BAB II

LANDASAN TEORI

2.1 Sejarah Kriptografi

Kriptografi memiliki sejarah yang panjang dan sangat menarik. Catatan nonteknikal terlengkap mengenai kriptografi secara non-teknikal adalah sebuah buku yang disusun oleh Kahn yang berjudul The Codebreakers. Buku ini menerjemahkan kriptografi dari inisial dan penggunaannya yang terbatas oleh bangsa Mesir sekitar 4000 tahun yang lalu, sampai ke abad 20 dimana kriptografi berperan penting dalam penentuan hasil dari perang dunia kesatu dan kedua. Buku ini diselesaikan pada tahun 1963, buku Kahn menjelaskan aspek sejarah yang nyata (terkini pada waktu itu) mengenai perkembangan dari kriptografi. Suatu bentuk yang cukup dominan terhadap seni tetapi juga berasosiasi dengan pendidikan militer, jasa diplomatik, dan pemerintahan pada umumnya. Kriptografi telah digunakan sebagai alat untuk melindungi rahasia nasional dan strategi.

Perkembangan komputer dan sistem komunikasi pada tahun 1960 sejalan dengan permintaan terhadap sektor rahasia/pribadi yang bertujuan untuk melindungi informasi dalam bentuk digital dan untuk menyediakan jasa keamanan. Dimulai dengan pekerjaan dari Feistel dari IBM pada awal 1970 dan memuncak pada tahun 1977 dengan adanya pengakuan sebagai U.S. Federal Information Processing Standard untuk mengenkripsi informasi yang tidak terklasifikasi, DES, Data Encryption Standard, merupakan nama lain yang lebih dikenal dari mekanisme kriptografi ini. Kata standar memiliki arti bahwa dapat

digunakan untuk mengamankan perdagangan elektronik untuk banyak lembaga keuangan diseluruh dunia.

Perkembangan yang paling menarik pada sejarah kriptografi ada pada tahun 1976 ketika Diffie dan Hellman menerbitkan New Directions in Cryptography. Tulisan ini memperkenalkan konsep revolusi dari kriptografi kunci publik dan juga menyediakan metode pertukaran kunci yang baru dan jenius, keamanan yang berdasarkan prinsip dari logaritma diskrit. Walaupun penulis tidak memiliki realisasi secara praktikal dari skema enkripsi kunci publik pada saat itu, tetapi idenya jelas dan membuat ketertarikan yang cukup besar dan berpengaruh pada aktivitas komunitas kriptografi. Pada tahun 1978 Rivest, Shamir, dan Adleman menemukan enkripsi kunci publik dan skema tandatangan untuk pertama kalinya secara praktikal, yang sekarang diserahkan menjadi RSA. Skema RSA berdasarkan masalah matematika sulit lainnya, seperti prinsip dari faktorial bilangan integer dalam jumlah besar. Aplikasi dari masalah matematika ini kepada kriptografi adalah bagian dari usaha untuk menemukan metode yang lebih efisien untuk menemukan faktor. Tahun 1980 menunjukkan kemajuan besar tetapi tidak ada yang membuat sistem RSA menjadi tidak aman. Kelas lain yang kuat dan ada secara praktikal dalam skema kunci-publik ditemukan oleh ElGamal pada tahun 1985, yang juga berdasarkan pada logaritma diskrit.

Salah satu kontribusi yang penting disediakan oleh kriptografi kunci publik adalah tanda tangan digital. Pada tahun 1991 standar internasional pertama untuk tandatangan digital telah disetujui (ISO/IEC 9796). Tandatangan digital ini dibuat berdasarkan pada skema RSA kunci publik. Pada 1994 pemerintah

Amerika memakai standar tandatangan digital, yang mekanismenya berdasarkan pada skema kunci publik ElGamal.

Pencarian pada skema kunci publik yang baru, meningkatkan mekanisme kriptografi yang telah ada, dan menjadi bukti dari sistem keamanan yang cepat berkembang. Beragam standar dan infrastruktur yang menyangkut dengan kriptografi dikumpulkan pada suatu tempat. Produk keamanan sedang dikembangkan untuk memenuhi keperluan keamanan dari informasi rahasia pada suatu lembaga.

