DAFTAR PUSTAKA

[1] Munir, Rinaldi. (2006). Diktat Kuliah IF2153 Matematika Diskrit. Program Studi Teknik Informatika, Institut Teknologi Bandung.

[2] Schneier, Bruce. (1996). Applied Cryptography 2nd. John Wiley & Sons.

[3] Menezes, A. J. and P. C. van Oorschot, S. A. Vanstone (2001). Handbook of Applied Cryptography

[4] http://en.wikipedia.org/wiki/Cryptography Tanggal akses: 2 Januari 2007 pukul 18:23.

[5] Prasetya, ISWB http://agorsiloku.wordpress.com/mengupas-rahasia-penyandian-informasi.htm Tanggal akses : 2 Januari 2007 pukul 18:41.

[6] Amwibowo. http://www.geocities.com_amwibowo/resource/komparasi/bab3/files Tanggal akses : 2 Januari 2007 pukul 19:12

[7] Insan, Budi. http://budi.insan.co.id/courses/ec5010/kriptografi Tanggal akses : 3 Januari 2007 pukul 15:20

pada harga informasi yang dienkripsi itu. Jika ya, maka untuk apa sang pencuri melakukannya? Untuk apa melakukan penyerangan dengan biaya 2 milyar rupiah kalau informasi yang dienkripsi hanya berharga Rp.10.000,- saja?

Kesimpulan

Kriptografi amat dibutuhkan untuk menyamarkan pesan-pesan yang memang sang pengirim tidak ingin pesannya diketahui orang lain sama-sekali. Karena itulah kriptografi digunakan agar apabila ada orang yang sanggup menyadap pesan yang dikirim, dia tidak akan mendapatkan manfaat apapun dari pesan tersebut.

Kriptografi modern ini pun terus berkembang tanpa batas. Metode-metode yang digunakan pun semakin banyak dan cukup menarik untuk ditelaah satu per satu.

Sejalan dengan itu penyerangan yang terjadi terhadap kriptografi pun semakin marak akhir-akhir ini. Metode-metode yang digunakan juga semakin beragam dan variatif.

Karena itu pembuat metode kriptografi pun harus semakin awas dan waspada terhadap segala kemungkinan penyerangan yang terjadi. Mereka pun harus semakin kreatif untuk mengembangkan metode-metode baru agar sistem enkripsi yang mereka buat tidak mudah ditembus oleh orang-orang yang berniat jahat untuk mengambil alih pesan yang mereka kirim.

Panjang kunci DES	Jaminan waktu untuk menemukan kunci
40-bit	0,4 detik
56-bit	7 jam
64-bit	74 jam 40 menit
128-bit	157.129.203.952.300.000 tahun

Tabel 2. Serangan brute-force pada DES

Protokol keamanan SSL (*Secure Socket Layer*) pada Netscape Navigator menggunakan algoritma RC4 40-bit untuk enkripsi simetrisnya. Tahun 1995, Damien Doligez menjebolnya menggunakan 120 komputer Unix yang terhubung pada jaringan dalam waktu 8 hari [Star 97]. Dengan cara seperti ini, dijamin bahwa dalam 15 hari kunci itu pasti ditemukan.

Panjang kunci RC4	Jaminan waktu untuk menemukan kunci
40-bit	15 hari
56-bit	2.691,49 tahun
64-bit	689.021,57 tahun
128-bit	12.710.204.652.610.000.000.000.000 tahun

Tabel 3. Serangan *brute-force* pada RC4

Panjang Kunci Asimetris

Sedangkan pada sistem enkripsi kunci publik-privat, yang memegang peranan dalam menjebol kunci privat adalah kesulitan mencari faktor prima bilangan yang sangat besar. Beberapa kunci yang dipergunakan 10 tahun lalu saja kini sama sekali tidak laik pakai seiring dengan perkembangan ilmu pengetahuan dan teknologi.

Kunci publik yang dimanfaatkan SSL adalah teknologi kunci publik 40-bit dari RSA, yang ternyata dapat dijebol dalam waktu 1,3 hari dengan 100 komputer menggunakan *brute-force attack* [DHMM 96].

Ronald Rivest, salah seorang penemu RSA, juga pernah menghitung bahwa untuk menemukan kunci RSA 512-bit dengan cara *brute-force attack* membutuhkan biaya 8,2 juta dollar AS [DaLe 96]. Untuk kasus tertentu, ini pun tidak aman. Kini perusahaan-perusahaan disarankan menggunakan kunci 2048 bit agar data aman sampai tahun 2015.

Prospek

Pada saat tulisan ini dibuat, ekspor teknologi enkripsi DES 56-bit keluar dari Amerika Serikat masih diizinkan. Untuk yang lainnya hanya diizinkan 40-bit. Setelah tanggal 31 Desember 1998, ekspor teknologi enkripsi DES dari Amerika Serikat hanya dibatasi sampai 40-bit saja, atau boleh saja tetap 56-bit, namun pengembang perangkat lunak itu harus menyediakan perangkat untuk membuka kunci itu juga [Star 97].

Panjang-pendeknya kunci dalam teknik-teknik enkripsi pada sistem perdagangan di Internet, akan menjadi salah satu titik lemah sistem perdagangan di Internet itu sendiri. Ada argumen yang menyatakan bahwa kalau pada suatu saat ukuran kunci publik-privat terasa terlalu pendek, maka panjangkan saja lagi kunci itu, tentu proses penyerangannya akan makin sulit. Hal ini memang benar, namun ada pertimbangan lain bahwa pengguna kunci tersebut harus bisa melakukan proses enkripsi-dekripsi dengan teknologi yang secara komersil memungkinkan. Terlihat di sini bahwa dibutuhkan ukuran kunci yang cukup panjang supaya aman, tapi tidak terlalu panjang agar memudahkan dalam penggunannya secara umum.

Beberapa teknik *brute-force attack* lain yang tidak akan dibahas panjang disini, seperti dengan penyebaran virus, komputasi paralel pada jaringan raksasa, undian Cina, atau penggunaan komputer biologis. Semua itu menunjukkan bahwa ada kemungkinan bahwa kunci bisa didapatkan dengan *brute-force attack.*

Satu hal yang patut dicatat adalah bukan berarti dengan mungkinnya suatu metoda enkripsi dijebol lantas metoda enkripsi itu tidak bermanfaat, namun yang penting apakah biaya untuk melakukan serangan itu lebih besar dari

Jika Anto menggandakan uang digitalnya lalu menggunakan uang digital yang sama itu dua kali, bank dapat mendeteksinya meskipun Badu tidak bisa. Badu memang 'membuka' identitas uang, namun hanya separuh-separuh. Kalau uang digital itu pernah diberikan Anto kepada Chandra, maka tentu Chandra juga pernah 'membuka' separuh identitas uang digital tadi secara acak. Nah, kemungkinan bahwa proses pembukaan identitas oleh Badu dan Chandra itu sama (maksudnya sama urutan pembukaannya, misalnya kiri-kiri-kanan-kiri-kanan, dan seterusnya) adalah 1 per 2n. Andaikan n cukup besar, katakanlah 16 saja, maka kemungkinan Badu dan Chandra secara acak membuka paruhan identitas dengan urutan sama adalah 1 : 65536. Artinya, jika Anto memberikan uangnya kepada dua orang yang berbeda, kemungkinan besar paruhan identitas yang dibuka juga berbeda. Jika saat otentikasi uang digital oleh bank ditemukan bahwa ada uang digital dengan nomor seri sama yang telah diuangkan, dan paruhan identitasnya berbeda, maka kemungkinan besar Anto menyerahkan uang digital yang sama kepada dua orang yang berbeda.

Sedangkan apabila Badu menguangkan uang digital yang sama dua kali, karena paruhan identitas dari uang digital yang diotentikasi itu sama persis dengan yang sudah tercatat, maka kemungkinan besar uang itu diberikan Anto kepada orang yang sama. Badulah yang ketahuan menguangkan uang digital yang sama dua kali. Penggunaan uang digital yang sama dua kali dikenal dengan istilah pembelanjaan ganda (*double spending*).

Panjang Kunci

Panjang Kunci Simetris

Meskipun ada beberapa cara bagi seorang kriptoanalis untuk memecahkan pesan rahasia, namun cara yang cukup umum dilakukan adalah dengan melakukan *brute-force attack*. Dengan cara ini, seorang penyerang mencoba seluruh kemungkinan kunci yang ada, sampai menemukan sebuah kunci yang jika dipergunakan untuk mendekripsi pesan yang disandikan akan memunculkan suatu pesan yang bermakna. Tentunya cara ini bermanfaat hanya jika sudah diketahui algoritmanya, namun tidak diketahui kuncinya apa.

PIN 5 digit berarti biasanya ada 100.000 kombinasi. Kelihatannya cukup, namun sebenarnya kurang. Dengan sebuah komputer pribadi saja bisa dengan mudah diselesaikan. Salah satu pencegahannya adalah dengan pembatasan seberapa banyak pemakai dapat mencoba memasukkan PIN. Biasanya dibatasi tiga kali.

Berikut ini diberikan contoh dari *brute-force attack* pada suatu algoritma 'geser pada papan ketik QWERTY':

Sandi *zsdyrtvstf* dicoba dengan kunci 3 menjadi *bjkwpqmjql*
Sandi *zsdyrtvstf* dicoba dengan kunci 2 menjadi *nkleqwzkwa*
Sandi *zsdyrtvstf* dicoba dengan kunci 1 menjadi *mastercard*

Ternyata kunci 1 cocok, karena dalam pesan yang disandikan itu mungkin ada transaksi yang menggunakan kartu kredit 'mastercard'. Dengan menggunakan kunci yang sama, kemudian penyerang berusaha mendekripsikan bagian-bagian lain dari pesan, mungkin berusaha mengambil nomor kartu kreditnya. Kunci itu juga dapat dipakai untuk keperluan lain, misalnya untuk melakukan penipuan (*spoofing*).

DES, sebuah algoritma simetris, memiliki panjang kunci 56-bit, artinya ada 2^{56} kemungkinan kunci. Sedangkan peraturan di Amerika Serikat yang akan diberlakukan pada tahun 1998 nanti akan melarang ekspor teknologi enkripsi lebih dari 40-bit. Sedangkan untuk keperluan dalam negeri Amerika Serikat, kunci 128-bit masih diizinkan penggunaannya [Star 97].

Tahun 1995, Michael Wierner merancang sebuah chip yang mengkhususkan diri untuk melakukan *brute-force attack* pada metoda enkripsi DES [Schn 96]. Chip tersebut dapat menemukan kunci rahasia dalam waktu rata-rata 3,5 jam dan kunci itu dijamin dapat ditemukan dalam waktu 7 jam. Harga pembuatannya adalah 1 juta dollar AS. Sesuai hukum Moore, setiap 18 bulan kemampuan komputer meningkat 2 kali lipat untuk harga yang sama. Maka, pada tahun 2000, harga chip itu hanya berkisar 100.000 dolar AS. Harga ini masih dalam jangkauan daya beli beberapa mafia kejahatan terorganisir. Karena itu, kini disarankan untuk menggunakan DES dengan kunci 112-bit.

Badu. Badu kurang lebih tahu apa isi pesan di amplop ke-100 itu. Protokol tanda tangan buta (*blind signature*) bekerja sebagai berikut:

1. Anto 'mengalikan' dokumen (yang akan ditandatangani) dengan sebuah faktor pembuta.
2. Anto mengirimkan dokumen itu kepada Badu
3. Badu menandatangani dokumen itu
4. Badu mengembalikan dokumen yang sudah ditandatangani tadi kepada Anto
5. Anto membaginya dengan faktor pembuta, sehingga mendapatkan dokumen yang asli sudah tertandatangani oleh Badu.

Protokol Uang Digital

1. Deskripsi Protokol

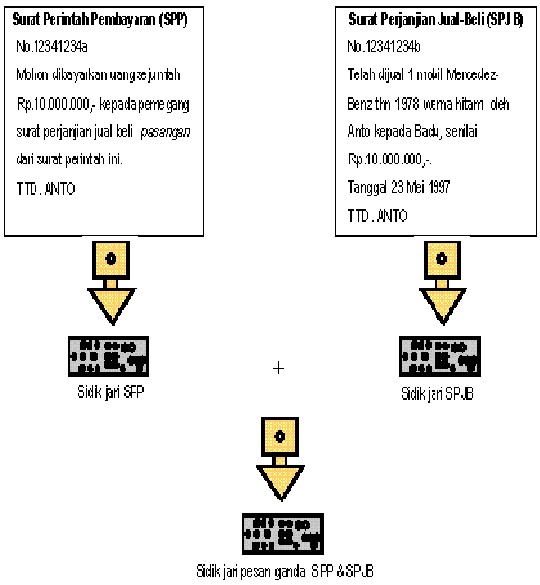
Berdasarkan beberapa teori penunjang di atas, maka dapatlah dibangun suatu protokol untuk uang digital. David Chaum, memiliki beberapa paten atas protokol uang digital yang diciptakannya. Berikut ini dijelaskan salah satu protokol uang digital:

1. Anto menyiapkan n lembar uang dengan nilai tertentu. Setiap uang diberi nomor seri acak X yang cukup panjang, sehingga kemungkinan 2 bilangan acak sama kecil sekali. Dalam setiap uang juga ada n (I_1, I_2, …, I_n) string identifikasi yang berguna untuk memberikan informasi mengenai pemilik uang, yakni Anto. Anto kemudian memecah tiap-tiap string identitas diri itu tadi menjadi dua bagian dengan menggunakan protokol pemecahan rahasia. Lantas Anto melakukan bit-komitmen pada setiap pecahan. Contoh uang yang disiapkan adalah:

   Nilai: Rp.1.000,-
   Nomor seri acak: X
   String identitas: $I_1 = (I_{1L}, I_{1R})$
   $I_2 = (I_{2L}, I_{2R})$
   . . . .
   $I_n = (I_{nL}, I_{nR})$

1. Anto memasukkan uang itu kedalam yang juga disisipi kertas karbon amplop (mengalikan uang dengan faktor pembuta), lalu memberikannya kepada bank.
2. Bank akan meminta Anto untuk membuka n - 1 amplop itu secara acak. Bank memeriksa apakah semua uang tersebut memiliki nilai yang sama. Bank juga meminta kepada Anto untuk membuktikan kejujuran dirinya saat menuliskan string identifikasi pada uang itu, dengan cara menggabungkan pasangan-pasangan string identifikasi.
3. Jika bank merasa bahwa Anto tidak melakukan kecurangan, maka bank akan menandatangani uang terakhir yang masih di dalam amplop itu dan menyerahkannya kepada Anto. Tanda tangan bank akan menembus amplop dan kertas karbon sehingga uang di dalamnya tertandatangani.
4. Anto membuka amplop. Uang siap dipakai.
5. Anto menyerahkan uang kepada Badu. Badu sebagai penerima uang, akan memeriksa apakah tanda tangan bank pada uang itu absah.
6. Badu akan menyuruh Anto untuk membuka salah satu sisi dari setiap string identifikasi di setiap uang dengan cara memberikan string pemilih sepanjang n-bit. Artinya, jika string pemilih itu $b_1, b_2, \ldots, b_n$ maka Anto harus membuka sisi kiri atau kanan dari I_i, tergantung apakah b_i itu 0 atau 1.
7. Setelah itu Badu membawa uang tersebut ke bank. Bank akan memeriksa apakah nomor seri uang tersebut sudah pernah diterima oleh bank. Kalau belum ada, maka uang tersebut dinyatakan sah.
8. Jika nomor seri uang itu sudah pernah diterima oleh bank, maka bank akan memeriksa string identitas yang sudah terbuka pada uang itu dan membandingkannya dengan string identitas pada uang dengan nomor seri sama yang pernah diterima bank sebelumnya. Jika ternyata string identitas itu sama, maka berarti Badu yang menggandakan uang tersebut. Namun jika berbeda, maka berarti Anto yang menggandakan uang digital tersebut.

2. Pembelanjaan Ganda



Gambar 13. Pembuatan sidik jari pesan ganda

Jika sidik jari pesan ganda SPP & SPJB dienkripsi dengan kunci privat Anto, maka akan menjadi tanda tangan pesan ganda (*dual-signature*) Anto untuk kedua perjanjian tersebut [ViMa 97].

Protokol Pembagian Rahasia

Jika Anto memiliki rahasia, ia dapat memberikan 'separuh' rahasia itu kepada Badu dan 'separuh' rahasia itu kepada Chandra. Badu, yang menerima paruh pertama rahasia Anto, tidak bisa mengetahui apa isi rahasia itu. Demikian pula dengan Chandra. Namun, jika Badu dan Chandra menggabungkan potongan-potongan rahasia itu, maka akan tergambar rahasia Anto. Pembagian rahasia (*secret splitting)* dapat dilakukan dengan cara:

1. Anto membuat seuntai string acak R yang panjangnya sama dengan pesan rahasia M.
2. Anto melakukan operasi XOR antara M dengan R, sehingga menghasilkan S.
3. Anto memberikan R kepada Badu dan S kepada Chandra
4. Jika Badu dengan Chandra bertemu, maka mereka sanggup mendapatkan pesan rahasia M dengan cara melakukan operasi XOR antara S dengan R.

Protokol Komitmen-Bit

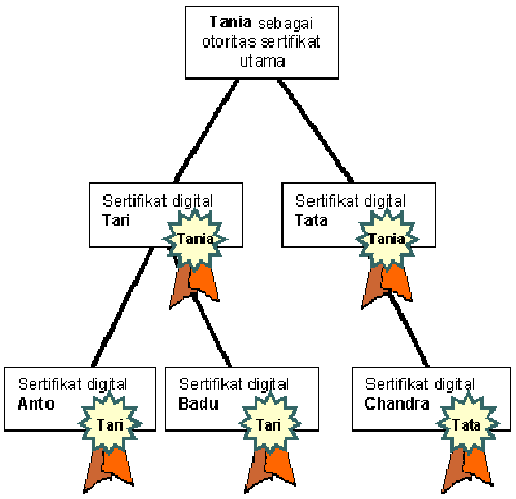
Protokol ini bermanfaat kalau misalnya Anto hendak membuat suatu pernyataan atau komitmen (katakanlah suatu string binari 1000), namun Anto tak ingin agar Badu mengetahui isi pernyatan tersebut sebelum saatnya. Badu harus merasa yakin bahwa Anto pada saatnya nanti, benar-benar mengeluarkan isi pernyataan yang sebenarnya saat melakukan komitmen, dan tidak mengeluarkan pernyataan yang sudah diubah (misalnya mengubah string tadi menjadi 1001). Ada beberapa jenis protokol komitmen-bit, namun di bawah ini hanya dijelaskan salah satu diantaranya, yakni dengan fungsi *hash* satu arah:

1. Anto membuat dua buah string secara acak, yakni R_1 dan R_2
2. Anto menggabungkan kedua string acak itu ke dalam pernyataannya (b) yang akan dikomitmenkan menjadi (R_1, R_2, b)
3. Anto menghitung *hash* dari gabungan string itu, *Hash*(R_1, R_2, b).
4. Anto kemudian mengirimkan *hash* tersebut beserta R_1 kepada Badu. Badu akan menyimpannya untuk pemeriksaan nanti.
5. Jika sudah tiba saatnya untuk menunjukkan pernyataannya, Anto memberikan seluruh string (R_1, R_2, b) kepada Badu.
6. Badu memeriksa fungsi *hash* dari (R_1, R_2, b). Jika cocok dengan *hash* yang diperiksanya dulu, maka pernyataan Anto tidak diubah.

Tanda Tangan Buta

Badu disodori 100 amplop tertutup oleh Anto. Amplop itu berisi secarik pesan dan kertas karbon. Badu membuka 99 amplop secara acak. Jika seluruh amplop yang dibuka ternyata berisi pesan yang mirip, maka Badu dapat merasa bahwa amplop ke-100 juga berisi pesan yang mirip pula. Namun, jika satu saja dari 99 amplop tadi ada yang isi berbeda dari yang lain, maka Badu dapat mencurigai bahwa isi amplop ke-100 bisa saja juga tidak mirip dengan isi ke-98 amplop lainnya.

Dalam kasus dimana ternyata ke-99 amplop yang dibuka secara acak tadi berisi pesan yang mirip, maka dengan keyakinan yang cukup tinggi Badu berani menandatangani amplop terakhir yang belum dibuka. Tanda tangan Badu akan menembus amplop dan kertas karbon, sehingga pesan dalam amplop akan tertandatangani oleh



Gambar 12. Contoh hirarki otoritas sertifikat digital

Serangan terhadap sistem yang memiliki pengamanan dengan sertifikat digital sulit dilakukan. Jelas Edi tidak mendapatkan apa-apa kalaupun ia memainkan ulang percakapan antara Anto dan Chandra. Edi membutuhkan kunci privat untuk bisa membuka pesan-pesan yang dipertukarkan, padahal kunci privat itu tidak ada di dalam sertifikat digital.

Penukaran sertifikat digital Chandra dengan sertifikat digital Maman akan segera diketahui, karena sertifikat digital itu pasti berbeda. Sedangkan jika sertifikat yang dipertukarkan antara Chandra dan Anto tidak diganti, tetapi yang diganti oleh Maman adalah pesan yang dipertukarkan, maka tentu ada ketidakcocokan dalam pemeriksaan tanda tangan digital.

Secara teoritis keunggulan dari tanda tangan digital adalah kemampuan untuk melakukan proses otentikasi secara *off-line*. Pemeriksa cukup memiliki kunci publik dari OS utama untuk mengetahui sah-tidaknya kunci publik dari lawan bicaranya. Selain itu untuk meningkatkan keamanan, kunci publik OS utama bisa saja diintegrasikan dalam program aplikasi. Namun kenyataannya, karena ada kemungkinan sertifikat digital tersebut hilang, tercuri atau identitas pemilik sertifikat berubah (perubahan alamat surat elektronik atau nomor KTP misalnya), maka sertifikat digital perlu diperiksa keabsahannya dengan melihat daftar sertifikat terbatalkan (*certificate revocation list*) yang disimpan oleh OS.

Tanda Tangan Pesan Ganda

Andaikan Anto membuat perjanjian jual-beli dengan Badu. Untuk masalah pembayaran, Anto menginstruksikan bank untuk memberikan kepada Badu sejumlah uang sesuai dengan perjanjian jual-beli, namun Anto tidak ingin agar bank mengetahui isi perjanjian jual-beli itu.

1. Anto membuat sidik jari dari SPP (yaitu *Hash*(SPP)) dan sidik jari SPJB (yakni *Hash*(SPJB)).
2. Kemudian, Anto membuat sebuah sidik jari baru dari gabungan kedua sidik jari sebelumnya (*Hash* ((*Hash*(SPP) + *Hash*(SPJB))). Hasil *hash* tersebut dinamakan sidik jari pesan ganda SPP & SPJB.
3. Anto menyerahkan surat perjanjian jual belinya kepada Badu. Selain itu Anto juga menyerahkan surat perintah pembayaran beserta sidik jari pesan ganda SPP & SPJB kepada bank.
4. Saat Badu ingin mengambil uang di bank, Badu membuat sidik jari dari surat perjanjian jual beli (SPJB). Badu menyerahkan sidik jari SPJB kepada bank.
5. Bank membuat sidik jari dari surat perintah pembayaran (SPP).
6. Bank menggabungkan sidik jari SPP dengan sidik jari SPJB yang diterimanya dari Badu, kemudian meng-*hash*-nya sehingga dihasilkan sidik jari pesan ganda SPP & SPJB.
7. Jika sidik jari pesan ganda SPP & SPJB yang baru dibuat itu sama dengan yang telah diberikan oleh Anto, maka bank menjalankan kewajibannya kepada Badu.

Masalah Pertukaran Kunci Publik

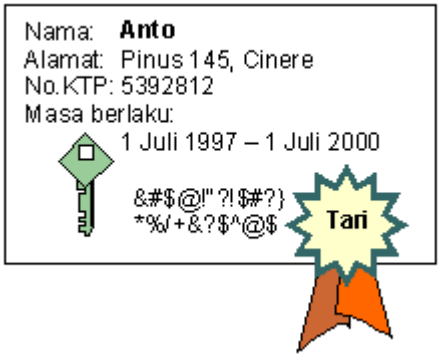
Anto hendak mengirimkan Badu suatu dokumen rahasia. Jika mereka belum pernah bertemu sebelumnya, tentu Badu harus mengirimkan kunci publiknya kepada Anto agar Anto dapat melakukan enkripsi yang pesannya hanya dapat dibuka oleh Badu. Demikian juga pula sebaliknya, Anto harus mengirimkan kepada Badu kunci publiknya agar Badu dapat memeriksa keaslian tanda tangan Anto pada pesan yang dikirim. Dengan cara ini Anto dapat memastikan pesan itu sampai ke tujuannya, sedangkan Badu dapat merasa yakin bahwa pengirim pesan itu adalah Anto.

Masalah yang muncul adalah bagaimana mereka dapat saling bertukar kunci dengan aman? Bisa saja di tengah pertukaran kunci-kunci publik milik Anto dan Budi itu diganti dengan kunci publik milik Maman. Dengan begitu Maman dengan bebas dapat menyadap dan mengubah seluruh informasi. Inilah suatu contoh dari *man-in-the-middle attack.*

Anto dan Badu harus sama-sama yakin bahwa kunci-kunci publik yang mereka dapatkan benar-benar otentik. Mereka bisa mendapatkannya dari sesorang yang dipercaya, Tari misalnya. Setiap anggota jaringan diasumsikan telah memiliki saluran komunikasi pribadi yang aman dengan Tari. Saluran inilah yang dimanfaatkan untuk mengirim kunci publik Badu ke Anto (dan sebaliknya). Tari menjadi penjamin keabsahan kunci jika Anto dan Badu sebelumnya tidak pernah bertukar kunci publik. Skenario ini tetap membutuhkan kunci-kunci kriptografi lagi (baik itu kunci simetris ataupun kunci asimetris) untuk pengamanan saluran komunikasi antara Tari dengan Anto atau Badu.

Sertifikat Digital

Masalah di atas dapat dipecahkan dengan penggunaan sertifikat digital. Tari tidak lagi setiap saat menjadi penukar kunci, namun Tari cukup menandatangani kunci publik milik setiap orang di jaringan tersebut. Sebenarnya dalam sertifikat tersebut tak hanya berisi kunci publik, namun dapat berisi pula informasi penting lainnya mengenai jati diri pemilik kunci publik, seperti misalnya nama, alamat, pekerjaan, jabatan, perusahaan dan bahkan *hash* dari suatu informasi rahasia. Semua orang mempercayai otoritas Tari dalam memberikan tanda tangan, sehingga orang-orang dalam jaringan itu merasa aman menggunakan kunci publik yang telah ditandatangani Tari.



Gambar 11. Contoh sertifikat digital

Jika Maman berhasil mencuri sertifikat digital yang dipertukarkan antara Anto dan Badu, serta menggantinya dengan sertifikat digital milik dirinya sendiri, maka Anto dan Badu dapat segera melihat bahwa sertifikat digital yang diterimanya bukan 'lawan bicara' yang semestinya.

Bagaimana jika Chandra – yang berada di luar jaringan Tari – hendak berkomunikasi dengan Anto? Chandra memiliki juga sertifikat, tetapi tidak ditandatangani oleh Tari, melainkan oleh Tata, seseorang yang dipercaya dalam jaringan tempat Chandra berada. Tari dan Tata adalah otoritas sertifikat (*certificate authority*), yaitu pihak-pihak yang berwenang memberikan sertifikat. Namun Anto tidak mengenal dan tidak mempercayai Tata. Masalah ini dapat diselesaikan jika ada otoritas sertifikat (OS) yang kedudukannya lebih tinggi dari Tata dan Tari – katakanlah Tania. Tania memberikan pengesahan kepada Tata dan Tari. Jadi ada hirarki dari sertifikat digital. Jika Tania berada pada kedudukan hirarki yang paling tinggi, maka Tania disebut otoritas sertifikat utama (*root certificate authority*).

Anto mempercayai tanda tangan Tari. Namun karena Tari sendiri keberadaannya disahkan oleh Tania, tentunya Anto harus mengakui otoritas Tania. Jika Tania memberikan pengesahan kepada OS lain dibawahnya, seperti Tata, maka dengan merunut struktur hirarki percabangan OS, Anto dapat memeriksa kebenaran sertifikat digital milik Chandra yang disahkan oleh Tata.

Gambar 9. Membuat sidik jari pesan

Fungsi *hash* untuk membuat sidik jari tersebut dapat diketahui oleh siapapun, tak terkecuali, sehingga siapapun dapat memeriksa keutuhan dokumen atau pesan tertentu. Tak ada algoritma rahasia dan umumnya tak ada pula kunci rahasia.

Jaminan dari keamanan sidik jari berangkat dari kenyataan bahwa hampir tidak ada dua pre-image yang memiliki *hash*-value yang sama. Inilah yang disebut dengan sifat *collision free* dari suatu fungsi *hash* yang baik. Selain itu, sangat sulit untuk membuat suatu pre-image jika hanya diketahui *hash*-valuenya saja.

Contoh algoritma fungsi *hash* satu arah adalah MD-5 dan SHA. *Message authentication code* (MAC) adalah salah satu variasi dari fungsi *hash* satu arah, hanya saja selain *pre-image,* sebuah kunci rahasia juga menjadi input bagi fungsi MAC.

Tanda Tangan Digital

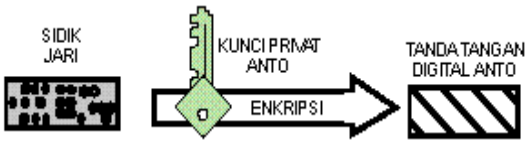
Badu memang dapat merasa yakin bahwa sidik jari yang datang bersama pesan yang diterimanya memang berkorelasi. Namun bagaimana Badu dapat merasa yakin bahwa pesan itu berasal dari Anto? Bisa saja saat dikirimkan oleh Anto melalui saluran komunikasi yang tidak aman, pesan tersebut diambil oleh Maman. Maman kemudian mengganti isi pesan tadi, dan membuat lagi sidik jari dari pesan yang baru diubahnya itu. Lalu, Maman mengirimkan lagi pesan beserta sidik jarinya itu kepada Badu, seolah-oleh dari Anto.

Untuk mencegah pemalsuan, Anto membubuhkan tanda tangannya pada pesan tersebut. Dalam dunia elektronik, Anto membubuhkan tanda tangan digitalnya pada pesan yang akan dikirimkan untuk Badu sehingga Badu dapat merasa yakin bahwa pesan itu memang dikirim oleh Anto.

Sifat yang diinginkan dari tanda tangan digital diantaranya adalah:

1. Tanda tangan itu asli (otentik), tidak mudah ditulis/ditiru oleh orang lain. Pesan dan tanda tangan pesan tersebut juga dapat menjadi barang bukti, sehingga penandatangan tak bisa menyangkal bahwa dulu ia tidak pernah menandatanganinya.
2. Tanda tangan itu hanya sah untuk dokumen (pesan) itu saja. Tanda tangan itu tidak bisa dipindahkan dari suatu dokumen ke dokumen lainnya. Ini juga berarti bahwa jika dokumen itu diubah, maka tanda tangan digital dari pesan tersebut tidak lagi sah.
3. Tanda tangan itu dapat diperiksa dengan mudah.
4. Tanda tangan itu dapat diperiksa oleh pihak-pihak yang belum pernah bertemu dengan penandatangan.
5. Tanda tangan itu juga sah untuk kopi dari dokumen yang sama persis.

Meskipun ada banyak skenario, ada baiknya kita perhatikan salah satu skenario yang cukup umum dalam penggunaan tanda tangan digital. Tanda tangan digital memanfaatkan fungsi *hash* satu arah untuk menjamin bahwa tanda tangan itu hanya berlaku untuk dokumen yang bersangkutan saja. Bukan dokumen tersebut secara keseluruhan yang ditandatangani, namun biasanya yang ditandatangani adalah sidik jari dari dokumen itu beserta *timestamp*-nya dengan menggunakan kunci privat. *Timestamp* berguna untuk menentukan waktu pengesahan dokumen.



Gambar 10. Pembuatan tanda tangan digital

Keabsahan tanda tangan digital itu dapat diperiksa oleh Badu. Pertama-tama Badu membuat lagi sidik jari dari pesan yang diterimanya. Lalu Badu mendekripsi tanda tangan digital Anto untuk mendapatkan sidik jari yang asli. Badu lantas membandingkan kedua sidik jari tersebut. Jka kedua sidik jari tersebut sama, maka dapat diyakini bahwa pesan tersebut ditandatangani oleh Anto.

6. Bejo menerima C dari Ahmad dan membuka teks-terang dengan fungsi $P = D(C, K_{privat[Bejo]})$

Hal yang sama terjadi apabila Bejo hendak mengirimkan pesan ke Ahmad

1. Bejo mengenkripsi teks-terang P ke Ahmad dengan fungsi $C = E(P, K_{publik[Ahmad]})$
2. Ahmad menerima C dari Bejo dan membuka teks-terang dengan fungsi $P = D(C, K_{privat[Ahmad]})$

Algoritma -Algoritma Sandi Kunci-Asimetris

- Knapsack
- RSA - Rivert-Shamir-Adelman
- Diffie-Hellman

Fungsi Hash Kriptografis

Fungsi hash Kriptografis adalah fungsi hash yang memiliki beberapa sifat keamanan tambahan sehingga dapat dipakai untuk tujuan keamanan data. Umumnya digunakan untuk keperluan autentikasi dan integritas data. Fungsi hash adalah fungsi yang secara efisien mengubah string input dengan panjang berhingga menjadi string output dengan panjang tetap yang disebut nilai hash.

Sifat-Sifat Fungsi Hash Kriptografi

- Tahan preimej (*Preimage resistant*): bila diketahui nilai hash *h* maka sulit (secara komputasi tidak layak) untuk mendapatkan *m* dimana *h* = hash(*m*).
- Tahan preimej kedua (*Second preimage resistant*): bila diketahui input m_1 maka sulit mencari input m_2 (tidak sama dengan m_1) yang menyebabkan hash(m_1) = hash(m_2).
- Tahan tumbukan (*Collision-resistant*): sulit mencari dua input berbeda m_1 dan m_2 yang menyebabkan hash(m_1) = hash(m_2)

Algoritma-Algoritma Fungsi Hash Kriptografi

Beberapa contoh algoritma fungsi hash Kriptografi:

- MD4
- MD5
- SHA-0
- SHA-1
- SHA-256
- SHA-512

Fungsi *Hash* Satu Arah

Kini akan dibahas mengenai keutuhan pesan saat dikirimkan. Bagaimana jika Anto mengirimkan surat pembayaran kepada Badu sebesar 1 juta rupiah, namun di tengah jalan Maman (yang ternyata berhasil membobol sandi entah dengan cara apa) membubuhkan angka 0 lagi dibelakangnya sehingga menjadi 10 juta rupiah? Di mata Tari, pesan tersebut harus utuh, tidak diubah-ubah oleh siapapun, bahkan bukan hanya oleh Maman, namun juga termasuk oleh Anto, Badu dan gangguan pada transmisi pesan (*noise*). Hal ini dapat dilakukan dengan fungsi *hash* satu arah (*one-way hash function*), yang terkadang disebut sidik jari (*fingerprint*), *hash*, *message integrity check,* atau *manipulation detection code*.

Saat Anto hendak mengirimkan pesannya, dia harus membuat sidik jari dari pesan yang akan dikirim untuk Badu. Pesan (yang besarnya dapat bervariasi) yang akan di-*hash* disebut *pre-image,* sedangkan outputnya yang memiliki ukurannya tetap, disebut *hash-value* (nilai *hash*). Kemudian, melalui saluran komunikasi yang aman, dia mengirimkan sidik jarinya kepada Badu. Setelah Badu menerima pesan si Anto – tidak peduli lewat saluran komunikasi yang mana – Badu kemudian juga membuat sidik jari dari pesan yang telah diterimanya dari Anto. Kemudian Badu membandingkan sidik jari yang dibuatnya dengan sidik jari yang diterimanya dari Anto. