lambat daripada enkripsi dengan kunci simetris. Oleh karena itu, biasanya bukanlah pesan itu sendiri yang disandikan dengan kunci asimetris, namun hanya kunci simetrislah yang disandikan dengan kunci asimetris. Sedangkan pesannya dikirim setelah disandikan dengan kunci simetris tadi. Contoh algoritma terkenal yang menggunakan kunci asimetris adalah RSA (merupakan singkatan penemunya yakni *Rivest, Shamir dan Adleman*) dan DSA (*Digital signature Algorithm*) [1].

Kriptografi kunci publik yang digunakan pada tandatangan digital dapat dinyatakan sebagai:

$E_{kd}(M) = C$

$D_{ke}(C) = M$

dimana:

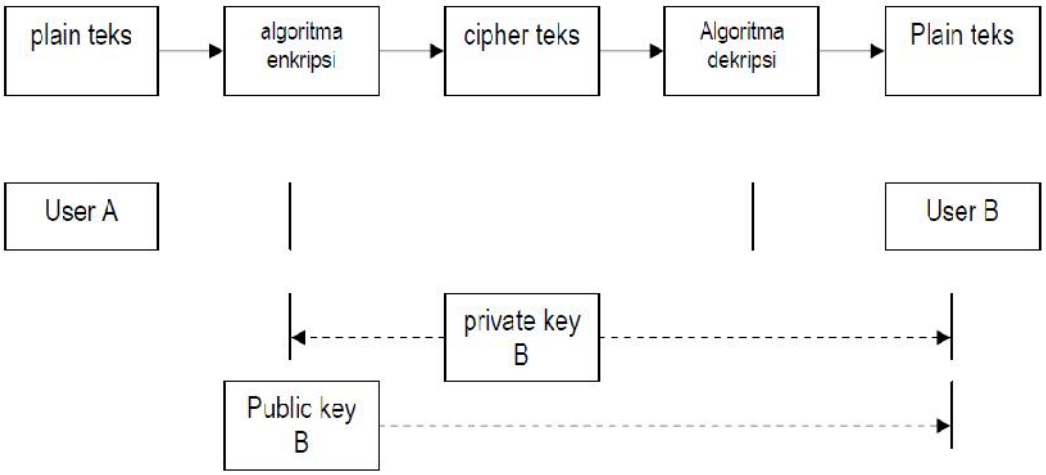
E = enkripsi, D = dekripsi, M = pesan, C = cipher, kd = kunci privat, ke = kunci public [8].

Kunci untuk enkripsi disebut juga sebagai kunci publik dan berifat tidak rahasia sehingga dapat didistribusikan melalui saluran yang tidak aman sekalipun. Sedangkan kunci untuk dekripsi disebut juga dengan kunci *Private* dan bersifat rahasia dan harus dijaga kerahasiaannya oleh pemegang kunci.

Dalam sistem kriptografi klasik selalu mengasumsikan pihak yang mengenkripsi dan pihak yang mendekripsi memiliki kunci rahasia yang sama misalnya K. Sistem kriptografi seperti ini dinamakan kriptografi dengan kunci simetri, dimana kunci rahasia K harus dibangkitkan secara rahasia dan didistribusikan ke pengenkripsi dan pendekripsi melalui saluran yang diasumsikan aman. Pada praktiknya mendapatkan saluran aman adalah hal yang sulit apalagi sekarang setiap aplikasi membutuhkan jaringan terbuka seperti internet yang dikategorikan sebagai jaringan tidak aman.

Kriptografi asimetrik (*asymmetric cryptography*) adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsi. Kunci enkripsi dapat disebarkan kepada umum dan dinamakan sebagai kunci publik (*Public Key*) sedangkan kunci dekripsi disimpan untuk digunakan sendiri dan dinamakan sebagai kunci pribadi (*Private Key*). Oleh karena itulah, Kriptografi ini

dikenal pula dengan nama Kriptografi kunci publik (*Public Key cryptography*). Contoh algoritma terkenal yang menggunakan kunci asimetris adalah RSA (*Riverst Shamir Adleman*) dan ECC (*Elliptic Curve Cryptography*).



Gambar 2.3 Proses Kriftografi Asimetris

2.1.4 Protokol Kriptografi

Protokol merupakan aturan yang berisi rangkaian langkah-langkah, yang melibatkan dua atau lebih orang, yang dibuat untuk menyelesaikan suatu kegiatan. Sedangkan protokol kriptografi adalah protokol yang menggunakan kriptografi. Orang yang berpartisipasi dalam protokol kriptografi memerlukan protokol tersebut misalnya untuk:

1. Berbagai komponen rahasia untuk menghitung sebuah nilai
2. Membangkitkan rangkaian bilangan acak
3. Menyakinkan identitas orang lain (otentikasi)

Protokol kriptografi dibangun dengan melibatkan beberapa algoritma kriptografi. Sebagian besar protokol kriptografi dirancang untuk dipakai oleh kelompok yang terdiri dari 2 orang pemakai, tetapi ada juga beberapa protokol yang dirancang untuk dipakai oleh kelompok yang terdiri dari lebih dari dua orang pemakai.

Protokol Komunikasi dengan Sistem Kriptografi Kunci Publik Berikut urutannya:

1. Bob dan Alice menyepakati algoritma kriptografi kunci publik yang akan digunakan.
2. Bob mengirimi Alice kunci publiknya (kunci publik Bob).
3. Alice mengenkripsi pesannya dengan kunci publik Bob kemudian mengirimkannya kepada Bob.
4. Bob mendekripsi pesan dari Alice menggunakan kunci privat miliknya (kunci privat Bob).

Pada umumnya, pengguna di jaringan menyepakati algoritma kriptografi kunci publik yang digunakan. Setiap pengguna jaringan mempunyai kunci publik dan kunci privat, yang dalam hal ini kunci publik dipublikasikan melalui basis data yang dapat diakses bersama. Dengan demikian, protokol kriptografi kunci publik menjadi lebih sederhana sebagai berikut:

1. Alice mengambil kunci publik Bob dari basisdata kunci-publik.
2. Alice mengenkripsi pesannya dengan kunci publik Bob kemudian mengirimkanya kepada Bob.
3. Bob mendekripsi pesan dari Alice dengan kunci privat miliknya (kunci privat Bob).

2.2 RSA (Ron Rivest, Adi Shamir, dan Len Adleman)

Algoritma kriptografi RSA merupakan algoritma kriptografi kunci publik (asimetris). Ditemukan pertama kali pada tahun 1977 dari MIT (Massachusetts Institute of Technology) yang merupakan algoritma standar Internasional dan bisa diterapkan untuk *digital signature*, pertukaran kunci (*Key exchange*), dan enkripsi. Huruf RSA itu sendiri berasal dari inisial nama mereka (Rivest—Shamir—Adleman). RSA dikembangkan tahun 1978 di MIT yang menyediakan otentikasi dan enkripsi kunci. Sebagai algoritma kunci publik, RSA mempunyai dua kunci, yaitu kunci publik dan kunci privat. Kunci publik boleh disebarkan dan diketahui oleh siapa saja, dan digunakan untuk proses enkripsi. Sedangkan kunci privat bersifat rahasia hanya pihak-pihak tertentu saja yang boleh mengetahuinya, dan

digunakan untuk proses dekripsi. Pertama, dokumen di-*hash* dan menghasilkan *message digest*. Kemudian, *message digest* dienkripsi oleh kunci publik menjadi *digital signature*. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci privat. Sampai saat ini RSA masih dipercaya dan digunakan secara luas di internet. RSA terdiri dari tiga proses, yaitu proses pembentukan kunci (*Key generation algorithm*), proses enkripsi (*encryption*) dan proses dekripsi (*decryption*).

Tabel 2.1 Besaran Algoritma RSA

1.	p dan q merupakan bilangan prima yang besar	(rahasia)
2.	n= p.q (modulus)	(tidak rahasia)
3.	φ(n) = (p-1)(q-1) (fungsi euler)	(rahasia)
4.	e (kunci publik)	(tidak rahasia)
5.	d (kunci privat)	(rahasia)
6.	m (plainteks)	(rahasia)
7.	c (cipherteks)	(tidak rahasia)

2.2.1 Oprasional RSA

RSA merupakan metode dalam kriptografi yang menggunakan *Public Key*. RSA terdiri dari dua bagian utama yaitu:

- *Public Key* (untuk enkripsi)
- *Private Key* (untuk dekripsi).

Bagian *Public Key* dapat dipublikasikan ke khlayak umum sehingga publik dapat melakukan enkripsi data sedangkan bagian *Private Key* tidak dipublikasikan dan tetap dijaga rahasianya untuk melakukan dekripsi data. Pesan yang dienkripsi dengan *Public Key* tertentu hanya dapat didekripsi dengan *Private Key* yang bersesuaian.

Secara garis besar, algoritma RSA melibatkan perkalian dua bilangan prima yang besar dan dengan tambahan operasi matematika lainnya menghasilkan

kedua buah set kunci seperti di atas. Jika kunci sudah disebarkan bilangan prima yang pertama tidak lagi penting dan dapat diabaikan.

Pembuatan kunci dilakukan dengan pemilihan bilangan acak yang besar lalu dibuat *Public Key* (dipublikasikan untuk umum) dan *Private Key* (dirahasiakan).

Setelah publik mendapatkan *Public Key* yang telah dipublikasikan maka pihak tersebut dapat melakukan enkripsi pesan kepada tujuannya. Penerima pesan akan mendekripsi pesan tersebut dengan *Private Key* miliknya.

Dengan menggunakan *Private Key* untuk mengenkripsi *signature* maka siapa saja dapat melakukan dekripsi dengan *Public Key*. Dengan demikian publik dapat mengetahui siapa pengirim pesan yang ia terima. Validasi pesan tergantung pada sistem keamanan dalam *Private Key.*

Dengan menggabungkan *Private Key* dengan *Public Key* dari orang lain maka kedua pihak dapat membagi pesan rahasia yang hanya diketahui oleh kedua pihak. Proses enkripsi dan dekripsi dalam penyampaian pesan seperti di atas di gambarkan dalam tabel berikut:

Tabel 2.2 Proses Enkripsi dan Dekripsi

Aksi	**Pemilik Kunci**	**Kunci**
Kirim pesan terenkripsi	Penerima	*Public Key*
Kirim *signature* terenkripsi	Pengirim	*Private Key*
Dekripsi pesan	Penerima	*Private Key*
Dekripsi dan enkripsi *signature*	Pengirim	*Public Key*

2.2.2 Fungsi *Totient Euler φ*

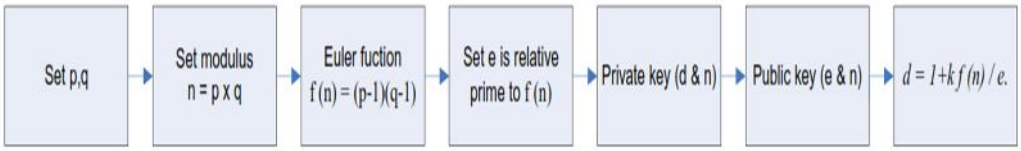
Fungsi *totient euler* atau biasa disebut dengan fungsi euler merupakan salah satu fungsi yang dipakai dalam perhitungan matematis pada algoritma RSA. Fungsi euler mendefinisikan *f(n)* untuk $n = 1$ yang menyatakan jumlah bilangan bulat positif $< n$ yang relatif prima dengan n. Dua bilangan bulat a dan b

dikatakan relatif prima jika *gcd*(a,b) =1(pembagi bersama terbesar dari *a* dan *b* adalah 1)

- Jika $n = pq$ (*p* dan *q* bilangan prima), maka $f(n) = f(p)\ f(q) = (p-1)(q-1)$.
- Contoh: $f(15) = f(3)\ f(5) = 2x4 = 8$ buah bilangan bulat yang relatif prima terhadap 15, yaitu 1,2,4,7,8,11,13,14

2.2.3 Algoritma Pembangkitan Kunci

Dalam proses pembangkitan kunci baik kunci publik maupun kunci privat pada algoritma RSA, dapat dilakukan dengan langkah-langkah sebagai berikut:



Gambar 2.4Proses Pembangkitan Kunci Algoritma RSA

Berikut ini akan djelaskan lebih lanjut tentang proses pembangkitan kunci pada algoritma RSA.

Misalkan Alice ingin Bob mengirimnya sebuah pesan melalui jalur yang aman. Alice akan memberikan *Public Key* nya kepada Bob dan menyimpan *Private Key* untuk dirinya:

1. Pilih dua buah bilangan prima secara random yakni p dan q akan tetapi nilai $p \neq q$.

2	3	5	7	11	13
17	19	23	29	31	37
41	43	[redacted]	53	59	61
67	[redacted]	73	79	83	….

2. Hitung n = *p.q*, sehingga nilai n = 47 x 71 adalah **3337**
3. Hitung f(n) = (p-1)(q-1), sehingga nilai f(n) = (47-1)(71-1) adalah **3220**
4. Pilih kunci publik (*e*), sehingga nilai *e* relatif prima terhadap f(n)

2	3	5	7	11	13
17	19	23	29	31	37
41	43	47	53	59	61
67	71	73	[redacted]	83	….

Misalkan ditentukan nilai *e* secara acak yang relatif prima terhadap nilai f(n) adalah 79.

79 relatif prima terhadap 3220. maka nilai *e* adalah **79**.

5. Hitung kunci privat (*d*) dengan menggunakan persamaan *d e = 1 (mod f(n))*. Perhatikan bahwa *d e = 1 (mod f(n))* ekivalen dengan *e.d = 1 + k f(n)*, sehingga secara sederhana *d* dapat dihitung dengan *d = 1+k f(n) / e.* Dengan rumus tersebut maka di dapat nilai d = {1+(*k* 3220)} / 79. *k*=1,2,3,4,.... Dengan mencoba nilai-nilai k = 1,2,3,..... sehingga diperoleh d yang bulat, dipilih k = 25 menghasilkan d=**1019**.

   Public Key = (e,n) = (79,3337)

   Private Key = (d,n) = (1019,3337)

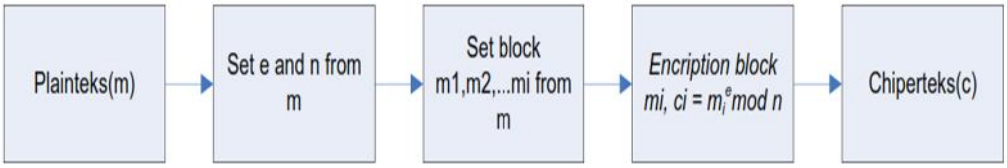
Pada kunci publik terdiri atas:

- n, modulus yang digunakan.
- e, kunci publik, kunci untuk enkripsi

Pada kunci privat terdiri atas:

- n, modulus yang digunakan.
- *d*, kunci privat, kunci untuk *dekripsi*, yang harus dijaga kerahasiaannya.

2.2.4 Proses Enkripsi Pesan



Gambar 2.5 Proses Enkripsi Algoritma RSA

Dalam pengenkripsian pesan dalam RSA dapat dimisalkan Alice ingin mengirim sebuah pesan (m) ke Bob. Maka Alice harus melakukan beberapa langkah sebagai berikut:

1. Alice menentukan kunci publik (e) dan modulus (n) dari pesan terlebih dahulu yakni nilai e = 17 dan nilai n =3337.
2. Kemudian Alice memecah *plainteks* atau pesan (*m)* menjadi blok-blok *m1,m2,....* sedemikian sehingga setiap blok merepresentasikan nilai di dalam selang [0, n-1]. Misal pesan yang akan dikirimkan adalah HARI INI. Sebelum memecah *plainteks* menjadi blok, Alice mengubah *plainteks* dalam kode ASCII yakni 7265827332737873. Kemudian, Alice memecah *plainteks* menjadi beberapa blok yang lebih kecil dan nilai-nilai m_i harus terletak dalam selang [0,3337-1] agar tranformasi menjadi satu-ke-satu,misal *m* dipecah menjadi 2 blok yang berukuran 3 digit :
3. Setiap blok m_i di*enkripsi* menjadi blok c_i dengan rumus $c_i = m_i^e \ mod \ n$ yakni Bob memberikan kunci publik ke pada Alice, *e* = 79 dan *n* = 3337. Alice melakukan *enkripsi* setiap blok pesan sebagai berikut:

$m_i^{\ e}$ mod n	$= c_i$

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

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

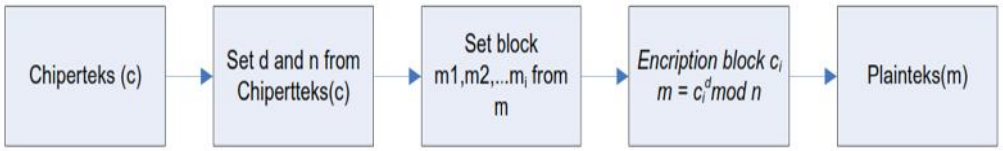
$773^{79} \bmod 3337 = 1743 = c_3$

$273^{79} \bmod 3337 = 933 = c_4$

$783^{79} \bmod 3337 = 1731 = c_5$

$003^{79} \bmod 3337 = 158 = c_6$

2.2.5 Proses Dekripsi Pesan



Gambar 2.6 Proses Dekripsi Algoritma RSA

Bob menerima cipherteks (*c*) dari Alice. Kemudian Bob melakukan *dekripsi* pesan dari Alice yang masih berupa Chiperteks. Setiap blok cipherteks (*c)* kembali menjadi blok m_i dengan rumus $m_i = c_i^d$ di*dekripsi mod n. Dekripsi* dilakukan dengan menggunakan kunci *Private* d = 1019, kemudian blok–blok chiperteks di*dekripsi*kan sebagai berikut:

$c_i^{\ d}$ mod n = m_i

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

$776^{1019} \bmod 3337 = 582 = m_2$

$1743^{1019} \bmod 3337 = 733 = m_3$

$993^{1019} \bmod 3337 = 273 = m_4$

$1731^{1019} \bmod 3337 = 787 = m_5$

$158^{1019} \bmod 3337 = 003 = m_6$

Blok *plainteks* yang lain dikembalikan dengan cara yang serupa. Akhirnya kita memperoleh kembali *plainteks* semula.

m = 7265827332737873

yang dalam karakter ASCII adalah

m = HARI INI

2.2.6 Padding Schemes

Padding scheme adalah protocol yang telah disepakati bersama untuk mengubah data atau pesan ke dalam bentuk numerik.

Padding Scheme harus dibangun secara hati-hati sehingga tidak ada nilai dari m yang menyebabkan masalah keamanan. Sebagai contoh, jika kita ambil contoh sederhana dari penampilan ASCII dari m dan menggabungkan bit-bit secara bersama-sama akan menghasilkan n, kemudian pessan yang berisi ASCII tunggal karakter NUL (nilai numeris 0) akan menghasilkan n= 0, yang akan menghasilkan ciphertext 0 apapun itu nilai dari e dan N yang digunakan. Sama halnya dengan karakter ASCII tunggal SOH (nilai numeris 1) akan selalu menghasilkan chiphertext 1. Pada kenyataannya, untuk sistem yang menggunakan nilai e yang kecil, seperti 3, seluruh karakter tunggal ASCII pada pesan akan disandikan menggunakan skema yang tidak aman, dikarenakan nilai terbesar n adalah nilai 255, dan 255^3 menghasilkan nilai yang lebih kecil dari modulus yang sewajarnya, maka proses dekripsi akan menjadi masalah sederhana untuk mengambil pola dasar dari ciphertext tanpa perlu menggunakan modulus N. Sebagai konsekuensinya, standar seperti PKCS didesain dengan sangat hati-hati sehingga membuat pesan asal-asalan dapat terenkripsi secara aman. Dan juga berdasar pada bagian Kecepatan, akan dijelaskan kenapa m hampir bukanlah pesan itu sendiri tetapi lebih pada message *Key* yang dipilh secara acak.

2.2.7 Pengesahan Pesan

RSA dapat juga digunakan untuk mengesahkan sebuah pesan. Misalkan Alice ingin mengirim pesan kepada Bob. Alice membuat sebuah hash value dari pesan tersebut, di pangkatkan dengan bilangan d dibagi N (seperti halnya pada deskripsi pesan), dan melampirkannya sebagai "tanda tangan" pada pesan tersebut. Saat Bob menerima pesan yang telah "ditandatangani", Bob memangkatkan "tanda tangan" tersebut dengan bilangan e dibagi N (seperti halnya pada enkripsi pesan), dan membandingkannya dengan nilai hasil

dari hash value dengan hash value pada pesan tersebut. Jika kedua cocok, maka Bob dapat mengetahui bahwa pemilik dari pesan tersebut adalah Alice, dan pesan pun tidak pernah diubah sepanjang pengiriman.

Harap dicatat bahwa padding scheme merupakan hal yang esensial untuk mengamankan pengesahan pesan seperti halnya pada enkripsi pesan, oleh karena itu kunci yang sama tidak digunakan pada proses enkripsi dan pengesahan.

2.2.8 Keamanan

Penyerangan yang paling umum pada RSA ialah pada penanganan masalah faktorisasi pada bilangan yang sangat besar. Apabila terdapat faktorisasi metode yang baru dan cepat telah dikembangkan, maka ada kemungkinan untuk membongkar RSA.

Pada tahun 2005, bilangan faktorisasi terbesar yang digunakan secara umum ialah sepanjang 663 bit, menggunakan metode distribusi mutakhir. Kunci RSA pada umumnya sepanjang 1024-2048 bit. Beberapa pakar meyakini bahwa kunci 1024-bit ada kemungkinan dipecahkan pada waktu dekat (hal ini masih dalam perdebatan), tetapi tidak ada seorangpun yang berpendapat kunci 2048-bit akan pecah pada masa depan yang terprediksi.

Semisal Eve, seorang *eavesdropper* (penguping), mendapatkan *Public Key* N dan e, dan ciphertext c. Bagimanapun juga, Eve tidak mampu untuk secara langsung memperoleh d yang dijaga kerahasiannya oleh Alice. Masalah untuk menemukan n seperti pada $n^e = c \bmod N$ di kenal sebagai permasalahan RSA.

Cara paling efektif yang ditempuh oleh Eve untuk memperoleh n dari c ialah dengan melakukan faktorisasi N kedalam p dan q, dengan tujuan untuk menghitung (p-1)(q-1) yang dapat menghasilkan d dari e. Tidak ada metode waktu polinomial untuk melakukan faktorisasi pada bilangan bulat berukuran besar di komputer saat ini, tapi hal tersebut pun masih belum terbukti.

Masih belum ada bukti pula bahwa melakukan faktorisasi N adalah satu-satunya cara untuk memperoleh n dari c, tetapi tidak ditemukan adanya metode yang lebih mudah (setidaknya dari sepengatahuan publik).

Bagaimanapun juga, secara umum dianggap bahwa Eve telah kalah jika N berukuran sangat besar.

Jika N sepanjang 256-bit atau lebih pendek, N akan dapat difaktorisasi dalam beberapa jam pada Personal Computer, dengan menggunakan perangkat lunak yang tersedia secara bebas. Jika N sepanjang 512-bit atau lebih pendek, N akan dapat difaktorisasi dalam hitungan ratusan jam seperti pada tahun 1999. Secara teori, perangkat keras bernama TWIRL dan penjelasan dari Shamir dan Tromer pada tahun 2003 mengundang berbagai pertanyaan akan keamanan dari kunci 1024-bit. Santa disarankan bahwa N setidaknya sepanjang 2048-bit.

Pada thaun 1993, Peter Shor menerbitkan Algoritma Shor, menunjukkan bahwa sebuah komputer quantum secara prinsip dapat melakukan faktorisasi dalam waktu polinomial, mengurai RSA dan algoritma lainnya. Bagaimanapun juga, masih terdapat pedebatan dalam pembangunan komputer quantum secara prinsip.

2.2.9 Pembangkitan Kunci (*Key generator*)

Menemukan bilangan prima besar p dan q pada biasanya didapat dengan mencoba serangkaian bilangan acak dengan ukuran yang tepat menggunakan probabilitas bilangan prima yang dapat dengan cepat menghapus hampir semua bilangan bukan prima.

p dan q seharusnya tidak "saling-berdekatan", agar faktorisasi fermat pada N berhasil. Selain itu pula, jika p-1 atau q-1 memeiliki faktorisasi bilangan prima yang kecil, N dapat difaktorkan secara mudah dan nilai-nilai dari p atau q dapat diacuhkan.

Seseorang seharusnya tidak melakukan metoda pencarian bilangan prima yang hanya akan memberikan informasi penting tentang bilangan prima tersebut kepada penyerang. Biasanya, pembangkit bilangan acak yang baik akan memulai nilai bilangan yang digunakan. Harap diingat, bahwa kebutuhan disini ialah "acak" dan "tidak-terduga". Berikut ini mungkin tidak memenuhi kriteria, sebuah bilangan mungkin dapat dipilah dari proses acak (misal, tidak dari pola

apapun), tetapi jika bilangan itu mudah untuk ditebak atau diduga (atau mirip dengan bilangan yang mudah ditebak), maka metode tersebut akan kehilangan kemampuan keamanannya. Misalnya, tabel bilangan acak yang diterbitkan oleh Rand Corp pada tahun 1950-an mungkin memang benar-benar teracak, tetapi dikarenakan diterbitkan secara umum, hal ini akan mempermudah para penyerang dalam mendapatkan bilangan tersebut. Jika penyerang dapat menebak separuh dari digit p atau q, para penyerang dapat dengan cepat menghitung separuh yang lainnya (ditunjukkan oleh Donald Coppersmith pada tahun 1997).

Sangatlah penting bahwa kunci rahasia d bernilai cukup besar, Wiener menunjukkan pada tahun 1990 bahwa jika p diantara q dan 2q (yang sangat mirip) dan d lebih kecil daripada $N^{1/4}/3$, maka d akan dapat dihitung secara effisien dari N dan e. Kunci enkripsi e = 2 sebaiknya tidak digunakan.

2.2.10 Kecepatan RSA

RSA memiliki kecepatan yang lebih lambat dibandingkan dengan DES dan algoritma simetrik lainnya. Pada prakteknya, Bob menyandikan pesan rahasia menggunakan algoritma simetrik, menyandikan kunci simetrik menggunakan RSA, dan mengirimkan kunci simetrik yang dienkripsi menggunakan RSA dan juga mengirimkan pesan yang dienkripasi secara simetrik kepada Alice.

Prosedur ini menambah permasalahan akan keamanan. Singkatnya, Sangatlah penting untuk menggunakan pembangkit bilangan acak yang kuat untuk kunci simetrik yang digunakan, karena Eve dapat melakukan bypass terhadap RSA dengan menebak kunci simterik yang digunakan.

2.2.11 Distribusi Kunci

Sebagaimana halnya chiper, bagaimana *Public Key* RSA didistribusi menjadi hal penting dalam keamanan. Distribusi kunci harus aman dari *man-in-the-middle attack*. Anggap Eve dengan suatu cara mampu memberikan kunci

kepada Bob dan membuat Bob percaya bahwa kunci tersebut milik Alice. Anggap Eve dapan "menghadang" sepenuhnya transmisi antara Alice dan Bob. Eve mengirim Bob *Public Key* milik Eve, dimana Bob percaya bahwa *Public Key* tersebut milik Alice. Eve dapat menghadap seluruh ciphertext yang dikirim oleh Bob, melakukan dekripsi dengan kunci rahasia milik Eve sendiri, menyimpan salinan dari pesan tersebut, melakukan enkripsi menggunakan *Public Key* milik Alice, dan mengirimkan ciphertext yang baru kepada Alice. Secara prinsip, baik Alice atau Bob tidak menyadari kehadiran Eve diantara transmisi mereka. Pengamanan terhadap serangan semacam ini yaitu menggunakan sertifikat digitalatau komponen lain dari *Public Key*.

2.2.12 Penyerangan Waktu

Kocher menjelaskan sebuah serangan baru yang cerdas pada RSA di tahun 1995: jika penyerang C, mengetahui perangkat keras yang dimiliki oleh A secara terperinci dan mampu untuk mengukur waktu yang dibutuhkan untuk melakukan dekripsi untuk beberapa *ciphertext*, maka C dapat menyimpulkan kunci dekripsi *d* secara cepat.

Penyerangan ini dapat juga diaplikasikan pada skema *signature* RSA. Salah satu cara untuk mencegah penyerangan ini yaitu dengan memastikan bahwa operasi dekripsi menggunakan waktu yang konstan untuk setiap *ciphertext* yang diproses. Cara yang lainnya, yaitu dengan menggunakan properti multipikatif dari RSA.

Cara lain penghitungan $c^d \bmod n$, A memilih nilai bilangan acak *r* terlebih dahulu dan menghitung $(r^e c^d \bmod n$. Hasil dari penghitungan tersebut ialah *rm mod n yang* kemudian hasil dari *r* dapat dihilangkan dengan perkalian dengan inversnya. Nilai baru dari *r* dipilih pada tiap *ciphertext*. Dengan teknik ini, dikenal sebagai *message blinding* (pembutaan pesan), waktu yang diperlukan untuk proses dekripsi tidak lagi berhubungan dengan nilai dari *ciphertext* sehingga penyerangan waktu akan gagal.

2.2.13 Penyerangan *ciphertext* Adaptive

Pada tahun 1998, Daniel Bleichenbacher menjelaskan penggunaan penyerangan *ciphertext* adaptive, terhadap pesan yang terenkripsi menggunakan RSA dan menggunakan PKCS #1 v1 *padding scheme*. Dikarenakan kecacatan pada skema PKCS #1, Bleichenbacher mampu untuk melakukan serangkaian serangan terhadap implementasi RSA pada protokol Secure Socket Layer, dan secara potensial dapat memperoleh kunci- kunci yang digunakan.

Oleh karena itu, para pengguna kriptografi menganjurkan untuk menggunakan *padding scheme* yang relatif terbukti aman seperti *Optimal Asymmetric Encryption Padding*, dan Laboratorium RSA telah merilis versi terbaru dari PKCS #1 yang tidak lemah terdapat serangan ini [6].

2.2.14 Kekuaatan dan Keamanan RSA

1. Keamanan algoritma *RSA* terletak pada tingkat kesulitan dalam memfaktorkan bilangan non prima menjadi faktor primanya, yang dalam hal ini $n = p \times q$.
2. Sekali n berhasil difaktorkan menjadi p dan q, maka $\phi(n) = (p - 1)(q - 1)$ dapat dihitung. Selanjutnya, karena kunci enkrispi *PK* diumumkan (tidak rahasia), maka kunci dekripsi *SK* dapat dihitung dari persamaan $PK \cdot SK \equiv 1 \pmod{\phi(n)}$.
3. Penemu algoritma *RSA* menyarankan nilai p dan q panjangnya lebih dari 100 digit. Dengan demikian hasil kali $n = p \times q$ akan berukuran lebih dari 200 digit. Menurut Rivest dan kawan-kawan, uasaha untuk mencari faktor bilangan 200 digit membutuhkan waktu komputasi selama 4 milyar tahun! (dengan asumsi bahwa algoritma pemfaktoran yang digunakan adalah algoritma yang tercepat saat ini dan komputer yang dipakai mempunyai kecepatan 1 milidetik).
4. Untunglah algoritma yang paling mangkus untuk memfaktorkan bilangan yang besar belum ditemukan. Inilah yang membuat algoritma *RSA* tetap dipakai hingga saat ini. Selagi belum ditemukan algoritma yang mangkus

untuk memfaktorkan bilangan bulat menjadi faktor primanya, maka algoritma *RSA* tetap direkomendasikan untuk menyandikan pesan.

2.3 *Digital signature* (Tandatangan Digital)

Digital signature adalah salah satu teknologi yang digunakan untuk meningkatkan keamanan jaringan. *Digital signature* memiliki fungsi sebagai penanda pada data yang memastikan bahwa data tersebut adalah data yang sebenarnya (tidak ada yang berubah). Dengan begitu, *digital signature* dapat memenuhi setidaknya dua syarat keamanan jaringan, yaitu *authenticity* dan *nonrepudiation*. *Digital signature* memungkinkan penerima informasi untuk menguji terlebih dahulu keaslian informasi yang didapat dan juga untuk meyakinkan bahwa data yang diterimanya itu dalam keadaan utuh.

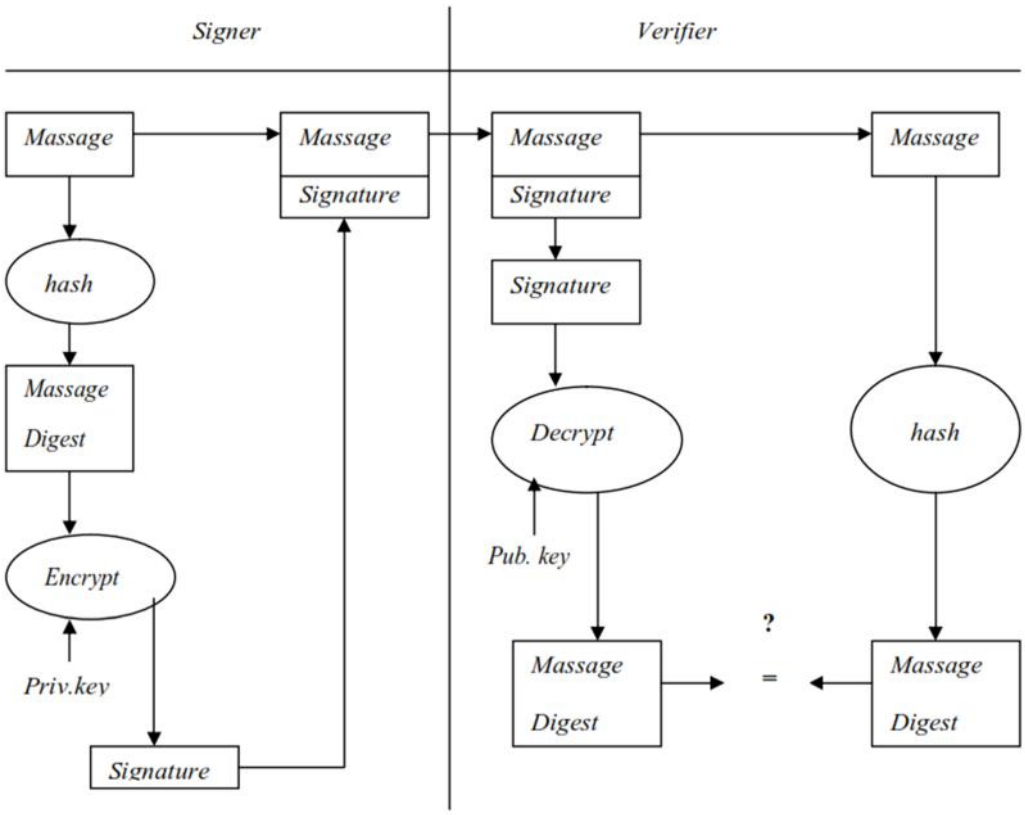
Untuk membuka *Digital signature* tersebut diperlukan kunci privat. Bila data telah diubah oleh pihak luar, maka *Digital signature* juga ikut berubah sehingga kunci privat yang ada tidak akan bisa membukanya. Ini merupakan salah satu syarat keaman jaringan, yaitu *authenticity*, artinya adalah keaslian data dapat terjamin dari perubahan-perubahan yang dilakukan pihak luar. Dengan cara yang sama, pengirim data tidak dapat menyangkal data yang telah dikirimkannya. Bila *digital signature* cocok dengan kunci privat yang dipegang oleh penerima data, maka dapat dipastikan bahwa pengirim adalah pemegang kunci privat yang sama.

Digital signature merupakan salah satu implementasi protokol kriptografi yang hasil enkripsi itulah yang digunakan sebagai tanda-tangan digital. Kemudian tanda-tangan digital digabungkan ke pesan, lalu keduanya dikirimkan lewat saluran komunikasi. Jika pada sistem Kriptografi kunci-asimetri, untuk mengenkripsi pesan menggunakan kunci-publik, sedangkan untuk mengdekripsi pesan menggunakan kunci-privat. Maka setiap orang memungkinkan untuk mengetahui kunci-publik sehingga tidak dapat digunakan untuk proses otentikasi pesan.

Pada sistem *digital signature* untuk mengenkripsi nilai hash dari pesan menggunakan kunci-privat, sedangkan untuk mendapatkan nilai hash dari pesan yang dikirimkan dengan melakukan dekripsi menggunakan kunci-publik. Dengan

melakukan pembalikan kunci tersebut maka kerahasiaan dan otentikasi pesan dapat tercapai.

Proses tanda-tangan digital terbagi menjadi dua proses, yaitu: proses pembentukan tanda-tangan digital (Signing) dan proses verifikasi tanda-tangan digital (Verifying).



Gambar 2.7 Proses Signing dan Verifikasi

Tandatangan digital dapat diberikan ke sembarang data digital, tidak hanya pesan, termasuk di dalamnya berkas program [1]. Ada dua cara yang dapat dilakukan dalam penggunaan tandatangan digital untuk berkas program. Cara pertama, tandatangan diletakkan pada berkas terpisah dari berkas executable. Namun, pemberian tandatangan dengan cara pertama ini memiliki kelemahan. Pihak ketiga yang memodifikasi program dapat membangkitkan kunci privat dan kunci publiknya sendiri. Kemudian menghitung tandatangan digital dengan menggunakan kunci privatnya, selanjutnya mengganti kunci publik

pengembang yang sah dengan kunci publiknya. Pengguna yang men-download berkas ini tidak mengetahui bahwa ia telah men-*download* program yang sudah berubah. Cara kedua yang lebih aman adalah menambahkan tandatangan digital (termasuk kunci publiknya) pada saat pengembangan program.

2.3.1 Proses Signing

Langkah-langkah dalam pembentukan tanda-tangan digital sebagai berikut:

1. Menghitung *message digest* dari pesan yang akan dikirimkan dengan menggunakan fungsi hash satu-arah. Dalam hal ini menggunakan algoritma MD5.
2. Mengenkripsi *message digest* dengan menggunakan algoritma Kriptografi kunci-asimetri. Dalam hal ini menggunakan algoritma RSA. Pengenkripsian *message digest* menggunakan kunci privat (*Private Key*) si pengirim. Hasil enkripsi inilah yang disebut sebagai tanda-tangan digital (*digital signature*).
3. Menggabungkan *digital signature* ke pesan dengan cara menyambungkannya ke bagian belakang pesan, lalu keduanya dikirim lewat saluran komunikasi.

2.3.2 Proses Verifikasi

Setelah pesan beserta tanda-tangan digitalnya diterima oleh penerima pesan, tanda-tangan digital diverifikasi untuk dibuktikan keasliannya dengan cara berikut:

1. Mendekripsi tanda-tangan digital dengan menggunakan kunci-publik (*Public Key*) si pengirim pesan. Hasil dekripsinya berupa *message digest* (MD) dari pesan semula.
2. Mengubah pesan menjadi menjadi *message digest* (MD') dengan menggunakan fungsi hash satu-arah yang sama dengan si pengirim pesan.

Mencocokan *message digest* dari pesan. Jika MD’ = MD, maka tanda-tangan digital yang diterima terbukti keasliannya/keotentikannya dan benar-benar dikirim oleh pengirim pesan yang benar.

2.3.3 Penggunaan Tanda Tangan Digital

Salah satu cara yang digunakan untuk memastikan surat tersebut adalah dengan mengecek tanda tangan yang ada di dalam surat tersebut dan stempel yang menunjukkan keaslian pengirim surat. Tanda tangan digital atau yang lebih dikenal dengan *digital signature* mempunyai fungsi yang sama dengan tanda tangan analog yang ditulis di atas kertas. Tanda tangan digital harus unik sehingga dapat membedakanpengirim yang satu degan yang lainnya. Tanda tangan digital juga harus sulit untuk ditiru dan dipalsukan sehingga integritas dan keabsahan pesan dapat terjaga. Dengan demikian diharapkan pencatutan identitas ketika pesan atau email tersebut dikirim dapat dihindari. Tidak hanya pencatutan Untuk keperluan yang penting ini, tersedia alat bantu yang dapat diperoleh secara cumacuma, yakni Pretty Good Privacy (PGP) dan Gnu Privacy Guard atau GPG. Tentu saja masih terdapat penyedia layanan tanda tangan digital lainnya, namun PGP dan GPG lebih dikenal luas. GPG adalah produk Open Source yang dapat diperoleh secara gratis tanpa harus membayar lisensi. Penggunaaan PGP di luar

Amerika Serikat harus menggunakan versi internasional. Sedangkan GPG sendiri karena dikembangkan di luar wilayah hukum Amerika Serikat, maka bebas digunakan oleh siapapun. Restriksi ini berkaitan dengan aturan ekspor produk enkripsi yang berkait dengan pemakaian kunci sandi untuk pemakaian tanda tangan digital ini [DIR04]. Penggunaan tanda tangan digital ini tidak terlalu sulit. Kedua belah pihak yang akan berkomunikasi harus menyiapkan sepasang kunci, yaitu kunci privat (*Private Key*) dan kunci publik (*Public Key*). Kunci privat hanya dipegang oleh pemiliknya sendiri. Sedangkan kunci publik dapat diberikan kepada siapapun yang memerlukannya

2.4 Biometrik

Biometrik berasal dari bahasa Yunani *bios* yang artinya hidup dan *metron* yang artinya mengukur. Secara umum biometrik adalah studi tentang karakteristik biologi yang terukur. Dalam dunia teknologi informasi, biometrik digunakan untuk menganalisa fisik dan kelakuan manusia dalam autentifikasi.

Biometrik adalah suatu metoda untuk mengenali manusia berdasar pada satu atau lebih ciri-ciri fisik atau tingkah laku yang unik. Alasan menggunakan Biometrik yaitu karena keterbatasan manusia memverifikasi segala hal hanya dari sisi:

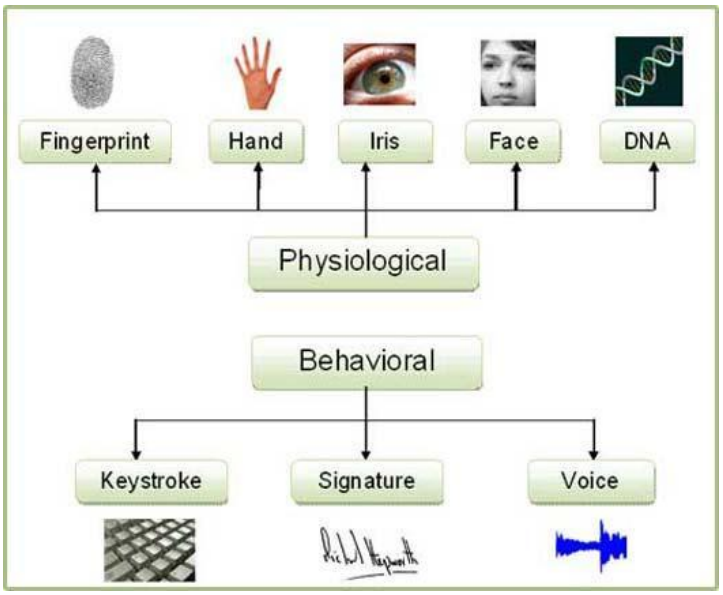
- Verifikasi berdasarkan benda: semua data-data yang dibutuhkan berada pada suatu benda (seperti dokumen atau kartu kredit). Apabila hilang maka orang lain dapat memalsukannya atau menyalahgunakannya.
- Verifikasi berdasarkan pengetahuan: biasanya menggunakan password, bahkan jika menggunakan algoritma enkripsi terbaikpun, tetap terdapat kunci yang bisa membukanya.

Keunggulan penggunaan biometrik yaitu:

- Tidak dapat hilang atau lupa
- Sulit di duplikasi, di-share ataupun dipindah tangankan
- Keaslian lebih terjamin karena harus menghadirkan person sebagai alat validasi

Karakteristik dari Biometrik yaitu:

a. Physiological : dihubungkan dengan bentuk tubuh/badan.
   Misalnya fingerprints, face recognition, DNA, hand geometry dan iris recognition
b. Behavioral : di hubungkan dengan tingkah laku seseorang.
   Misalnya *Key*stroke, *signature*, voice.



Gambar 2.8 Karakteristik Biometrik

2.4.1 Biometrik Tandatangan

Tanda tangan manual atau tanda tangan *offline* adalah tanda tangan pada dokumen cetak/ fisik (*handwritten*) yang didigitasi dengan alat pemindai (*scanner*). Tanda tangan manual tersebut menunjukkan identifikasi otomatis karena mendeskripsikan fisik unik dari seseorang yang telah diketahui, integritas dan kepercayaan untuk menjaga keotentikan isi e-dokumen dan kepercayaan pada pihak penandatangan/ *signer* yang sebenarnya.

Biometrik tandatangan didefinisikan sebagai proses menurunkan kunci privat dari sampel biometrik dan menggunakan kunci privat tersebut untuk menandatangani e-dokumen [6]. Jadi kunci privat unik dapat dibangkitkan dinamis dari salah satu sampel biometrik, tanpa memerlukan penyimpanan. Ini menghilangkan masalah tempat penyimpanan kunci privat yang memecahkan isu manajemen kunci. Pembangkitan dinamis kunci privat membuktikan kemudahan penandatanganan dokumen sebagaimana dapat menandatangani dokumen kapanpun dimanapun tanpa membawa disk atau *smart card* [9].

2.5 Image Processing (Pengolahan Citra)

2.5.1 Pengertian Pengolahan Citra (Image Processing)

Pengolahan citra adalah setiap bentuk pengolahan sinyal dimana input adalah gambar, seperti foto atau video bingkai, sedangkan output dari pengolahan gambar dapat berupa gambar atau sejumlah karakteristik atau parameter yang berkaitan dengan gambar. Kebanyakan gambar-teknik pemrosesan melibatkan atau memperlakukan foto sebagai dimensi dua sinyal dan menerapkan standar-teknik pemrosesan sinyal untuk itu, biasanya hal tersebut mengacu pada pengolahan gambar digital, tetapi dapat juga digunakan untuk optik dan pengolahan gambar analog. Akuisisi gambar atau yang menghasilkan gambar input di tempat pertama disebut sebagai pencitraan.

Pengolahan citra merupakan proses pengolahan dan analisis citra yang banyak melibatkan persepsi visual. Proses ini mempunyai ciri data masukan dan informasi keluaran yang berbentuk citra. Istilah pengolahan citra digital secara umum didefinisikan sebagai pemrosesan citra dua dimensi dengan komputer. Dalam definisi yang lebih luas, pengolahan citra digital juga mencakup semua data dua dimensi. Citra digital adalah barisan bilangan nyata maupun kompleks yang diwakili oleh bit-bit tertentu.

Umumnya citra digital berbentuk persegi panjang atau bujur sangkar (pada beberapa sistem pencitraan ada pula yang berbentuk segienam) yang memiliki lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau piksel sehingga ukuran citra selalu bernilai bulat. Setiap titik memiliki koordinat sesuai posisinya dalam citra. Koordinat ini biasanya dinyatakan dalam bilangan bulat positif, yang dapat dimulai dari 0 atau 1 tergantung pada sistem yang digunakan. Setiap titik juga memiliki nilai berupa angka digital yang merepresentasikan informasi yang diwakili oleh titik tersebut.

Format data citra digital berhubungan erat dengan warna. Pada kebanyakan kasus, terutama untuk keperluan penampilan secara visual, nilai data digital merepresentasikan warna dari citra yang diolah. Format citra digital yang banyak dipakai adalah citra biner (monokrom), citra skala keabuan (gray scale), citra warna (true color), dan citra warna berindeks.

Perangkat sistem pengolah citra dikelompokkan menjadi tiga bagian yaitu:

1. Perangkat keras (*hardware*)
2. Perangkat lunak (*software*)
3. Intelejensi manusia (*brainware*)

Ketiga pengelompokkan sistem pengolah citra tersebut sudah menjadi hal mutlak dalam pengolah citra. Dimana pada komputer-komputer saat ini sudah hampir dikatakan memenuhi standart spesifikasi untuk melakukan pengolahan citra. Namun kenyataanya masih banyak perangkat yang lainnya yang perlu kita lengkapi untuk melakukan pengolahan citra, bukan hanya sekedar komputer, melainkan perangkat-perangkat lainnya yang tidak include dalam sebuah komputer atau PC.

2.5.2 Operasi Pengolahan Citra

1. Perbaikan kualitas citra (*image enhacement*)

   Operasi perbaikan citra:

   - Perbaikan kontras gelap/terang
   - Perbaikan tepian objek (*edge enhancement*)
   - Penajaman (*sharpening*)
   - Pemberian warna semu (*pseudocoloring*)
   - Penapisan derau (*noise filtering*)

2. Pemugaran citra (*image restoration*)

   Tujuan: menghilangkan cacat pada citra.Perbedaannya dengan perbaikan citra: penyebab degradasi citra diketahui.

   Operasi pemugaran citra:

   - Penghilangan kesamaran (deblurring)
   - Penghilangan derau (noise)

3. Pemampatan citra (*image compression*)

   Tujuan: citra direpresentasikan dalam bentuk lebih simple, sehingga keperluan memori lebih sedikit namun dengan tetap mempertahankan kualitas gambar (misal dari .BMP menjadi .JPG)

4. Segmentasi citra (*image segmentation*)

Tujuan: memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu.

Berkaitan erat dengan pengenalan pola.

5. Analisis citra (*image analysis*)

   Tujuan: menghitung besaran kuantitatif dari citra untuk menghasilkan deskripsinya. Diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya

   Operasi pengorakan citra:

   - Pendeteksian tepi objek (*edge detection*)
   - Ekstraksi batas (*boundary*)
   - Represenasi daerah (*region*)

6. Rekonstruksi citra (*Image recontruction*)

   Tujuan: membentuk ulang objek dari beberapa citra hasil proyeksi.

2.5.3 Aplikasi Pengolahan Citra dan Pengenalan Pola

Pengolahan citra di berbagai bidang:

- Bidang Perdagangan
  - Pembacaan bar codepada barang di supermarket
  - Pengenalan huruf/angka pada formulir secara otomatis
- Bidang Militer
  - Mengenali peluru kendali melalui sensor visual
  - Mengidentifikasi jenis pesawat musuh
- Bidang Kedokteran
  - Deteksi kanker dengan sinar X
  - Rekonstruksi foto janin hasil USG
- Bidang Biologi
  - Penenalan kromosom melalui gambar mikroskopik
- Komunikasi Data
  - Pemampatan citra transmisi
- Hiburan
  - Pemampatan video MPEG

- Robotika
    - Visual guided autonomous navigation
- Pemetaan
    - Klasifikasi penggunaan tanah melalui foto udara
- Geologi
    - Mengenali jenis bebatuan melalui foto udara
- Hukum
    - Pengenalan sidik jari
    - Pengenalan foto narapidana

2.5.4 Fungsi Pengolahan Citra (*Image Processing*)

Pengolahan citra memiliki beberapa fungsi, diantaranya adalah:

- Digunakan sebagai proses memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia atau komputer.
- Digunakan untuk Teknik pengolahan citra dengan mentrasformasikan citra menjadi citra lain.
  contoh: pemampatan citra (*image compression*)
- Sebagai proses awal (*preprocessing*) dari komputer visi.

2.5.5 Analisis Pengolaha Citra Digital

Secara umum, pengolahan citra digital menunjukan pada pemrosesan gambar dua dimensi menggunakan komputer. Dalam konteks lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data dua dimensi. Citra digital merupakan sebuah larik (array) yang berisi nilainilai real maupun komplek yang direpresentasikan dengan deret bit tertentu.

Suatu citra dapat didefinisikan sebagai fungsi (x,y) berukuran M baris dan N kolom, dengan x dan y adalah koordinat spasial, dan amplitudo f di titik koordinat (x,y) dinamakan intensitas atau tingkat keabuan dari citra pada titik tersebut. Apabila nilai x, y, dan nilai amplitudo f secara keseluruhan berhingga (finite) dan bernilai diskrit maka dapat dikatakan bahwa citra tersebut adalah citra digital. Citra digital dinyatakan dengan matriks berukuran N x M (baris/tinggi = N, kolom/lebar = M)

N = jumlah baris $0 \leq y \leq N - 1$

M = jumlah kolom $0 \leq x \leq M - 1$

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

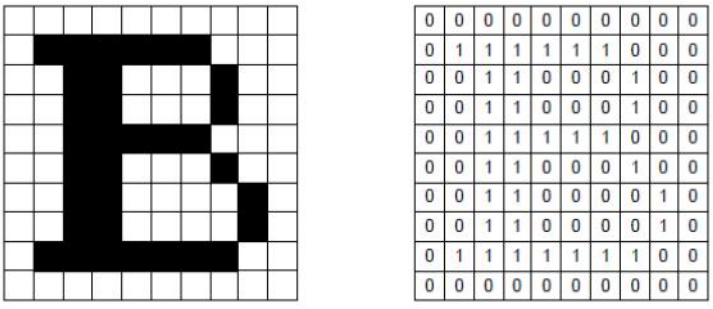
Gambar 2.9 Matriks Citra Digital

Citra digital dibentuk dari blok-blok kecil yang dinamakan piksel. Piksel merupakan elemen penyusun warna terkecil yang menyusun suatu citra. Untuk menunjukkan tingkat pencahayaan piksel, digunakan bilangan bulat yang besarnya 8 bit (1 byte) untuk setiap pikselnya, dengan lebar selang antara 0-255, dimana 0 untuk warna hitam, 255 untuk warna putih, dan tingkat keabuan ditandai dengan nilai diantara 0-255.

Citra digital dapat dikualifikasikan berdasarkan sifat dari nilai terkuantisasinya sebagai berikut:

a. Citra Biner

   Citra biner (binary image) adalah citra yang hanya mempunyai dua nilai derajat keabuan: hitam dan putih. Pixel-pixel objek bernilai 1 dan pixel-pixel latar belakang bernilai 0. Pada waktu menampilkan gambar, 0 adalah putih dan 1 adalah hitam. Jadi, pada citra biner, latar belakang berwarna putih sedangkan objek berwarna hitam.

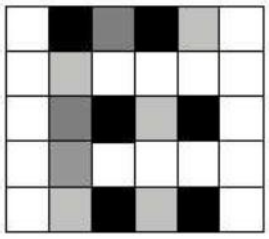


0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	1	1	1	0	0	0
0	0	1	1	0	0	0	1	0	0
0	0	1	1	0	0	0	0	1	0
0	0	1	1	0	0	0	0	1	0
0	1	1	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0

Gambar 2.10 Citra Biner dan Representasi Citra Biner

b. Citra *Grayscale*

Citra *grayscale* disebut juga dengan citra 8-*bit* karena memiliki 2^8 (256) kemungkinan nilai pada masing-masing pikselnya.Nilai tersebut dimulai dari nol untuk warna hitam dan 255 untuk warna putih.Citra ini disebut juga citra hitam putih atau citra *monoakromatik.* Skala keabuan 4 *bit* jumlah kemungkinan 2^4= 16 warna, kemungkinan warna 0 (*min*) sampai 15 (*max*). Skala keabuan 8 *bit* jumlah kemungkinan 2^8 = 256 warna kemungkinan warna 0 (*min*) sampai 255 (*max*).



= 15 0 6 0 13 15
= 15 12 15 15 15 15
= 15 5 0 12 0 15
= 15 8 15 15 15 15
= 15 10 0 13 0 15

Gambar 2.11 Skala Keabuan 4 bit

c. Citra Warna (RGB)

Citra warna adalah citra yang nilai piksel-nya merepresentasikan warna tertentu. Banyaknya warna yang mungkin digunakan bergantung kepada kedalaman pixel citra yang bersangkutan. Citra berwarna direpresentasikan dalam beberapa kanal (channel) yang menyatakan komponen-komponen warna penyusunnya. Banyaknya kanal yang digunakan bergantung pada model warna yang digunakan pada citra tersebut.

Intensitas suatu pada titik pada citra berwarna merupakan kombinasi dari tiga intensitas : derajat keabuan (merah fmerah(x,y)), hijau fhijau(x,y) dan biru (fbiru(x,y)). Persepsi visual citra berwarna umumnya lebih kaya di bandingkan dengan citra hitam putih. Citra berwarna menampilkan objek seperti warna aslinya (meskipun tidak selalu tepat demikian).Warna-warna yang diterima oleh mata manusia merupakan hasil kombinasi cahaya dengan panjang gelombang berbeda.

2.5.5.1 Proses Penghitungan Nilai *Pixel* Citra

Untuk melakukan penyisipan pesan ke dalam file citra, terlebih dahulu dilakukan penghitungan nilai pixel (Px) citra dengan rumus:

Nilai R = Px Mod 256

Nilai G = (Px and 256) Mod 256

Nilai B = (Px and 256 and 256) Mod 256

Proses pertama adalah mengambil nilai R, G dan B dari suatu citra bertipe RGB. Pada tipe .bmp citra direpresentasikan dalam 24 bit, sehingga diperlukan proses untuk mengambil masing-masing 3 kelompok 8 bit dari 24 bit tadi. Sebagai contoh suatu pixel memiliki nilai RGB 24 bit sebagai berikut :

Sebagai contoh nilai piksel (1,1) Citra adalah 111100001111000011111111 (15790335).

Nilai komponen Red (R) dilakukan perhitungan modulo dengan bilangan 256 dengan nilai ASCII 100000000 sebagai berikut:

Nilai komponen R dihitung dengan persamaaan (3.1) = nilai piksel Blok-1 mod 10000000.

R = 111100001111000011111111 mod 100000000

R = 11111111

Nilai komponen Green (G) dihitung dengan persamaan (3.2):

G = (111100001111000011111111 and 100000000) mod 100000000

G = 11110000

Nilai komponen Blue (B) dihitung dengan persamaan (3.3):

B = (111100001111000011111111 and 10000000 and 10000000) mod 10000000

B = 11110000

Sehingga diperoleh nilai RGB piksel citra frame-1 pada Blok-1 adalah:

11110000 11110000 11111111:

R = 11111111= 255, G = 11110000 = 240 dan B = 11110000 = 240

Sehingga diperoleh nilai piksel (1,1) dengan komponen RGB = (255,240,240).

Untuk perhitungan nilai piksel selanjutnya adalah sama dengan di atas dan hasil perhitungan digambarkan seperti Tabel 2.3

Tabel 2.3 Matriks Citra RGB

255,240,240	200,200,45	210,230,101	190, 200,20	100, 190,100
169,190,210	187,180,74	50,120,56	150,150,200	100,150,10
145,200,67	201,160,55	80,110,60	150,190,205	125,190,157
178,204,45	196,120,50	70,100,45	150,200,158	180,200,158
187,200,120	106,100,150	170,100,100	150,210,133	90,100,138

2.5.5.2 Proses Konversi Grayscale

Sebelum melakukan proses penyisipan citra, maka nilai RGB per piksel yang diperoleh di atas di konversi ke nilai grayscale dengan cara mencari nilai rata-rata per piksel.

$$f_0\ x,y\ = \frac{f^R\ x,y\ + f^G\ x,y\ + f^B\ x,y}{3} \quad \text{..... (3.4)}$$

Dimana:

f^R adalah nilai komponen *Red*

f^G adalah nilai komponen *Green*

f^B adalah nilai komponen *Blue*

Dengan menggunakan persamaan (3.4),

1. Piksel f(1,1)= (255,240,240) = (254+240+240)/3 = 245
2. Piksel f(1,2)= (200,200,45) = (200+200+45)/3 = 148.33 = 148
3. Piksel f(1,3)= (210,230,101) = (210,230,101)/3 = 180.33 = 180
4. Piksel f(1,4)= (190, 200,20) = (190, 200,20)/3 = 136.6 = 137
5. Piksel f(1,5)= (100, 190,100) = (100, 190,100)/3 = 130

Untuk piksel selanjutnya perhitungan nilai grayscale sama seperti cara di atas, hasilnya berupa matrik nilai grayscale seperti pada Tabel 3.2

Tabel 2.4 Matriks Citra Grayscale

245	148	180	137	130
180	180	137	150	200
137	130	135	178	120
130	148	200	125	198

180	137	125	120	145

2.6 Fungsi Hash

Didalam sistem Kriptografi, selain ada algortima enkripsi/dekripsi yang mempunyai tujuan untuk kerahasiaan data, juga ada suatu fungsi yang mempunyai tujuan untuk menjaga integritas dan otentikasi data. Fungsi itu disebut dengan fungsi hash. Fungsi hash merupakan fungsi yang menerima masukan untuk diubah menjadi keluaran yang mempunyai panjang yang tetap (biasanya lebih pendek dari masukannya). Keluaran dari fungsi hash disebut dengan nilai hash atau *message digest*.

Salah satu cara untuk menguji integritas sebuah data adalah dengan memberikan "checksum" atau tanda bahwa data tersebut tidak berubah. Cara yang paling mudah dilakukan adalah dengan menjumlahkan karakter-karakter atau data-data yang ada sehingga apabila terjadi perubahan, hasil penjumlahan menjadi berbeda. Cara ini tentunya mudah dipecahkan dengan menggunakan kombinasi data yang berbeda akan tetapi menghasilkan hasil penjumlahan yang sama.

Pada sistem digital biasanya ada beberapa mekanisme pengujian integritas seperti antara lain:

- parity checking
- checksum
- hash function

Fungsi Hash merupakan sebuah algoritma yang mengubah text atau message menjadi sederetan karakter acak yang memiliki jumlah karakter yang sama. Hash juga termasuk salah satu bentuk teknik kriptografi dan dikategorikan sebagai kriptografi tanpa *Key* (*unKeyed cryptosystem*). Selain itu hash memiliki nama lain yang juga dikenal luas yaitu "*one-way function*".

Kita sering sekali menjumpai hash di website-website yang menyediakan layanan untuk download file ataupun program secara resmi. Hash memang umumnya digunakan untuk mengecek integritas dari sebuah pesan atau file. File atau pesan yang sudah berubah akan memiliki nilai hash yang berbeda. Sebagai

contoh, dengan sebuah algoritma hash, pesan'hello' akan memberikan nilai hash 12345 sedangkan pesan 'hallo' memiliki nilai hash83746. Dengan kata lain output hash dari kata 'hello' tidak akan sama dengan 'hallo'. Bahkan sekalipun dalam kacamata kita kedua pesan tersebut terlihat hanya memiliki perbedaan sedikit saja, namun nilai hash yang dimiliki oleh kedua pesan tersebut sangat jauh berbeda.

Berbeda dengan teknik enkripsi dalam kriptografi, tujuan hash memang mengubah sebuah pesan yang dapat dibaca (*readable* text) menjadi pesan acak (*unreadable* text) sama seperti enkripsi, namun hal mendasar yang menjadi perbedaan dari hash adalah pesan yang telah acak tadi tidak dapat diubah kembali menjadi pesan yang seharusnya. Inilah mengapa hash disebut juga sebagai "*one-way function*".

Misalkan ada pesan 'Hello', pesan ini akan kita hash dengan algortima yang sederhana, yaitu pertama-tama huruf-huruf tersebut akan kita ubah kedalam bilangan angka.

- a menjadi 1
- b menjadi 2
- c menjadi 3
- dst…

Sehingga pesan 'hello' akan menjadi '8.5.12.12.15'. Kemudian kita jumlahkan bilangan-bilangan tersebut sehingga kata 'hello' akan menghasilkan jumlah 52.

h		e		l		l		o		
8	+	5	+	12	+	12	+	15	=	52

Kemudian langkah terakhir adalah kita ambil satu digit yang paling belakang sebagai nilai hashnya, yaitu 2. Nah nilai hash 2 ini bukankah bisa dibuat dari banyak kombinasi huruf? jumlah huruf pun juga bisa bervariasi, tidak harus 5 seperti 'hello'.

Tabel 2.5 Contoh text yang memiliki hash 2

Text	**Integer**	**Sum**	**Hash**
bye	2.25.5	32	2
confidential	3.15.14.6.9.4.5.14.20.9.1.12	112	2
enemy	5.14.5.13.25	62	2

Perlu dicatat disini, bahwa tabulasi diatas adalah ilustrasi bagaimana hash dikatakan sebagai sebuah fungsi yang tidak dapat di dekripsi. Dalam contoh diatas hasil hash sudah sangat acak sehingga kita tidak dapat mengetahui lagi apakah sebuah hash yang ingin kita dekripsi memiliki kandungan huruf e atau z atau m atau lainnya.

2.6.1 Kegunaan

Kegunaan dari hash Menurut Kaufman et. al. (2002) adalah sebagai:

1. **Menyimpan Password**

   Password didalam sebuah sistem dianjurkan untuk disimpan dengan menggunakan fungsi hash. Dengan demikian administrator sistem tersebut sekalipun tidak akan dapat melihat atau menggunakan password user yang telah menjadi membernya. Hal ini akan memberikan impact yang baik kepada user bahwa baik sistem maupun administrator sangat menghargai privasi dari anggotanya.

   Selain itu pula, jika database password pengguna, dengan cara tertentu dapat diakses oleh publik, maka siapapun tetap tidak langsung dapat menggunakan password tersebut. Untuk menggunakan password yang telah *Public*ly accessible terlebih dahulu seorang hacker harus melakukan buteforce terhadap kumpulan password-password teresbut.

2. **Sebagai *Message Integrity***

   Andaikata Alice ingin mengirimkan pesan kepada Bob. Untuk mencegah ada seseorang ditengah perjalanan yang ingin mengganti pesan tersebut, Alice melakukan hash terhadap pesannya sendiri yang kemudian dikirim bersama dengan pesannya yang asli. Namun,

ternyata apa yang Alice lakukan tidak menjamin integritas keamanan pesan miliknya. Seseorang tetap dapat mengubahnya dan menyediakan fungsi hash dari pesan tersebut

Alice bermaksud mengirimkan pesan ‘Hello Bob’ kepada Bob. Alice kemudian mengirimkan pesan tersebut bersama dengan fungsi hash MD5 dari pesannya, yaitu b4c9c3086946666f7ec8014629e105f7. Alice mengirimkan pesan dan hash MD5 nya melalui jasa pengantar barang, TIKI. Ditengah perjalanan, Eve berhasil mencuri pesan Alice untuk Bob dari sang kurir TIKI. Eve membuat pesan baru yang berbunyi ‘Hello Honey’, membuat hash MD5 dari pesannya tersebut, dan mengembalikannya kedalam tas sang Kurir tanpa sepengetahuan sang Kurir.

Kurir yang baik hati tersebut memberikan pesan Alice tadi kepada Bob. Ketika Bob membacanya, Bob merasakan ada yang aneh terhadap pesan tersebut. Bob mengecek hash MD5 dari pesan yang dibacanya, namun pengecekan MD5 pesan yang diterima Bob dengan MD5 yang dikirim bersama paket tersebut cocok. Tapi apakah Bob tahu pesan yang bertuliskan ‘Hello Honey’ tersebut memang berasal dari Alice?

Dalam kasus ini tentu saja Bob tidak memiliki cara untuk mengetahui keabsahan pesan tersebut hanya dari paket yang berisi pesan dan MD5 yang katanya dikirim oleh Alice. Hal ini tidak akan terjadi jika Alice dan Bob menggunakan sejumlah karakter rahasia yang hanya diketahui oleh kedua orang tersebut. Karakter-karakter rahasia ini disebut juga dengan nama *Key*ed hash. Penggunan karakter ini disebut sebagai message integrity, dimana cara pengunaannya adalah sebelum dilakukannya hash, Alice terlebih dahulu menggabungkan pesannya dengan karakter rahasia tersebut.

Jika pesan Alice adalah ‘Hello Bob’ dan karakter rahasia Alice dan Bob adalah 1234, maka hasil hash yang dibuat oleh Alice akan bernilai 508e1ae04417ccb03953aa2a320d1714. Jika Eve berhasil mencuri

pesan Alice tersebut dan menggantinya dengan pesan miliknya sendiri, dan karena Eve tidak mengetahui karakter rahasia milik Alice dan Bob, Eve berasumsi bahwa MD5 yang dikirim oleh Alice tidak mengikutsertakan *Key*ed hash sehingga ia hanya melakukan MD5 terhadap pesannya 'Hello honey' tersebut. Bob yang kemudian menerima pesan yang kata sang kurir berasal dari Alice mengecek integritas dari pesan yang katanya dari Alice tersebut dengan cara menambahkan *Key*ed hash miliknya kedalam pesan yang diterima tersebut. Setelah dilakukan hash terhadap pesan 'Hello Honey1234' Bob dengan yakin dapat mengatakan bahwa pesan tersebut bukan berasal dari Alice.

Tabel 2.6 Contoh Keyed Hash

Pesan	***Key*ed Hash**	**MD5**
'Hello Bob'		b4c9c3086946666f7ec8014629e105f7
'Hello Honey'		e42afb241fceefa05e3e897fa0686f14
'Hello Bob'	1234	508e1ae04417ccb03953aa2a320d1714
'Hello Honey'	1234	7dc14667a10ce5116b50f4d8a7e1ad9c

3. Sebagai *Message Fingerprint*

   Penggunaan Hash sebagai sidik jari pesan (message fingerprint) digunakan untuk mengecek apakah file yang kita simpan masih sesuai dengan file asli ketika hash file tersebut dibuat dan belum berubah. Jika terdeteksi adanya file yang tidak memiliki hash yang sama dengan yang telah dibuat sebelumnya, maka dapat dipastikan bahwa file tersebut telah berubah. Contoh: Pada bulan Januari 2008 Alice memiliki file Bob.txt yang berisi profil mengenai Bob. Alice mengkomputasi hash dari file tersebut sehingga diperoleh:

   MD5 (Bob.txt) --> f06defdef28e15706f974f6d080b57fe

Dua tahun kemudian Alice membuka kembali file Bob.txt tersebut dan mencurigai ada yang tidak lazim dari isi file tersebut. Alice kemudian membuat hash dari file Bob.txt tersebut dan mendapatkan nilai hashnya.

MD5 (Bob.txt) --> 5f3b147ae863e541701d7011e597e98b

Alice kemudian menyadari bahwa hash file Bob.txt pada Januari 2010 berbeda dengan hash pada bulan Januari 2008. Dengan demikian, dapat dipastikan bahwa seseorang telah mengubah file tersebut antara Bulan Januari 2008 hingga Januari 2010.

Ketidak cocokan message fingerprint tidak selama disebabkan karena ada manusia yang mengubah file ataupun pesan tersebut. Terkadang message fingerprint tidak cocok antara 2 waktu dapat disebabkan oleh transfer file yang gagal, sektor pada hardisk dimana sebuah file disimpan rusak, dan lain sebagainya. Jadi, tidak selamanya manusia dapat disalahkan juga.

2.6.2 Kegunaan

Dobbertin (1996) dalam artikelnya menyebutkan bahwa dalam membuat fungsi hash harus memenuhi beberapa kriteria berikut:

1. Relative cepat dalam melakukan komputasi.
2. 2 buah pesan yang berbeda tidak boleh memiliki nilai hash yang sama.
3. Bebas dari serangan Birthday Attack (dari sebanyak 264 atau ~18 446 744 073 709 551 616 pesan kemungkinan besar didapat 2 pesan yang memiliki nilai hash yang sama).
4. 2 Pesan yang memiliki sangat sedikit perbedaan harus memiliki nilai hash yang sangat berbeda.

2.6.3 Algoritma

Hash umumnya disajikan dalam bentuk bilangan hexadecimal, yaitu kombinasi antara angka 0-9 dengan huruf a hingga f. Beberapa algoritma hash yang terkenal dan masih sering digunakan hingga saat ini yaitu:

- MD5 (*Message digest* 5)
- SHA-1 (Secure Hash Algortihm 1)
- SHA-2 (Secure Hash Algorithm 2), yang meliputi 4 fungsi hash:
- SHA-224
- SHA-256
- SHA-384
- SHA-512

Algoritma MD5 dikembangkan oleh seorang Professor MIT yang bernama Ronald L. Rivest. Istilah "MD" yang digunakan merupakan abrieviation dari *Message digest*. Perkembangan MD5 telah melalui 5 kali revisi, dimana MD generasi pertama dan kedua di desain untuk membantu algoritma RSA dalam melakukan komputasi *signature* dari pesan rahasia yang akan dikirim dan dienkripsi oleh RSA. Generasi ke tiga dan empat MD hadir karena adanya persaingan dari algortima hash lain yang bernama SNEFRU, yang memiliki keunggulan kecepatan pada proses komputasinya dibandingkan MD2. Ketika ditemukan adanya celah keamanan dari SNEFRU pada tahun 1992, ditahun yang sama ditemukan pula kelemahan MD4, yang kemudian Profesor Rivest segera menambal kelemahan tersebut dan menggantinya menjadi generasi ke lima *Message digest*, yaitu MD5. Dari kelima generasi ini, MD generasi pertama dan ketiga merupakan algoritma yang tidak dipublikasikan. Sementara spesifikasi algoritma MD2, MD4, dan MD5 terdapat dalam RFC1319, RFC1320, dan RFC1321.

Secure Hash Algoritm (SHA) dikembangkan oleh National Institute and Standard Technology (NIST) pertama kali pada tahun 1993. Generasi pertama SHA diberi nama SHA-0. Kemudian pada tahun 1995, generasi kedua SHA, SHA-1, muncul dan dipublikasikan oleh NIST dengan kode publikasi FIPS PUB 180-1. Generasi kedua SHA ini muncul dalam waktu 12 jam dari setelah dilaporkannya terdapat kelemahan dalam algoritma SHA-0. Generasi ketiga algortima SHA, SHA-2, dipublikasikan pada tahun 2001 dengan berbagai pilihan jumlah bit yang digunakan, yaitu: 224, 256, 384, dan 512. Baik SHA-1 dan SHA-2 pada dasarnya memiliki algortima yang serupa, hanya berbeda di jumlah karakter outputnya saja.

SHA-1, SHA-256, dan SHA-516 memiliki jumlah karakter output masing-masing secara berurutan adalah 40, 32, dan 64.

2.6.4 *Message digest*

Message digest adalah sebuah nilai yang berasal dari suatu data/pesan yang memiliki sifat yang unik bahwa pesan tersebut mempunyai suatu besaran tertentu yang diciptakan dengan melakukan enkripsi terhadap suatu data dengan menggunakan kriptografi satu arah (*one way crypthography*), yaitu suatu tehnik kriptografi yang terhadapnya tidak dapat dilakukan proses pembalikan (*reversed*). *Message digest* dimaksudkan untuk meningkatkan keamanan transformasi data, seperti pasword. Dalam implementasinya nilai *message digest* diperbandingkan dengan nilai yang juga berupa *message digest* guna mengetahui kesamaannya.

Jadi secara umum *message digest* digunakan untuk memeriksa integritas sebuah data. Hal tersebut bisa anda lakukan ketika anda mendownload sebuah file dari server, maka anda dapat menghitung nilai digest-nya dan membandingkannya dengan nilai hasil komputasi di server. Jika hasilnya sama, maka anda boleh yakin bahwa selama proses pengiriman, data tidak mengalami modifikasi oleh pihak lain.

2.6.5 MD5

MD5 ialah fungsi hash kriptografi yang digunakan secara luar dengan nilai hash 128- bit. Pada standard internet (RFC 1321), MD5 telah dimanfaatkan secara bermacam-macam pada aplikasi keamanan dan MD5 juga umum digunakan untuk melakukan pengujian integritas sebuah file. MD5 adalah salah satu dari serangkaian algortima *message digest* yang didesain oleh Profesor Ronald Rivest dari MIT.

Algoritma MD5 menerima sebuah input/pesan dengan panjang sebarang dan akan menghasilkan sebuah output dengan panjang tertentu, yaitu 128-bit, “fingerprint” atau “*message digest*” dari input. Secara komputasional, diduga sangat tidak mungkin untuk menghasilkan *message digest* yang sama dari dua pesan yang berbeda, ataupun mendapatkan pesan asli dari sebuah *message digest*.

Algoritma MD5 dimaksudkan untuk aplikasi *digital signature*, dimana sebuah file besar “dikompres” secara terstruktur, sebelum dienkripsi dengan metode enkripsi yang ada, umumnya *Public-Key* cryptosystem seperti RSA. Algoritma MD5 didesain agar cepat pada komputer 32-bit. Selain itu, MD5 juga tidak memerlukan tabel substitusi yang besar sehingga algoritma ini dapat dikodekan secara ringkas.

Algoritma MD5 adalah pengembangan dari algoritma MD4. MD5 lebih lambat dari MD4, namun lebih “konservatif”dari segi desain. MD5 didesain karena MD4 dirasa sudah berada pada batas akan dapat dibobol dengan serangan *cryptanalytic*. MD5 mengorbankan sedikit kecepatan untuk keamanan yang jauh lebih baik. Banyak reviewer yang memberikan saran-saran terutama pada sisi optimisasi. Algoritma MD5 diberikan secara terbuka kepada publik untuk direview dan dapat juga diadopsi untuk dijadikan sebuah standar.

Saat kerja analitik menunjukkan bahwa pendahulu MD5, yaitu MD4 mulai tidak aman, MD5 kemudian didesain pada tahun 1991 sebagai pengganti dari MD4 (kelemahan MD4 ditemukan oleh Hans Dobbertin). Pada tahun 1993, den Boer dan Bosselaers memberikan awal, bahkan terbatas, hasil dari penemuan pseudo-collision dari fungsi kompresi MD5. Dua vektor inisialisasi berbeda I dan J dengan beda 4-bit diantara keduanya.

Cara kerja *Message digest* 5 misal kita memiliki sebuah pesan dengan panjang b-bit sebagai input, dan kita ingin mendapatkan *message digest*-nya. Disini b adalah pesan dengan panjang sebarang. Panjang sebuah pesan tentu saja tidak negatif, tapi masih mungkin untuk nol. Panjang tersebut tidak harus kelipatan dari delapan dan boleh juga sangat panjang. Misal bit dari pesan tersebut ditulis sebagai

m_0 m_1 …. m_{b-1}

Preprocessing dimulai dengan penambahan padding bits sebagai berikut :

Pesan tersebut akan diberikan tambahan bit sehingga panjangnya, dalam bit, sama dengan 448, modulo 512. Penambahan selalu dilakukan, walaupun panjang dari pesan sudah sama dengan 448, modulo 512. Penambahan bit adalah sebagai berikut: sebuah bit '1' ditambahkan diikuti dengan bit '0' sebanyak yang

diperlukan sehinggan panjangnya menjadi 448, modulo 512. Minimal dilakukan penambahan sebanyak satu bit dan maksimal sebanyak 512 bit.

64-bit ditambahkan untuk merepresentasikan panjang sebenarnya dari pesan b pada hasil dari penambahan sebelumnya. Jika panjang pesan asli lebih dari 264 bit maka hanya 64 lower order bit yang dimasukkan. Lower order word untuk panjang pesan asli dimasukkan sebelum high-order word. Setelah langkah kedua ini, maka panjang dari hasilnya adalah kelipatan dari 512 bit. Atau jika dalam word, maka panjangnya akan sama dengan kelipatan dari 16 word (32-bit). Maka hasil dari dua langkah di atas kita notasikan sebagai M [0 …. N-1] dengan N adalah kelipatan dari 16.

Setelah padding, pesan terdiri dari n word M [0 … n - 1] dimana n adalah kelipatan 16. Langkah berikutnya dalam preprocessing adalah menyiapkan MD buffer sebesar 4 word:

(A, B, C, D)

dimana A merupakan lower order word. Buffer diberi nilai awal sebagai berikut (nilai dalam hexadecimal dimulai dengan lower order byte).

A: 01 23 45 67

B: 89 ab cd ef

C: fe dc ba 98

D: 76 54 32 10

Proses hashing dilakukan per blok, dengan setiap blok melalui 4 putaran. Proses hashing menggunakan 4 fungsi F, G, H, dan I yang masing-masing mempunyai input 3 word dan output 1 word:

F(X,Y,Z) = (X ^ Y) v (~X ^ Z)

G(X,Y,Z) = (X ^ Z) v (Y ^ ~Z)

H(X,Y,Z) = X xor Y xor Z

I(X,Y,Z) = Y xor (X ^ ~Z)

dimana ^ adalah bitwise and, v adalah bitwise or, xor adalah bitwise exclusive or, dan ~ adalah bitwise not (one's complement).

Pada setiap posis bit, F berperilaku sebagai: if X then Y else Z. Setiap bit dari X, Y dan Z independent dan tidak bias, maka setiap bit dari F akan independen dan tidak bisa juga.

Fungsi G, H dan I juga serupa dengan fungsi F, beroperasi pada bitwise parallel untuk menghasilkan output dari input X, Y dan Z. Serta dalam hal jika X, Y dan Z independent dan tidak bias, maka G, H dan I juga independent dan tidak bias.

Tahap ini menggunakan tabel T [1 …. 64] yang dibentuk dari fungsi sinus. Misal T[i] adalah elemen ke-i dari tabel, maka T[i] didefinisikan sebagai 4294967296 * abs (sin(i)), dimana I dalam radian.

Contoh Hash MD5:

Contoh:

Pesan: unikom

Konversi setiap 4 byte kedalam word:

unik: 6b696e75

Mengolah final bits, menambahkan padding bit 1 kemudian sisanya bit 0

Panjang pesan = 6

panjang pesan % 4 = 2, Sisa Byte : 'om' (2 byte padding) = 00806d6f

****

Proses padding bit (dalam decimal):

Menghadle akhir dari teks dan memulai pading bit = 1802071669,8416623

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0,0,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0,0,0,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0,0,0,0,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0,0,0,0,0,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0,0,0,0,0,0,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0,0,0,0,0,0,0,0,0

padding ke 448 bits (mod 512 bits) = 14 word (mod 16 word) = 1802071669,8416623,0,0,0,0,0,0,0,0,0,0,0,0

menambahkan pesan dengan panjang 64-bit = 1802071669,8416623,0,0,0,0,0,0,0,0,0,0,0,48,0

****

Block untuk setiap variabel:

Permulaan, a (0x067452301) = 67452301

b (0x0efcdab89) = efcdab89

c (0x098badcfe) = 98badcfe

d (0x010325476) = 10325476

Padding bit dimasukkan kedalam array, konversikan ke hexa:

x[0]=6b696e75

x[1]=00806d6f

x[2]=00000000

x[3]=00000000

x[4]=00000000

x[5]=00000000

x[6]=00000000

x[7]=00000000

x[8]=00000000

x[9]=00000000

x[10]=00000000

x[11]=00000000

x[12]=00000000

x[13]=00000000

x[14]=00000030

x[15]=00000000

****

Konstanta:

S11 = 7 S12 = 12 S13 = 17 S14 = 22

S21 = 5 S22 = 9 S23 = 14 S24 = 20

S31 = 4 S32 = 11 S33 = 16 S34 = 23

S41 = 6 S42 = 10 S43 = 15 S44 = 21

****

Round 1:

Lakukan operasi [abcd k s i]

a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s)

Lakukan 16 operasi berikut ini setiap roundnya:

[ABCD 0 7 1] [DABC 1 12 2] [CDAB 2 17 3] [BCDA 3 22 4]

[ABCD 4 7 5] [DABC 5 12 6] [CDAB 6 17 7] [BCDA 7 22 8]

[ABCD 8 7 9] [DABC 9 12 10] [CDAB 10 17 11] [BCDA 11 22 12]

[ABCD 12 7 13] [DABC 13 12 14] [CDAB 14 17 15] [BCDA 15 22 16]

Lakukan a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). (Catatan: <<< berarti rotate left)

****

Fungsi yang digunakan pada masukan F(b,c,d) = (b & c) | ((~b) & d)

****

FF (a, b, c, d, x[0], S11, 0xd76aa478)

FF: a=67452301, b=efcdab89, c=98badcfe, d=10325476, xk=6b696e75, s=7, ti=d76aa478

F(efcdab89, 98badcfe, 10325476)=98badcfe

a+F(b,c,d)+xk+ti=42d412ec

FF: b+(... <<< 7)=59d721aa

FF (d, a, b, c, x[1], S12, 0xe8c7b756)

FF: a=10325476, b=59d721aa, c=efcdab89, d=98badcfe, xk=00806d6f, s=12, ti=e8c7b756

F(59d721aa, efcdab89, 98badcfe)=c9edfddc

a+F(b,c,d)+xk+ti=c3687717

FF: b+(... <<< 12)=e1489de0

FF (c, d, a, b, x[2], S13, 0x242070db)

FF: a=98badcfe, b=e1489de0, c=59d721aa, d=efcdab89, xk=00000000, s=17, ti=242070db

F(e1489de0, 59d721aa, efcdab89)=4fc523a9

a+F(b,c,d)+xk+ti=0ca07182

FF: b+(... <<< 17)=c44cb720

FF (b, c, d, a, x[3], S14, 0xc1bdceee)

FF: a=efcdab89, b=c44cb720, c=e1489de0, d=59d721aa, xk=00000000, s=22, ti=c1bdceee

F(c44cb720, e1489de0, 59d721aa)=d9db95aa

a+F(b,c,d)+xk+ti=8b671021

FF: b+(... <<< 22)=ccaf90e4

FF (a, b, c, d, x[4], S11, 0xf57c0faf)

FF: a=59d721aa, b=ccaf90e4, c=c44cb720, d=e1489de0, xk=00000000, s=7, ti=f57c0faf

F(ccaf90e4, c44cb720, e1489de0)=e54c9d20

a+F(b,c,d)+xk+ti=349fce79

FF: b+(... <<< 7)=1c96cd7e

FF (d, a, b, c, x[5], S12, 0x4787c62a)

FF: a=e1489de0, b=1c96cd7e, c=ccaf90e4, d=c44cb720, xk=00000000, s=12, ti=4787c62a

F(1c96cd7e, ccaf90e4, c44cb720)=ccceb264

a+F(b,c,d)+xk+ti=f59f166e

FF: b+(... <<< 12)=0dfdbcd7

FF (c, d, a, b, x[6], S13, 0xa8304613)

FF: a=c44cb720, b=0dfdbcd7, c=1c96cd7e, d=ccaf90e4, xk=00000000, s=17, ti=a8304613

F(0dfdbcd7, 1c96cd7e, ccaf90e4)=cc968c76

a+F(b,c,d)+xk+ti=391389a9

FF: b+(... <<< 17)=21502efe

FF (b, c, d, a, x[7], S14, 0xfd469501)

FF: a=ccaf90e4, b=21502efe, c=0dfdbcd7, d=1c96cd7e, xk=00000000, s=22, ti=fd469501

F(21502efe, 0dfdbcd7, 1c96cd7e)=1dd6edd6

a+F(b,c,d)+xk+ti=e7cd13bb

FF: b+(... <<< 22)=104a2242

FF (a, b, c, d, x[8], S11, 0x698098d8)

FF: a=1c96cd7e, b=104a2242, c=21502efe, d=0dfdbcd7, xk=00000000, s=7, ti=698098d8

F(104a2242, 21502efe, 0dfdbcd7)=0df5bed7

a+F(b,c,d)+xk+ti=940d252d

FF: b+(... <<< 7)=16dcb90c

FF (d, a, b, c, x[9], S12, 0x8b44f7af)

FF: a=0dfdbcd7, b=16dcb90c, c=104a2242, d=21502efe, xk=00000000, s=12, ti=8b44f7af

F(16dcb90c, 104a2242, 21502efe)=314826f2

a+F(b,c,d)+xk+ti=ca8adb78

FF: b+(... <<< 12)=c49445b4

FF (c, d, a, b, x[10], S13, 0xffff5bb1)

FF: a=21502efe, b=c49445b4, c=16dcb90c, d=104a2242, xk=00000000, s=17, ti=ffff5bb1

F(c49445b4, 16dcb90c, 104a2242)=14de2346

a+F(b,c,d)+xk+ti=362dadf5

FF: b+(... <<< 17)=207eb20f

FF (b, c, d, a, x[11], S14, 0x895cd7be)

FF: a=104a2242, b=207eb20f, c=c49445b4, d=16dcb90c, xk=00000000, s=22, ti=895cd7be

F(207eb20f, c49445b4, 16dcb90c)=16940904

a+F(b,c,d)+xk+ti=b03b0304

FF: b+(... <<< 22)=e1aac0cf

FF (a, b, c, d, x[12], S11, 0x6b901122)

FF: a=16dcb90c, b=e1aac0cf, c=207eb20f, d=c49445b4, xk=00000000, s=7, ti=6b901122

F(e1aac0cf, 207eb20f, c49445b4)=243e853f

a+F(b,c,d)+xk+ti=a6ab4f6d

FF: b+(... <<< 7)=375277a2

FF (d, a, b, c, x[13], S12, 0xfd987193)

FF: a=c49445b4, b=375277a2, c=e1aac0cf, d=207eb20f, xk=00000000, s=12, ti=fd987193

F(375277a2, e1aac0cf, 207eb20f)=212ec08f

a+F(b,c,d)+xk+ti=e35b77d6

FF: b+(... <<< 12)=eecfe5d7

FFF (c, d, a, b, x[14], S13, 0xa679438e)

FF: a=207eb20f, b=eecfe5d7, c=375277a2, d=e1aac0cf, xk=00000030, s=17, ti=a679438e

F(eecfe5d7, 375277a2, e1aac0cf)=2762658a

a+F(b,c,d)+xk+ti=ee5a5b57

FF: b+(... <<< 17)=a57fc28b

FF (b, c, d, a, x[15], S14, 0x49b40821)

FF: a=e1aac0cf, b=a57fc28b, c=eecfe5d7, d=375277a2, xk=00000000, s=22, ti=49b40821

F(a57fc28b, eecfe5d7, 375277a2)=b64ff5a3

a+F(b,c,d)+xk+ti=e1aebe93

FF: b+(... <<< 22)=4a782e3a

****

Round 2:

Lakukan operasi [abcd k s i]

a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s)

Lakukan 16 operasi berikut ini setiap roundnya:

[ABCD 1 5 17] [DABC 6 9 18] [CDAB 11 14 19] [BCDA 0 20 20]

[ABCD 5 5 21] [DABC 10 9 22] [CDAB 15 14 23] [BCDA 4 20 24]

[ABCD 9 5 25] [DABC 14 9 26] [CDAB 3 14 27] [BCDA 8 20 28]

[ABCD 13 5 29] [DABC 2 9 30] [CDAB 7 14 31] [BCDA 12 20 32]

Lakukan a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). (Catatan: <<< berarti rotate left)

****

Fungsi yang digunakan pada masukan G(b,c,d) = (b & d) | (c & ~d)

****

GG (a, b, c, d, x[1], S21, 0xf61e2562)

GG: a=375277a2, b=4a782e3a, c=a57fc28b, d=eecfe5d7, xk=00806d6f, s=5, ti=f61e2562

G(4a782e3a, a57fc28b, eecfe5d7)=4b78261a

a+G(b,c,d)+xk+ti=7969308d

GG: b+(... <<< 5)=779e3fe9

GG (d, a, b, c, x[6], S22, 0xc040b340)

GG: a=eecfe5d7, b=779e3fe9, c=4a782e3a, d=a57fc28b, xk=00000000, s=9, ti=c040b340

G(779e3fe9, 4a782e3a, a57fc28b)=6f1e2eb9

a+G(b,c,d)+xk+ti=1e2ec7d0

GG: b+(... <<< 9)=d52de025

GG (c, d, a, b, x[11], S23, 0x265e5a51)

GG: a=a57fc28b, b=d52de025, c=779e3fe9, d=4a782e3a, xk=00000000, s=14, ti=265e5a51

G(d52de025, 779e3fe9, 4a782e3a)=75ae31e1

a+G(b,c,d)+xk+ti=418c4ebd

GG: b+(... <<< 14)=e8dd3088

GG (b, c, d, a, x[0], S24, 0xe9b6c7aa)

GG: a=4a782e3a, b=e8dd3088, c=d52de025, d=779e3fe9, xk=6b696e75, s=20, ti=e9b6c7aa

G(e8dd3088, d52de025, 779e3fe9)=e0bdf08c

a+G(b,c,d)+xk+ti=805654e5

GG: b+(... <<< 20)=373535ed

GG (a, b, c, d, x[5], S21, 0xd62f105d)

GG: a=779e3fe9, b=373535ed, c=e8dd3088, d=d52de025, xk=00000000, s=5, ti=d62f105d

G(373535ed, e8dd3088, d52de025)=3df530ad

a+G(b,c,d)+xk+ti=8bc280f3

GG: b+(... <<< 5)=af85545e

GG (d, a, b, c, x[10], S22, 0x2441453)

GG: a=d52de025, b=af85545e, c=373535ed, d=e8dd3088, xk=00000000, s=9, ti=02441453

G(af85545e, 373535ed, e8dd3088)=bfa5156d

a+G(b,c,d)+xk+ti=971709e5

GG: b+(... <<< 9)=dd991f8c

GG (c, d, a, b, x[15], S23, 0xd8a1e681)

GG: a=e8dd3088, b=dd991f8c, c=af85545e, d=373535ed, xk=00000000, s=14, ti=d8a1e681

G(dd991f8c, af85545e, 373535ed)=9d91559e

a+G(b,c,d)+xk+ti=5f106ca7

GG: b+(... <<< 14)=f8c2f750

GG (b, c, d, a, x[4], S24, 0xe7d3fbc8)

GG: a=373535ed, b=f8c2f750, c=dd991f8c, d=af85545e, xk=00000000, s=20, ti=e7d3fbc8

G(f8c2f750, dd991f8c, af85545e)=f8985fd0

a+G(b,c,d)+xk+ti=17a19185

GG: b+(... <<< 20)=11147169

GG (a, b, c, d, x[9], S21, 0x21e1cde6)

GG: a=af85545e, b=11147169, c=f8c2f750, d=dd991f8c, xk=00000000, s=5, ti=21e1cde6

G(11147169, f8c2f750, dd991f8c)=3152f158

a+G(b,c,d)+xk+ti=02ba139c

GG: b+(... <<< 5)=6856e4e9

GG (d, a, b, c, x[14], S22, 0xc33707d6)

GG: a=dd991f8c, b=6856e4e9, c=11147169, d=f8c2f750, xk=00000030, s=9, ti=c33707d6

G(6856e4e9, 11147169, f8c2f750)=6956e469

a+G(b,c,d)+xk+ti=0a270bfb

GG: b+(... <<< 9)=b66edafd

GG (c, d, a, b, x[3], S23, 0xf4d50d87)

GG: a=f8c2f750, b=b66edafd, c=6856e4e9, d=11147169, xk=00000000, s=14, ti=f4d50d87

G(b66edafd, 6856e4e9, 11147169)=7846d4e9

a+G(b,c,d)+xk+ti=65ded9c0

GG: b+(... <<< 14)=6cdef474

GG (b, c, d, a, x[8], S24, 0x455a14ed)

GG: a=11147169, b=6cdef474, c=b66edafd, d=6856e4e9, xk=00000000, s=20, ti=455a14ed

G(6cdef474, b66edafd, 6856e4e9)=fe7efe74

a+G(b,c,d)+xk+ti=54ed84ca

GG: b+(... <<< 20)=b984434c

GG (a, b, c, d, x[13], S21, 0xa9e3e905)

GG: a=6856e4e9, b=b984434c, c=6cdef474, d=b66edafd, xk=00000000, s=5, ti=a9e3e905

G(b984434c, 6cdef474, b66edafd)=f894664c

a+G(b,c,d)+xk+ti=0acf343a

GG: b+(... <<< 5)=136aca8d

GG (d, a, b, c, x[2], S22, 0xfcefa3f8)

GG: a=b66edafd, b=136aca8d, c=b984434c, d=6cdef474, xk=00000000, s=9, ti=fcefa3f8

G(136aca8d, b984434c, 6cdef474)=914ac30c

a+G(b,c,d)+xk+ti=44a94201

GG: b+(... <<< 9)=65eecd16

G (c, d, a, b, x[7], S23, 0x676f02d9)

GG: a=6cdef474, b=65eecd16, c=136aca8d, d=b984434c, xk=00000000, s=14, ti=676f02d9

G(65eecd16, 136aca8d, b984434c)=23eec985

a+G(b,c,d)+xk+ti=f83cc0d2

GG: b+(... <<< 14)=96238b25

GG (b, c, d, a, x[12], S24, 0x8d2a4c8a)

GG: a=b984434c, b=96238b25, c=65eecd16, d=136aca8d, xk=00000000, s=20, ti=8d2a4c8a

G(96238b25, 65eecd16, 136aca8d)=76a68f17

a+G(b,c,d)+xk+ti=bd551eed

GG: b+(... <<< 20)=84ff6076

****

Round 3:

Lakukan operasi [abcd k s i]

a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s)

Lakukan 16 operasi berikut ini setiap roundnya:

[ABCD 5 4 33] [DABC 8 11 34] [CDAB 11 16 35] [BCDA 14 23 36]

[ABCD 1 4 37] [DABC 4 11 38] [CDAB 7 16 39] [BCDA 10 23 40]

[ABCD 13 4 41] [DABC 0 11 42] [CDAB 3 16 43] [BCDA 6 23 44]

[ABCD 9 4 45] [DABC 12 11 46] [CDAB 15 16 47] [BCDA 2 23 48]

Lakukan a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). (Catatan: <<< berarti rotate left)

****

Fungsi yang digunakan pada masukan H(b,c,d) = b ^ c ^ d

****

HH (a, b, c, d, x[5], S31, 0xfffa3942)

HH: a=136aca8d, b=84ff6076, c=96238b25, d=65eecd16, xk=00000000, s=4, ti=fffa3942

H(84ff6076, 96238b25, 65eecd16)=77322645

a+H(b,c,d)+xk+ti=8a972a14

HH: b+(... <<< 4)=2e7201be

HH (d, a, b, c, x[8], S32, 0x8771f681)

HH: a=65eecd16, b=2e7201be, c=84ff6076, d=96238b25, xk=00000000, s=11, ti=8771f681

H(2e7201be, 84ff6076, 96238b25)=3caeeaed

a+H(b,c,d)+xk+ti=2a0fae84

HH: b+(... <<< 11)=abe6230e

HH (c, d, a, b, x[11], S33, 0x6d9d6122)

HH: a=96238b25, b=abe6230e, c=2e7201be, d=84ff6076, xk=00000000, s=16, ti=6d9d6122

H(abe6230e, 2e7201be, 84ff6076)=016b42c6

a+H(b,c,d)+xk+ti=052c2f0d

HH: b+(... <<< 16)=daf3283a

HH (b, c, d, a, x[14], S34, 0xfde5380c)

HH: a=84ff6076, b=daf3283a, c=abe6230e, d=2e7201be, xk=00000030, s=23, ti=fde5380c

H(daf3283a, abe6230e, 2e7201be)=5f670a8a

a+H(b,c,d)+xk+ti=e24ba33c

HH: b+(... <<< 23)=79644e0b

HH (a, b, c, d, x[1], S31, 0xa4beea44)

HH: a=2e7201be, b=79644e0b, c=daf3283a, d=abe6230e, xk=00806d6f, s=4, ti=a4beea44

H(79644e0b, daf3283a, abe6230e)=0871453f

a+H(b,c,d)+xk+ti=dc229eb0

HH: b+(... <<< 4)=3b8e3918

HH (d, a, b, c, x[4], S32, 0x4bdecfa9

HH: a=abe6230e, b=3b8e3918, c=79644e0b, d=daf3283a, xk=00000000, s=11, ti=4bdecfa9

H(3b8e3918, 79644e0b, daf3283a)=98195f29

a+H(b,c,d)+xk+ti=8fde51e0

HH: b+(... <<< 11)=2e1d3d96

HH (c, d, a, b, x[7], S33, 0xf6bb4b60)

HH: a=daf3283a, b=2e1d3d96, c=3b8e3918, d=79644e0b, xk=00000000, s=16, ti=f6bb4b60

H(2e1d3d96, 3b8e3918, 79644e0b)=6cf74a85

a+H(b,c,d)+xk+ti=3ea5be1f

HH: b+(... <<< 16)=ec3c7c3b

HH (b, c, d, a, x[10], S34, 0xbebfbc70)

HH: a=79644e0b, b=ec3c7c3b, c=2e1d3d96, d=3b8e3918, xk=00000000, s=23, ti=bebfbc70

H(ec3c7c3b, 2e1d3d96, 3b8e3918)=f9af78b5

a+H(b,c,d)+xk+ti=31d38330

HH: b+(... <<< 23)=845565fc

HH (a, b, c, d, x[13], S31, 0x289b7ec6)

HH: a=3b8e3918, b=845565fc, c=ec3c7c3b, d=2e1d3d96, xk=00000000, s=4, ti=289b7ec6

H(845565fc, ec3c7c3b, 2e1d3d96)=46742451

a+H(b,c,d)+xk+ti=aa9ddc2f

HH: b+(... <<< 4)=2e3328f6

HH (d, a, b, c, x[0], S32, 0xeaa127fa)

HH: a=2e1d3d96, b=2e3328f6, c=845565fc, d=ec3c7c3b, xk=6b696e75, s=11, ti=eaa127fa

H(2e3328f6, 845565fc, ec3c7c3b)=465a3131

a+H(b,c,d)+xk+ti=ca820536

HH: b+(... <<< 11)=3e5cdf4a

HH (c, d, a, b, x[3], S33, 0xd4ef3085)

HH: a=ec3c7c3b, b=3e5cdf4a, c=2e3328f6, d=845565fc, xk=00000000, s=16, ti=d4ef3085

H(3e5cdf4a, 2e3328f6, 845565fc)=943a9240

a+H(b,c,d)+xk+ti=55663f00

HH: b+(... <<< 16)=7d5d34b0

HH (b, c, d, a, x[6], S34, 0x4881d05)

HH: a=845565fc, b=7d5d34b0, c=3e5cdf4a, d=2e3328f6, xk=00000000, s=23, ti=04881d05

H(7d5d34b0, 3e5cdf4a, 2e3328f6)=6d32c30c

a+H(b,c,d)+xk+ti=f610460d

HH: b+(... <<< 23)=84583cd3

HH (a, b, c, d, x[9], S31, 0xd9d4d039)

HH: a=2e3328f6, b=84583cd3, c=7d5d34b0, d=3e5cdf4a, xk=00000000, s=4, ti=d9d4d039

H(84583cd3, 7d5d34b0, 3e5cdf4a)=c759d729

a+H(b,c,d)+xk+ti=cf61d058

HH: b+(... <<< 4)=7a75425f

HH (d, a, b, c, x[12], S32, 0xe6db99e5)

HH: a=3e5cdf4a, b=7a75425f, c=84583cd3, d=7d5d34b0, xk=00000000, s=11, ti=e6db99e5

H(7a75425f, 84583cd3, 7d5d34b0)=83704a3c

a+H(b,c,d)+xk+ti=a8a8c36b

HH: b+(... <<< 11)=c0909fa4

HH (c, d, a, b, x[15], S33, 0x1fa27cf8)

HH: a=7d5d34b0, b=c0909fa4, c=7a75425f, d=84583cd3, xk=00000000, s=16, ti=1fa27cf8

H(c0909fa4, 7a75425f, 84583cd3)=3ebde128

a+H(b,c,d)+xk+ti=dbbd92d0

HH: b+(... <<< 16)=53617b61

HH (b, c, d, a, x[2], S34, 0xc4ac5665)

HH: a=84583cd3, b=53617b61, c=c0909fa4, d=7a75425f, xk=00000000, s=23, ti=c4ac5665

H(53617b61, c0909fa4, 7a75425f)=e984a69a

a+H(b,c,d)+xk+ti=328939d2

HH: b+(... <<< 23)=3c7abffd

****

Round 4:

Lakukan operasi [abcd k s i]

a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s)

Lakukan 16 operasi berikut ini setiap roundnya:

[ABCD 0 6 49] [DABC 7 10 50] [CDAB 14 15 51] [BCDA 5 21 52]

[ABCD 12 6 53] [DABC 3 10 54] [CDAB 10 15 55] [BCDA 1 21 56]

[ABCD 8 6 57] [DABC 15 10 58] [CDAB 6 15 59] [BCDA 13 21 60]

[ABCD 4 6 61] [DABC 11 10 62] [CDAB 2 15 63] [BCDA 9 21 64]

Lakukan a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). (Catatan: <<< berarti rotate left)

****
Fungsi yang digunakan pada masukan I(b,c,d) = c ^ (b | ~d)
****
II (a, b, c, d, x[0], S41, 0xf4292244)
II: a=7a75425f, b=3c7abffd, c=53617b61, d=c0909fa4, xk=6b696e75, s=6, ti=f4292244
I(3c7abffd, 53617b61, c0909fa4)=6c1e849e
a+I(b,c,d)+xk+ti=462657b6
II: b+(... <<< 6)=c610ad8e

II (d, a, b, c, x[7], S42, 0x432aff97)
II: a=c0909fa4, b=c610ad8e, c=3c7abffd, d=53617b61, xk=00000000, s=10, ti=432aff97
I(c610ad8e, 3c7abffd, 53617b61)=d2e41263
a+I(b,c,d)+xk+ti=d69fb19e
II: b+(... <<< 10)=44d728e8

II (c, d, a, b, x[14], S43, 0xab9423a7)
II: a=53617b61, b=44d728e8, c=c610ad8e, d=3c7abffd, xk=00000030, s=15, ti=ab9423a7
I(44d728e8, c610ad8e, 3c7abffd)=01c7c564
a+I(b,c,d)+xk+ti=00bd649c
II: b+(... <<< 15)=f7252946

II (b, c, d, a, x[5], S44, 0xfc93a039)
II: a=3c7abffd, b=f7252946, c=44d728e8, d=c610ad8e, xk=00000000, s=21, ti=fc93a039
I(f7252946, 44d728e8, c610ad8e)=bb38539f
a+I(b,c,d)+xk+ti=f446b3d5
II: b+(... <<< 21)=71e3b21c

III (a, b, c, d, x[12], S41, 0x655b59c3)

II: a=c610ad8e, b=71e3b21c, c=f7252946, d=44d728e8, xk=00000000, s=6, ti=655b59c3

I(71e3b21c, f7252946, 44d728e8)=0ccede59

a+I(b,c,d)+xk+ti=383ae5aa

II: b+(... <<< 6)=809d1caa

I (d, a, b, c, x[3], S42, 0x8f0ccc92)

II: a=44d728e8, b=809d1caa, c=71e3b21c, d=f7252946, xk=00000000, s=10, ti=8f0ccc92

I(809d1caa, 71e3b21c, f7252946)=f93c6ca7

a+I(b,c,d)+xk+ti=cd206221

II: b+(... <<< 10)=0225a3de

II (c, d, a, b, x[10], S43, 0xffeff47d)

II: a=f7252946, b=0225a3de, c=809d1caa, d=71e3b21c, xk=00000000, s=15, ti=ffeff47d

I(0225a3de, 809d1caa, 71e3b21c)=0ea0f355

a+I(b,c,d)+xk+ti=05b61118

II: b+(... <<< 15)=0ab1a6b9

II (b, c, d, a, x[1], S44, 0x85845dd1)

II: a=71e3b21c, b=0ab1a6b9, c=0225a3de, d=809d1caa, xk=00806d6f, s=21, ti=85845dd1

I(0ab1a6b9, 0225a3de, 809d1caa)=7dd64423

a+I(b,c,d)+xk+ti=75bec17f

II: b+(... <<< 21)=3aa05e91

II (a, b, c, d, x[8], S41, 0x6fa87e4f)

II: a=809d1caa, b=3aa05e91, c=0ab1a6b9, d=0225a3de, xk=00000000, s=6, ti=6fa87e4f

I(3aa05e91, 0ab1a6b9, 0225a3de)=f54bf808

a+I(b,c,d)+xk+ti=e5919301

II: b+(... <<< 6)=9f051f0a

II (d, a, b, c, x[15], S42, 0xfe2ce6e0)

II: a=0225a3de, b=9f051f0a, c=3aa05e91, d=0ab1a6b9, xk=00000000, s=10, ti=fe2ce6e0

I(9f051f0a, 3aa05e91, 0ab1a6b9)=c5ef01df

a+I(b,c,d)+xk+ti=c6418c9d

II: b+(... <<< 10)=a5379623

II (c, d, a, b, x[6], S43, 0xa3014314)

II: a=0ab1a6b9, b=a5379623, c=9f051f0a, d=3aa05e91, xk=00000000, s=15, ti=a3014314

I(a5379623, 9f051f0a, 3aa05e91)=7a7aa865

a+I(b,c,d)+xk+ti=282d9232

II: b+(... <<< 15)=6e50aa39

II (b, c, d, a, x[13], S44, 0x4e0811a1)

II: a=3aa05e91, b=6e50aa39, c=a5379623, d=9f051f0a, xk=00000000, s=21, ti=4e0811a1

I(6e50aa39, a5379623, 9f051f0a)=cbcd7cde

a+I(b,c,d)+xk+ti=5475ed10

II: b+(... <<< 21)=105b38f6

II (a, b, c, d, x[4], S41, 0xf7537e82)

II: a=9f051f0a, b=105b38f6, c=6e50aa39, d=a5379623, xk=00000000, s=6, ti=f7537e82

I(105b38f6, 6e50aa39, a5379623)=348bd3c7

a+I(b,c,d)+xk+ti=cae47153

II: b+(... <<< 6)=c9778de8

II (d, a, b, c, x[11], S42, 0xbd3af235)

II: a=a5379623, b=c9778de8, c=105b38f6, d=6e50aa39, xk=00000000, s=10, ti=bd3af235

I(c9778de8, 105b38f6, 6e50aa39)=c9a4e518

a+I(b,c,d)+xk+ti=2c176d70

II: b+(... <<< 10)=272d4e98

II (c, d, a, b, x[2], S43, 0x2ad7d2bb)

II: a=6e50aa39, b=272d4e98, c=c9778de8, d=105b38f6, xk=00000000, s=15, ti=2ad7d2bb

I(272d4e98, c9778de8, 105b38f6)=26da4271

a+I(b,c,d)+xk+ti=c002bf65

II: b+(... <<< 15)=86e02e99

II (b, c, d, a, x[9], S44, 0xeb86d391)

II: a=105b38f6, b=86e02e99, c=272d4e98, d=c9778de8, xk=00000000, s=21, ti=eb86d391

I(86e02e99, 272d4e98, c9778de8)=91c53007

a+I(b,c,d)+xk+ti=8da73c8e

II: b+(... <<< 21)=18b1e380

****

Lakukan operasi penambahan berikut. (That is increment each of the four registers by the value it had before this block was started.)

a = (a + AA) & 0x0ffffffff;

b = (b + BB) & 0x0ffffffff;

c = (c + CC) & 0x0ffffffff;

d = (d + DD) & 0x0ffffffff;

Round selesai,

a = 817672425, Konversi nilai 32-bit ke 8-char hex string --> 30bcb0e9, kembalikan ke nilai hex LSB (Least Significant Bit) --> e9b0bc30
b = 142577417, Konversi nilai 32-bit ke 8-char hex string --> 087f8f09, kembalikan ke nilai hex LSB (Least Significant Bit) --> 098f7f0800000000
c = 530254743, Konversi nilai 32-bit ke 8-char hex string --> 1f9b0b97, kembalikan ke nilai hex LSB (Least Significant Bit) --> 970b9b1f00000000
d = 929014542, Konversi nilai 32-bit ke 8-char hex string --> 375fa30e, kembalikan ke nilai hex LSB (Least Significant Bit) --> 0ea35f3700000000
Hasil hash terakhir: lsb_hex(a) + lsb_hex(b) + lsb_hex(c) + lsb_hex(d) = e9b0bc30098f7f08970b9b1f0ea35f37

2.7 ASCII

ASCII (American Standard Code for Information Interchange) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Ia selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan biner sebanyak 7 bit. Namun, ASCII disimpan sebagai sandi 8 bit dengan menambakan satu angka 0 sebagai bit significant paling tinggi. Bit tambahan ini sering digunakan untuk uji prioritas. Karakter control pada ASCII dibedakan menjadi 5 kelompok sesuai dengan penggunaan yaitu berturut-turut meliputi logical communication, Device control, Information separator, Code extention, dan physical communication. Code ASCII ini banyak dijumpai pada papan ketik (*Key*board) computer atau instrument-instrument digital.

Jumlah kode ASCII adalah 255 kode. Kode ASCII 0..127 merupakan kode ASCII untuk manipulasi teks; sedangkan kode ASCII 128..255 merupakan kode ASCII untuk manipulasi grafik. Kode ASCII sendiri dapat dikelompokkan lagi kedalam beberapa bagian:

- Kode yang tidak terlihat simbolnya seperti Kode 10 (Line Feed), 13(Carriage Return), 8(Tab), 32(Space)

- Kode yang terlihat simbolnya seperti abjad (A..Z), numerik (0..9), karakter khusus (~!@#$%^&*()_+?:”{})
- Kode yang tidak ada di *Key*board namun dapat ditampilkan. Kode ini umumnya untuk kode-kode grafik.

Dalam pengkodean kode ASCII memanfaatkan 8 bit. Pada saat ini kode ASCII telah tergantikan oleh kode UNICODE (*Universal Code*). UNICODE dalam pengkodeannya memanfaatkan 16 bit sehingga memungkinkan untuk menyimpan kode-kode lainnya seperti kode bahasa Jepang, Cina, Thailand dan sebagainya.

Pada papan *Key*board, aktifkan numlock, tekan tombol ALT secara bersamaan dengan kode karakter maka akan dihasilkan karakter tertentu. Misalnya: ALT + 44 maka akan muncul karakter koma (,). Mengetahui kode-kode ASCII sangat bermanfaat misalnya untuk membuat karakter-karakter tertentu yang tidak ada di *Key*board.

Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	sp
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29	)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>
011 1111	077	63	3F	?

Binary	Oct	Dec	Hex	Glyph
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O
101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D	]
101 1110	136	94	5E	^
101 1111	137	95	5F	_

Binary	Oct	Dec	Hex	Glyph
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o
111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	|
111 1101	175	125	7D	}
111 1110	176	126	7E	~

Gambar 2.12 Kode ASCII

2.8 Base64

Base64 adalah istilah umum untuk sejumlah skema pengkodean serupa yang mengkodekan data biner dan menerjemahkannya ke dalam representasi basis 64. Istilah Base64 berasal dari konten pengkodean MIME tertentu.

Skema encoding base64 biasanya digunakan ketika ada kebutuhan untuk menyandikan data biner yang perlu disimpan dan ditransfer melalui media yang dirancang untuk menangani data tekstual. Hal ini untuk memastikan bahwa data tetap utuh tanpa perubahan selama pengiriman. Base64 digunakan umum dalam beberapa aplikasi termasuk email melalui MIME, dan penyimpanan data yang kompleks dalam XML.

Transformasi base64 banyak digunakan di dunia Internet sebagai media data format untuk mengirimkan data. Dikarenakan hasil dari transformasi base64 berupa plaintext, maka nilai ini akan jauh lebih mudah dikirim, dibandingkan format data berupa binary.

Value	Char	Value	Char	Value	Char	Value	Char
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Gambar 2.13 Encoding Base64

2.8.1 Penggunaan Base64

Base64 digunakan untuk melakukan metode *encoding* (penyandian) terhadap data binary sehingga menjadi "*printable*" atau pendek katanya data binary disandikan menjadi format 7-bit character semacam bagian isi email.

Ada satu baris data binary tapi ingin digabung dengan data teks dan atau dikirim dalam format teks, maka data binary itu tidak akan bisa menghasil informasi yang jelas sebelum di *encode*/sandi dulu ke format teks.

Di dunia ini, manusia sebagian besar terbiasa dengan angka base 10 atau lebih dikenal dengan decimal, dimana suatu nilai disusun dari 10 karakter yang terdiri dari 0 – 9.

Begitu juga base 2 atau binary, dimana nilai angka ini disusun hanya oleh 2 karakter yaitu 0 dan 1.

Lalu kenapa ada base 2, padahal manusia hanya menggunakan base 10. Karena sang kompor (mesin/komputer) hanya mengenal 0 dan 1. Atau istilahnya off dan on. Di chip semua proses hanya terdiri dari apakah on atau off (tidak ada yang lain) yang direfresentasikan ke dalam nilai 0 dan 1. Maka munculah base 2. Jadi kemunculannya karena kebutuhan tersebut.

Jadi Base64 ini muncul akan kebutuhan suatu encoding yang bisa mengubah serangkaian data binary menjadi format yang bisa di cetak normal, atau “printable”.

Intinya dari Base64 adalah base 64 yang disusun oleh 64 karakter, dimana karakternya (berdasarkan: RFC 1421) terdiri dari (A-Z, a-z, 0-9, +, /) atau “ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz012345 6789+/”. Tapi ada satu karater tambahan yaitu “=”, yang fungsinya untuk menggenapkan atau istilahnya sebagai pengisi pad.

Teknik *encoding* Base64 sebenarnya sederhana, jika ada satu senarai (*string*) bytes yang akan disandikan ke Base64 maka caranya adalah:

1. Pecah string bytes tersebut ke per-3 bytes.
2. Gabungkan 3 bytes menjadi 24 bit. Ingat 1 bytes = 8 bit, sehingga 3 x 8 = 24 bit.
3. Lalu 24 bit yang disimpan dibufferkan (disatukan) dipecah – pecah menjadi 6 bit - 6 bit, maka akan menghasilkan 4 pecahan.
4. Masing-masing pecahan diubah kedalam nilai decimal, dimana maksimal nilai 6 bit adalah 63.

5. Terakhir, jadikan nilai-nilai decimal tersebut menjadi indeks untuk memilih karakter penyusun dari base64 dan maksimal adalah 63 atau indeks ke 64. Dan seterusnya sampai akhir string bytes yang mau kita konversikan.

Jika ternyata dalam proses encoding terdapat sisa pembagi, maka tambahkan sebagai penggenap sisa tersebut karakter '='. Maka terkadang pada base64 akan muncul satu atau dua karakter '='.

Contoh Base64:

Data teks:

Harga BBM Naik

Diencoding akan menghasilkan:

SGFyZ2EgQkJNIE5haWs=

Cara kerjanya:

1. Pecah string bytes tersebut ke per-3 bytes.

   Pertama dapatkan dulu nilai bytes dari data teks tersebut. (Ingat Base64 Encoding adalah operasi manipulasi bit)

   "Harga BBM Naik" nilai ASCI nya adalah:

   72,97,114,103,97,32,66,66,77,32,78,97,105,107

   Lalu pecah per 3 bytes.

   72,97,114 – 103,97,32 – 66,66,77 – 32,78,64 – 105,107

   untuk memudahkan membaca bit datanya, kita konversi nilai decimal ini ke nilai hexa. Gunakan kalkulator Dec2Hex.

   72,97,114 = 0x48,0x61,0x72

   Lalu kita pisahkan angkanya satu persatu dan ubah masing – masing angka menjadi nilai biner lalu digabung menjadi 24 bit.

   0x48,0x61,0x72 = 4 &8, 6&1, 7&2 = 0100-1000,0110-0001,0111-0010

   Hasilnya:

   Har = 72,97,114 = 01001000,01100001,01110010

   Nah sekarang kita sudah mendapatkan string bytes (data binari) dari data teks "Har"

2. Gabungkan 3 bytes menjadi 24 bit.

01001000,01100001,01110010 = 010010000110000101110010

3. Pecah menjadi 6 bit – 6 bit dan akan menghasilkan 4 pecahan.
   010010000110000101110010 = 010010,000110,000101,110010
4. Masing-masing pecahan (adalah nilai octal) diubah kedalam nilai decimal.
   010010,000110,000101,110010 = 18,6,5,50
   kita dapat nilai (decimal) untuk “Har” =18,6,5,50
5. Jadikan nilai – nilai tersebut menjadi indeks untuk memilih karakter penyusun dari base64
   ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
   dimana [A = 0], maka atau Base64Char[0] = “A”
   18 = S, 6 = G, 5 = F, 50 = y
   sehingga,
   “Har” = “SGFy”

Huruf	H								a								r							
ASCII	72								97								114							
Bit	0	1	0	0	1	0	0	0	0	1	1	0	0	0	0	1	0	1	1	1	0	0	1	0
Index	18								6								5							
Base64 *Encode*	S								G								F							
	y																							

Jadi bisa dipahami, bahwa setiap 3 bytes data akan menghasilkan 4 bytes data artinya 4/3 (33%) lebih banyak.

“ga” = “Z2Eg” (spasi termasuk 1 karakter)

Dan lakukan seterusnya seperti diatas untuk 3 bytes berikutnya.

Nah yang “ik” ini tidak genap 3 bytes ini yang dinamakan terjadi “pad”.

105,107 = 01101001,01101011 = 011010,010110,1011

011010,010110,1011 + 00 = 011010,010110,101100 = 26,22,44

“ik” = “aWs”

Huruf	i	k	

ASCII	105								107															
Bit	0	1	1	0	1	0	0	1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	0
Index	26								22				44				0							
Base64 *Encode*	a								W				s				=							

lalu tambahkan "=" sebagai penggenap sehingga lengkapnya -> SGFyZ2EgQkJNIE5haWs=

Dan inilah cara kerja yang dinamakan Base64*Encode*.

Ada beberapa modifikasi dari Base64*Encode* ini khususnya pada karakter "+/"

contohnya urlbase64*encode* dimana "+" diubah menjadi "-" dan "/" menjadi "_" serta dihilangkannya "=", tujuan penggantian ini disebabkan pada URL karakter "+" akan diubah menjadi "'%2B" dan "/" menjadi "%2F" sehingga bisa merubah makna.

2.8.2 Implementasi Pada .NET

Pada .NET Framewok encoding base64 telah disediakan. Sehingga kita tinggal memanggil fungsinya untuk melakukan *encode* maupun *decode*.

2.9 OpenSSL

OpenSSL merupakan open source implementasi dari protokol SSL (Secure Socket Layer) dan TLS (Transport Layer Security). SSL dan TLS, merupakan kelanjutan dari protokol kriptografi yang menyediakan komunikasi yang aman di internet. OpenSSL termasuk salah satu program open source yang direkomendasikan pada standar keamanan FIPS 140-2 yang dikeluarkan oleh NIST pada CMVP (Cryptographic Module Validation Program). Jadi sudah menjadi tool standard dan bebas pakai. OpenSSL dapat diunduh di http://www.openssl.org/.

Tujuan diciptakannya OpenSSL adalah untuk membuat aman aliran data dari komputer pengguna ke suatu server yang dituju. OpenSSL mulai diterapkan pada akhir tahun 1998, yang dikembangkan secara Open Source dan merupakan sebuah proyek kolaborasi dari para programmer di seluruh dunia. Oleh karena itu OpenSSL menjadi toolkit keamanan situs yang banyak digunakan, termasuk situs-situs besar yang ada saat ini seperti Facebook, Google, Amazon, maupun Yahoo.

Library OpenSSL digunakan untuk implementasi semua versi protokol SSL, terutama untuk penggunaan algoritma kriptografi simetris dan kriptografi kunci publik, algoritma hash dan *message digest*, juga pembangkit bilangan pseudorandom dan mendukung tandatangan digital untuk manipulasi format sertifikat umum/ bersama serta me-manage material kunci/ *Key*. OpenSSL mendukung semua penggunaan kriptografi dan akselerasi hardware. Kriptografi pada openSSL merupakan link sederhana pada socket library ketika membangun program. OpenSSL tidak membutuhkan developer memahami algoritma dan protocol kriptografi, tetapi hanya membutuhkan pemahaman bagaimana mengaplikasikan algoritma kriptografi.

Aplikasi OpenSSL ini merupakan command line tool yang menggunakan berbagai fungsi kriptografi OpenSSL’s crypto library dari shell. ini dapat digunakan untuk :

1. Membuat ceriticate SSL hotspot mikrotik
2. Penciptaan sertifikat X.509, CSRs dan CRLs
3. Perhitungan Pesan Digests
4. Enkripsi dan Dekripsi dengan Ciphers
5. Pengujian SSL / TLS Client dan Server
6. Penanganan S/MIME signed or encrypted mail
7. Penciptaan RSA, DH dan DSA parameter kunci

2.10 Keamanan Jaringan

Perkembangan teknologi jaringan komputer menyebabkan terkaitnya satu komputer dengan komputer lainya. Hal ini membuka banyak peluang dalam pengembangan aplikasi komputer tetapi juga mebuat peluang adanya ancaman terhadap pengubahan dan pencurian data. Sebuah aplikasi yang melintasi jaringan

publik seperti internet diasumsikan dapat diakses oleh siapapun termasuk orangorang atau pihak-pihak yang memang berniat untuk mencuri atau mengubah data.

Oleh karena itu, untuk melindungi data terhadap akses, pengubahan dan penghalangan yang tidak dilakukan oleh pihak yang berwenang, peranti keamanan data yang melintas di jaringan komputer harus disediakan

Layanan-layanan keamanan jaringan didefinisikan berdasarkan kebutuhan yang harus disediakan untuk memenuhi permintaan terhadap keamanan jaringan.

Berikut ini beberapa aplikasi yang memerlukan enkripsi untuk pengamanan data atau komunikasi diantaranya adalah:

- Jasa telekomunikasi
  - Enkripsi untuk mengamankan informasi konfidensial baik berupa suara, data, maupun gambar yang akan dikirimkan ke lawan bicaranya.
  - Enkripsi pada transfer data untuk keperluan manajemen jaringan dan transfer on-line data billing.
  - Enkripsi untuk menjaga copyright dari informasi yang diberikan.
- Militer dan pemerintahan
  - Enkripsi diantaranya digunakan dalam pengiriman pesan.
  - Menyimpan data-data rahasia militer dan kenegaraan dalam media penyimpanannya selalu dalam keaadan terenkripsi.
- Data Perbankan
  - Informasi transfer uang antar bank harus selalu dalam keadaan terenkripsi.
- Data konfidensial perusahaan
  - Rencana strategis, formula-formula produk, database pelanggan/karyawan dan database operasional.
  - Pusat penyimpanan data perusahaan dapat diakses secara on-line.
  - Teknik enkripsi juga harus diterapkan untuk data konfidensial untuk melindungi data dari pembacaan maupun perubahan secara tidak sah.
- Pengamanan *e-mail*

- Mengamankan pada saat ditransmisikan maupun dalam media penyimpanan.
- Aplikasi enkripsi telah dibuat khusus untuk mengamankan e-mail, diantaranya PEM (*Privacy Enhanced Mail*) dan PGP (*Pretty Good Privacy*), keduanya berbasis DES dan RSA.

- Kartu Plastik
  - Enkripsi pada SIM Card, kartu telepon umum, kartu langganan TV kabel, kartu kontrol akses ruangan dan komputer, kartu kredit, kartu ATM, kartu pemeriksaan medis, dll.
  - Enkripsi teknologi penyimpanan data secara magnetic, optik, maupun chip.

2.11 Serangan Terhadap Jaringan Komputer

Serangan keamanan jaringan yang dioperasionalkan pada jaringan publik rentan terhadap serangan oleh pihak yang tidak berwenang. Serangan terhadap sistem keamanan jaringan bertujuan untuk mengalahkan tujuan layanan keamanan jaringan, misalnya penyerang ingin mengungkap isi teks asli sehingga ia dapat mengungkap teks sandi lainya. Secara umum serangan pada sistem keamanan jaringan dapat dikategorikan menjadi 2 jenis, yaitu serangan pasif dan serangan aktif.

2.11.1 Serangan Aktif

Serangan aktif dapat mengakibatkan perubahan data yang sedang dikirimkan melaui jaringan dan mengganggu sistem. Serangan pasif memungkinkan penyerang untuk mengubah data pada lalu lintas data dijaringan.
Jenis – jenis serangan aktif adalah sebagai berikut:

1. Masquerade, merupakan serangan aktif yang dilakukan oleh penyerang dengan cara mengambil alih (menirukan) perilaku pengirim atau penerima. Contohnya pada saat Alice ingi membuat kunci bersama dengan Bob, disaat yang bersamaan secara tidak sadar Rani mengambil alih peran Bob sehingga Alice mengirim pesan ke Rani bukan kepada Bob.

2. Modification, merupakan serangan dengan cara mengambil alih jalur komunikasi untuk mengubah atau menghapus pesan yang sedang dikirimkan untuk keuntungan penyerang. Misalnya pesan “kirim Rp 100.000, 00 ke nomer rekening Alice” dirubah oleh Rani menjadi “kirim Rp 100.000, 00 ke nomer rekening Rani”.
3. Replay, merupakan serangan yang terdiri dari pencatatan secara pasif data dan transmisi ulang untuk menimbullkan efek yang diinginkan oleh penyerang. Misalnya Rani pernah meminta Bob untuk mengirim Rp 100.000, 00 ke Rani, lalu Bob mengirim pesan “kirim Rp 100.000, 00 ke Rani” ke Bank, pada saat pesan dikirimkan Rani mencatat pesan “kirim Rp 100.000, 00 ke Rani” lalu mengirim ulang pesan tersebut ke Bank sehingga Rani mendapat uang dengan jumlah dua kali lebih besar.
4. *Denial of Service*, merupakan jenis serangan yang bertujuan menyebabkan sistem menjadi tidak dapat bekerja sama sekali sehingga tidak bisa memeberikan pelayanan yang seharusnya kepada pengguna. Contohnya adalah penyerang membuat permintaan kosong (dummy) sehingga server menjadi overload.

Biasanya sebuah sistem keamanan jaringan dikatakan aman apabila tahan terhadap serangan aktif. Untuk mengembangkan sistem keamanan yang baik, seorang perancang keamanan jaringan harus menganalisis kemungkinan serangan – serangan terhadap layanan keamanan jaringan.

2.11.2 Serangan Pasif

Pada seragnan pasif, penyerang hanya mengumpulkan data yang melintas pada jaringan publik (jaringan yang bisa diakses oleh semua orang). Serangan pasif tidak melakukan modifikasi data yang melintas atau merusak sistem, penyerang hanya bisa membaca (read only). Berdasarkan data yang dikumpulkan tadi penyerang melakukan analisis untuk menggagalkan tujuan layanan keamanan jaringan. Karena tidak melakukan modifikasi dan mengganggu sistem, serangan pasif sulit di deteksi, tapi dapat dicegah dengan cara menggunakan sandi pada saat mengirim pesan misalnya. Berikut beberapa jenis serangan pasif:

1. *Snooping*, merupakan kegiatan yang bermaksud untuk mendapatkan data yang sedang dikirimkan melalui jaringan oleh pihak yang tidak berwenang. Contoh kegiatan *snooping* yaitu penyadapan email, snooping ini bisa dicegah dengan cara menyandikan pesan menggunakan mekanisme penyandian (*enchiperment*).

   Taffic Analysis, merupakan kegiatan serangan dengan melakukan pemantauan terhadap lalu lintas jaringan untuk kemudian datanya dikumpulkan dan dianalisis untuk mendapatkan informasi dari data – data tersebut.