2.2 Pengertian Kriptografi

Kriptografi (*cryptography*) merupakan ilmu dan seni untuk menjaga pesan agar aman. (*Cryptography is the art and science of keeping messages secure.*) "*Crypto*" berarti "*secret*" (rahasia) dan "*graphy*" berarti "*writing*" (tulisan). 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.

Proses yang dilakukan untuk mengamankan sebuah pesan (yang disebut *plaintext*) menjadi pesan yang tersembunyi (disebut *ciphertext*) adalah enkripsi (*encryption*). *Ciphertext* adalah pesan yang sudah tidak dapat dibaca dengan mudah. 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*. *Cryptology* merupakan gabungan dari *cryptography* dan *cryptanalysis*.

2.3 Permasalahan Kriptografi Kunci Privat

Salah satu cara untuk menambah tingkat keamanan sebuah algoritma enkripsi dan dekripsi adalah dengan menggunakan sebuah kunci (*key*) yang biasanya disebut *K*. Kunci *K* ini dapat memiliki rentang (*range*) yang cukup lebar. Rentang dari kemungkinan angka (harga) dari kunci *K* ini disebut *keyspace*. Kunci K ini digunakan dalam proses enkripsi dan dekripsi sehingga persamaan matematisnya menjadi:

$$EK\ (M) = C$$

$$DK\ (M) = M$$

Keamanan sistem yang digunakan kemudian tidak bergantung kepada pengetahuan algoritma yang digunakan, melainkan bergantung kepada kunci yang digunakan. Artinya, algoritma dapat diketahui oleh umum atau dipublikasikan. Usaha untuk memecahkan keamanan sistem menjadi usaha untuk memecahkan atau mencari kunci yang digunakan.

Usaha mencari kunci sangat bergantung kepada keyspace dari kunci *K*. Apabila keyspace ini cukup kecil, maka cara *brute force* atau mencoba semua kunci dapat dilakukan. Akan tetapi apabila keyspace dari kunci yang digunakan cukup besar, maka usaha untuk mencoba semua kombinasi kunci menjadi tidak realistis. Keyspace dari *DES*, misalnya, memiliki 56-bit. Untuk mencoba semua kombinasi yang ada diperlukan kombinasi. (Cerita tentang kelemahan DES akan diutarakan di bagian lain.)

Pada mulanya semua proses kriptografi menggunakan satu kunci yang sama untuk mengunci data dan membuka data. Jadi, kerahasiaan kunci ini sangat esensial. Jika kunci ini jatuh ke tangan pihak yang tidak berwenang, maka terbukalah rahasia. Penggunaan satu kunci ini membuat sistem pengamanan data tadi disebut *private-key cryptosystem*, atau sistem kriptografi berbasis kunci privat. Penekanan ada pada kata "privat", dimana kunci ini harus dirahasiakan, privat.

Selain itu sistem ini juga disebut symmetric cryptosystem, atau system kriptografi simetris karena kunci yang dipakai untuk proses enkripsi sama dengan kunci yang digunakan pada proses ekripsi. Simetris. Dalam aplikasinya, sistem kripto kunci privat ini memiliki beberapa masalah. Masalah pertama adalah **kesulitan dalam distribusi kunci. (Key Distribution Problem.)** Jika Anwar (A) ingin berkomunikasi melalui email dengan Broto (B) dengan mengenkripsi datanya (karena tidak yakin jalur data mereka aman dari penyadapan), apa kunci yang mereka gunakan? Bagaimana cara mereka untuk membuat kesepakatan kunci yang akan digunakan? Jika kunci tersebut dikirimkan melalui jalur

komunikasi yang dianggap tidak aman tersebut, maka ada kemungkinan disadap orang.

Ada beberapa solusi terhadap masalah ini, misalnya Anwar dan Broto bertemu dahulu secara fisik kemudian mendiskusikan kunci rahasia mereka. Atau mereka menggunakan media lain (misalnya telepon, fax, handphone, SMS) untuk mengirimkan kunci rahasia mereka. Pendekatan ini disebut

dengan *out of band communication*. Tapi masalahnya tidak semua orang memiliki cara komunikasi lain, atau kemungkinannya cara lain menjadi mahal dan tidak nyaman. Bayangkan jika anda harus mengkomunikasikan password ini, "s%Xy7&*!ih198907@1", kepada lawan bicara anda melalui telepon. Sangat tidak nyaman dan sulit.

Kesulitan akan semakin bertambah jika kedua belah pihak belum pernah kenal satu sama lainnya. Misalnya kita membuat sebuah situs web untuk melakukan transaksi online. Kita belum kenal dengan (calon) pembeli yang mengunjungi situs web kita. Bagaimana memilih kunci rahasia antara kita dengan sang pembeli tersebut? (Ini permasalahan *key exchange*.) Permasalahan kedua adalah peningkatan jumlah kunci yang eksponensial terhadap jumlah pengguna. Pada contoh sebelumnya, jika Anwar ingin berkomunikasi dengan Broto, mereka harus punya satu kunci rahasia. Bagaimana jika Anwar ingin berkomunikasi dengan Dodi? Tentunya mereka tidak bisa menggunakan kunci yang sama dengan kunci Anwar-Broto. Anwar dan Dodi harus sepakat untuk menggunakan satu kunci yang lain, kunci Anwar-Dodi. Bagaimana jika Broto ingin berkomunikasi dengan Dodi? Maka akan ada kunci Broto-Dodi yang berbeda dengan kunci yang

sudah-sudah. Jika skenario ini kita teruskan dengan menambahkan pengguna lain, maka dapat kita lihat peningkatan jumlah kunci secara eksponensial.

Jika *n* merupakan jumlah pengguna yang akan saling berkomunikasi, maka jumlah kunci yang ada adalah:

jumlah kunci = (n) (n-1) / 2

Mari kita coba tabel jumlah kunci yang digunakan dengan jumlah pengguna.

Jumlah Kunci dan Pengguna

Tabel 2.1 Jumlah Kunci dan Pengguna

Jumlah Pengguna (n)	Jumlah Kunci
10	45
100	4950
1000	499500
10000	49.995.00
100000	5 milyar

Dapat kita lihat pada tabel di atas bahwa peningkatan jumlah kunci meledak secara eksponensial. (Dari rumus pun dapat dilihat bahwa jumlah kunci merupakan hasil kuadrat dari *n*.) Dengan hanya seratus ribu pengguna saja, sudah ada lima (5) milyar kunci. Padahal jumlah pengguna Internet sangat jauh lebih besar dari seratus ribu orang. Jika satu kunci membutuhkan penyimpanan sebesar 1 kByte, maka dibutuhkan 5 TerraBytes untuk menyimpan kunci 100.000 orang. Jika kita berbicara tentang transaksi di Internet, e-commerce, maka bisa kita lihat dua kesulitan di atas sudah membuat kriptografi kunci privat menjadi tidak cocok. Jumlah pengguna e-commerce lebih dari 100.000 orang. Sementara itu key distribution juga sulit. Harus dicari sistem lain yang lebih baik.

2.4 Permasalahan Kriptografi Kunci Public

Kesulitan dalam penggunaan kriptografi kunci privat membuat banyak orang berpikir keras untuk mencari solusinya. Salah satu ide yang muncul adalah bagaimana jika kita membuat sebuah sistem penyadian dengan dua kunci, dimana satu kunci digunakan untuk proses enkripsi dan satu kunci lain digunakan untuk proses dekripsi.

Ide ini muncul dari Ralph Merkle ketika dia menjadi mahasiswa di sebuah perguruan tinggi. Ide tersebut dikemukakannya kepada dosennya. Namun ditolak mentah-mentah. Ide dua kunci tersebut tidak akan dapat dilaksanakan. Itu ide gila. Ralph Merkle kemudian menulis sebuah artikel yang dikirimkan ke journal, tapi artikel ini juga ditolak. Bagaimana ide itu bermula? Saya ambil sebuah cerita. (Cerita ini bukan contoh yang digunakan oleh Ralph Merkle.) Ceritanya adalah sebagai berikut.

Anwar dan Broto ingin bertukar pesan atau benda melalui pos. Mereka tidak ingin orang lain, termasuk pak Pos, mengetahui isi kirimannya. Anwar punya ide yang brilian. Anwar bertemu dengan Broto dan memberikan sebuah gembok yang terbuka, belum terkunci. Sementara itu Anwar tetap memegang kunci gemboknya tersebut. Kita sebut gembok ini adalah gembok-A. Ketika Broto ingin mengirimkan pesan (atau benda) kepada Anwar, dia letakkan pesan tersebut di dalam sebuah peti. Beserta pesan tersebut Broto juga memasukkan gembok dia (kita sebut gembok-B) yang terbuka juga. Kemudian pesan dan gembok-B ini dimasukkan di peti dan

peti dikunci dengan gembok-A. Dalam kondisi seperti ini, tidak ada seorangpun yang dapat membuka peti itu kecuali Anwar, karena hanya Anwar yang memiliki kunci gembok-A. Broto pun setelah mengunci peti tersebut tidak bisa membukanya kembali.

Di sisi penerima, Anwar, dia menerima peti yang sudah terkunci dengan gembok-A. Tentu saja dia dengan mudah dapat membuka peti tersebut karena dia memiliki kunci gembok-A. Setelah dia buka, maka dia dapat melihat pesan yang dikirimkan oleh Broto beserta gembok-B milik Broto yang terbuka.

Jika kemudian Anwar ingin mengirimkan jawaban atau pesan kepada Broto, maka dia dapat memasukkan jawabannya ke dalam peti dan tidak lupa mengikutsertakan gembok-A lagi yang terbuka ke dalamnya. Peti tersebut kemudian dikunci dengan gembok-B lagi, yang hanya dapat dibuka oleh Broto. Proses ini dapat berlangsung terus menerus. Contoh cerita di atas tentu saja masih belum sempurna. Inti yang ingin disampaikan adalah bahwa ada kemungkinan untuk melakukan pengamanan dengan tidak menggunakan enkripsi kunci privat. Penerima dan pengirim pesan dapat menggunakan kunci yang berbeda untuk pengamanan datanya.

Di tempat lain, ada seorang yang bernama Whitfield Diffie, juga memiliki ide yang mirip. Setelah mengembara kesana kemari, akhirnya Diffie bertemu dengan Martin Hellman yang menjadi profesor di Stanford University. Keduanya kemudian merumuskan ide *public-key cryptography* dalam sebuah makalah yang berjudul "*New Directions in Cryptography*" di tahun 1976. Lucunya Diffie dan

Hellman tidak kenal Ralph Merkle dan tidak tahu bahwa ada ide yang mirip. Pasalnya, artikel Merkle ditolak oleh berbagai publikasi.

Ide utama pada *public-key cryptography* adalah kunci yang digunakan untuk melakukan proses enkripsi berbeda dengan proses dekripsi. Hal ini dimungkinkan dengan penggunaan rumus matematik yang indah. Namun pencarian rumus matematik yang mana merupakan persoalan tersendiri.

Setelah keluarnya makalah tersebut, banyak orang yang mulai menaruh perhatian pada kriptografi kunci publik. Ternyata ide Ralph Merkle benar juga. Bahkan akhirnya Ralph Merkle mendapat penghargaan *Kanellakis Award* dari ACM dan *Kobayashi Award* dari IEEE. Salah satu kelompok yang tertarik kepada ide kriptografi kunci public tersebut adalah kelompok di MIT yang terdiri atas Ron Rivest, Adi Shamir, dan Len Adleman. Mereka mencoba mencari rumus matematik yang dapat mengimplementasikan ide kunci publik tersebut. Akhirnya setelah sekian lama berusaha, mereka menemukan algoritmanya yang kemudian dikenal dengan nama RSA (yang merupakan singkatan dari nama keluarga ketiga orang tersebut). Algoritma ini kemudian mereka patenkan. Saat ini banyak aplikasi di Internet yang menggunakan algoritma RSA ini.

Pada kriptografi kunci publik, seorang pengguna memiliki dua buah kunci yang saling berhubungan (secara matematik yang akan dijelaskan kemudian). Kunci pertama disebut **kunci publik**. Kunci ini boleh diketahui oleh umum. Bahkan kunci ini harus diketahui oleh pihak yang ingin mengirimkan informasi rahasia ke pengguna. Umumnya kunci publik ini disimpan di sebuah database. Kunci kedua disebut **kunci privat**. Kunci ini tidak boleh diketahui oleh siapa pun

kecuali oleh pengguna itu sendiri. Itulah sebabnya dia disebut privat. Mari kita ambil contoh pengamanan data dengan menggunakan kriptografi kunci publik ini. Sebelum dimulai, Anwar dan Broto masing-masing sudah memiliki sepasang kunci. Anwar memiliki Kpublik-A dan Kprivat-A sebagai pasangan kunci publik dan privatnya. Sementara itu Broto memiliki Kpublik-B dan Kprivat-B sebagai pasangan kunci publik dan privatnya.

Kunci publik milik Anwar dan Broto keduanya disimpan di database (website) umum sehingga dapat diakses oleh siapa saja. Misalkan Anwar ingin mengirimkan sebuah pesan kepada Broto. Anwar mencari kunci publik Broto. Setelah dicek di database Anwar menemukannya, Kpublik-B. Maka Anwar kemudian mengenkripsi pesannya dengan sebuah algoritma kunci publik (yang akan dijelaskan kemudian) dengan kunci Kpublik-B.

Algoritma kunci publik (seperti misalnya RSA, ECC) memiliki sifat bahwa jika dia dikunci oleh sebuah kunci publik, maka dia hanya dapat dibuka dengan menggunakan kunci privat pasangannya. Dalam contoh di atas, pesan dikunci dengan menggunakan Kpublik-B. Maka pesan di atas hanya dapat dibuka dengan Kprivat-B. Satu-satunya orang yang memiliki akses terhadap Kprivat-B adalah Broto. Dengan kata lain, pesan di atas hanya dapat dibuka oleh Broto. Anwar pun sebagai pengirim, setelah mengunci pesan tersebut dengan Kpublik-B, tidak dapat membuka pesan itu kembali.

Demikianlah proses enkripsi yang terjadi pada kriptografi kunci publik. Karena kunci yang digunakan untuk melakukan enkripsi berbeda dengan kunci

yang digunakan untuk proses dekripsi, maka sistem ini sering juga disebut dengan *asymetric cryptosystem*, kriptografi kunci asimetrik.

2.5 Permasalahan Kriptografi Gabungan

Sejak dikembangkannya kriptografi kunci publik, selalu timbul pertanyaan mana yang lebih baik antara kripto kunci publik dengan kripto kunci privat. Para pakar kriptografi mengatakan bahwa keduanya tidak dapat dibandingkan karena mereka memecahkan masalah dalam domain yang berbeda. Kriptografi kunci privat (simetrik) merupakan hal yang terbaik untuk mengenkripsi data. Kecepatannya dan keamanan akan *choosenciphertext attack* merupakan kelebihannya. Sementara itu kriptografi dengan menggunakan kunci publik dapat melakukan hal-hal lain lebih baik, misalnya dalam hal *key* management.

Karena masing-masing jenis kriptografi tersebut memiliki keuntungan tersendiri, maka aplikasi sekarang banyak yang menggabungkan keduanya (*hybrid system*). Kriptografi kunci public digunakan untuk melakukan pertukaran kunci (*key exchange*) dimana kunci yang dipertukarkan ini (*session key*) akan digunakan untuk enkripsi dengan kunci privat.

Aplikasi yang menggunakan mekanisme seperti di atas antara lain SSL, dan PGP.

2.6 Metoda Fungsi Hash

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 karakterkarakter 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:

a. parity checking
b. checksum
c. hash function

Fungsi Hash (*hash function*) merupakan fungsi yang bersifat satu arah dimana jika kita masukkan data, maka dia akan menghasilkan sebuah "checksum" atau "fingerprint" dari data tersebut. Sebuah pesan yang dilewatkan ke fungsi hash akan menghasilkan keluaran yang disebut *Message Authenticated Code* (MAC). Dilihat dari sisi matematik, hash function memetakan satu set data ke dalam sebuah set yang lebih kecil dan terbatas ukurannya.

Mari kita ambil sebuah contoh sederhana, yaitu fungsi matematik *modulus* (atau dalam pemrograman menggunakan *mod*). Hasil dari operasi mod adalah sisa pembagian bilangan bulat (integer). Sebagai contoh, "11 mod 7" menghasilkan nilai 4, karena 11 dibagi 7 menghasilkan nilai 1 dan sisanya adalah 4. Contoh lain "17 mod 7" menghasilkan bilangan 3, karena 17 dibagi 7 menghasilkan 2 dan

sisanya adalah 3. Demikian pula “18 mod 7” akan menghasilkan 4. Dalam sehari-hari, operasi modulus kita gunakan dalam penunjukkan jam, yaitu modulus 12.

Kalau kita perhatikan contoh di atas. Hasil dari opreasi *mod* tidak akan lebih besar dari angka pembaginya. Dalam contoh di atas, hasil “mod 7” berkisar dari 0 ke 6. Bilangan berapapun yang akan di-*mod*-kan akan menghasilkan bilangan dalam rentang itu. Tentu saja angka 7 bisa kita ganti dengan angka lain, misalnya sebuah bilangan prima yang cukup besar sehingga rentang bilangan yang dihasilkan bisa lebih besar.

Hal kedua yang perlu mendapat perhatian adalah bahwa diketahui hasil operasi modulus, kita tidak tahu bilangan asalnya. Jadi kalau diberitahu bahwa hasil operasi modulus adalah 4, bilangan awalnya bisa 11, 18, 25, dan seterusnya. Ada banyak sekali. Jadi, dalam aplikasinya nanti agak sukar mengkonstruksi sebuah pesan asli jika kita hanya tahu hasil dari fungsi hashnya saja.

Tentu saja operator mod sendirian tidak dapat digunakan sebagai fungsi hash yang baik. Ada beberapa persyaratan agar fungsi hasil baru dapat digunakan secara praktis. Misalnya, rentang dari hasil fungsi hash harus cukup sehingga probabilitas dua pesan yang berbeda akan menghasilkan keluaran fungsi hash yang sama. Perlu ditekankan kata “probabilitas”, karena secara teori pasti akan ada dua buah data yang dapat menghasilkan keluaran fungsi hash yang sama. Hal ini disebabkan rentang fungsi hash yang sangat jauh lebih kecil dibandingkan *space* dari inputnya. Tapi hal ini masih tidak terlalu masalah karena untuk membuat dua pesan yang samasama terbaca (intelligible) dan memiliki keluaran

fungsi hash yang sama tidaklah mudah. Hal yang terjadi adalah pesan (data) yang sama itu dalam bentuk sampah (*garbage*).

Syarat lain dari bagusnya sebuah fungsi hash adalah perubahan satu karakter (dalam berkas teks) atau satu bit saja dalam data lainnya harus menghasilkan keluaran yang jauh berbeda, tidak hanya berbeda satu bit saja. Sifat ini disebut *avalanche effect*.

Ada beberapa fungsi hash yang umum digunakan saat ini antara lain:

a. MD5 (*Message-Digest Algorithm 5*)
b. SHA (Secure Hash Algorithm)

2.6.1 MD5 (*Message-Digest Algorithm 5*)

MD5 (*Message-Digest Algorithm 5*), sebuah algoritma yang dibuat oleh Ron Rivest di tahun 1991, melakukan fungsi hash dengan menggunakan algoritma yang dijabarkan di RFC1321, "The MD5 Message-Digest Algorithm". Algoritma MD5 ini merupakan pengganti algoritma MD4 yang juga dibuat oleh Rivest. Hasil keluaran dari MD5 adalah sebuah nilai hash dalam 128-bit.

Salah satu aplikasi yang lazim menggunakan MD5 adalah pengamanan berkas password (/etc/passwd atau /etc/shadow) di sistem UNIX. Berkas password menyimpan data password dalam bentuk yang sudah terenkripsi dengan menggunakan DES. Implementasi awal dari sistem UNIX adalah menyimpan data password yang sudah terenkripsi tersebut langsung pada salah satu field di berkas password.

Meskipun sudah terenkripsi, penyimpanan data password yang sudah terenkripsi tersebut masih menimbulkan potensi lubang keamanan karena DES merupakan enkripsi yang *reversible*. anjang data yang dihasilkan oleh proses enkripsi juga bergantung kepada panjang data yang dimasukkan. Sehingga ada sedikit info tambahan mengenai kemungkinan panjangnya password. Apabila seseorang berhasil mendapatkan berkas password tersebut, dia bisa mencoba proses dekripsi yaitu dengan melakukan *brute force attack* dengan mencoba melakukan proses dekripsi.

MD5 menambahkan satu tingkat keamanan lagi. Kali ini data password yang disimpan bukanlah password yang terenkripsi saja, melainkan data yang terenkripsi yang sudah dilewatkan oleh MD5. Karena sifatnya yang satu arah, sangat sulit untuk mencari data password terenkripsi dengan basis data hasil fungsi MD5. Jadi skema penyimpanan data tersebut kira-kira seperti ini:

password > DES > password terenkripsi > MD5 > hashed encrypted password

Dengan cara ini, potensi untuk melakukan serangan brute force terhadap encrypted password menjadi lebih sukar. Satu-satunya cara untuk melakukan serangan brute force adalah dengan melakukan enkripsi juga dengan melalui jalan maju seperti di atas dan kemudian membandingkan hasil hashed encrypted passwordnya. Jika sama persis, maka kata yang dipilih sebagai percobaan sama dengan password yang ingin dipecahkan tersebut.

MD5 juga digunakan dalam autentikasi dengan menggunakan protocol CHAP (RFC 1994). Masih ada banyak aplikasi lain yang menggunakan MD5 ini.

Di tahun 1996 ditemukan kelemahan dari MD5 sehingga disarankan untuk menggantinya dengan menggunakan SHA-1. Di tahun 2004, ditemukan lagi kelemahan yang lebih serius sehingga penggunaan MD5 lebih dipertanyakan lagi. Xiaoyun Wang dan kawan-kawan menemukan kelemahan ini dan membuat makalah yang dipresentasikan di Crypto 2004.

Mereka menunjukkan bahwa ada tabrakan (collions) dimana dua buah data menghasilkan keluaran hash MD5 yang sama. Selain collision di MD5, mereka juga menemukan hal yang sama di MD5, HAVAL-128, dan RIPEMD.

2.6.2 SHA (Secure Hash Algorithm)

Hash adalah suatu fungsi yang memetakan suatu dokumen asli ke suatu dokumen hasil pemetaannya dengan sifat :

a. Dokumen hasil pemetaannya tidak dapat dipetakan kembali menjadi dokumen aslinya.
b. Tidak ada dua atau lebih dokumen asli dipetakan menjadi dokumen hasil pemetaan yang sama.

Pada protokol SET algoritma fungsi hash yang dipakai adalah Secure Hash Algorithm One (SHA.1). Dengan algoritma ini, besar dokumen hasil pemetaannya adalah 160 bit atau 20 byte 1. Hasil pemetaan fungsi *hash* dari suatu data atau pesan pada protokol SET disebut *message digest* atau untuk lebih singkatnya bisa disebut *digest* saja.

2.7 PHP

1.1 Pembuatan Login

Tentang Register, login, dan logout. Atau mungkin juga ada sebagian teman yang belum terlalu paham. Baiklah, Saya akan menjelaskan secara singkat saja yah.. Biasanya pada saat kita memasuki sebuah situs, kita diwajibkan untuk login terlebih dahulu. Tetapi sebelum login, kita harus mendaftar dulu nama, email, dan password yang kita inginkan untuk login nantinya. Baiklah, saya akan langsung menjelaskan scriptnya. Dalam contoh kali ini, saya membuat register, login, dan logout dalam 6 buah script. Yakni :

1. Index.php
2. Config.php
3. Daftar.php
4. Daftar2.php
5. Login.php
6. Dan logout.php

Pertama -tama yang harus dilakukan tentu saja adalah membuat database-nya terlebih dahulu. Buatlah database dengan nama ?reglog?. Lalu buat table dengan nama ?members?. Lalu masukkan

1.2 Pembuatan Security Image

Apa itu Security Image/ Security Code? Security image adalah suatu bentuk perlindungan terhadap suatu form dari serangan SPAM.

Sebagai contoh pernakah anda mengisi formulir pendaftaran untuk mendapatkan email account di yahoo??

Lihat bagian bawah diatas tombol submit, di situ ada security code yang muncul...dan anda harus mengisi security code tersebut Cara Kerjanya?

Pertama:

Formulir akan menampilkan security code yang di generate oleh script PHP, apabila page tersebut di refresh maka security code akan di generate lagi sehingga berubah codenya hal ini sangat berguna sekali untuk menangkal serangan spam.

Kedua:

Setelah di submit, script php akan memvalidasi data anda dan mencocok kan security code yang telah anda masuk kan apabila security code tidak sama, maka proses akan di batalkan apabila security code sama, maka proses akan dilanjutkan.

2.8 **SSL (Secure Socket Layer)**

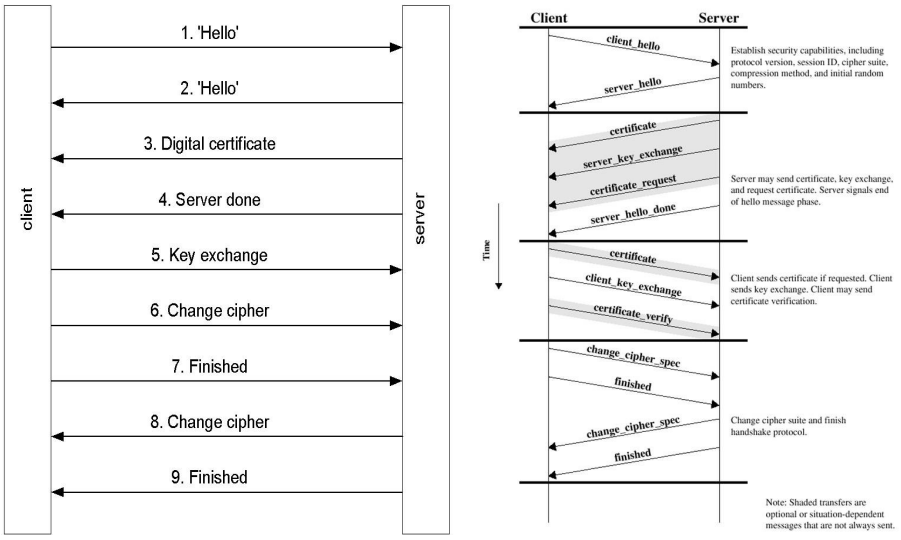
Secure Socket Layer (*SSL*) adalah protokol yang digunakan untuk *browsing web* secara aman, *SSL* bertindak sebagai protokol yang mengamankan komunikasi antara *client* dan *server, SSL* dikembangkan oleh *Netscape Communations* pada tahun 1994 Ada beberapa versi *SSL*, versi 2 dan versi 3, tetapi versi 3 paling banyak digunakan saat ini

2.8.1 Cara kerja TCP/IP (tanpa SSL)

1. Kebanyakan transmisi pesan di Internet dikirim sebagai kumpulan potongan pesan yang disebut **paket**.

2. *IP* bertanggung jawab untuk merutekan paket (lintasan yang dilalui oleh paket).
3. Pada sisi penerima, *TCP* memastikan bahwa suatu paket sudah sampai, menyusunnya sesuai nomor urut, dan menentukan apakah paket tiba tanpa mengalami perubahan.
4. Jika paket mengalami perubahan atau ada data yang hilang, *TCP* meminta pengiriman ulang.
5. *TCP/IP* tidak memiliki pengamanan komunikasi yang bagus.
6. *TCP/IP* tidak dapat mengetahui jika pesan diubah oleh pihak ketiga (*man-in-the-middle attack*).
7. *SSL* membangun hubungan (*connection*) yang aman antara dua *socket*, sehingga pengiriman pesan antara dua entitas dapat dijamin keamanannya.
8. Perlu dicatat bahwa *SSL* adalah protokol *client-server*, yang dalam hal ini *web browser* adalah *client* dan *website* adalah *server*.
9. *Client* yang memulai komunikasi, sedangkan *server* memberi respon terhadap permintaan *client*.
10. Protokol *SSL* tidak bekerja kalau tidak diaktifkan terlebih dahulu (biasanya dengan meng-klik tombol yang disediakan di dalam *web server*)

2.8.2 **Sub-protokol *handshaking***



Gambar 2.1 Sub-protokol *handshaking*

Sampai di sini, proses pembentukan kanal yang aman sudah selesai, Bila sub-protokol ini sudah terbentuk, maka

1. *http://* pada *URL* berubah menjadi
2. *https://* (*http secure*)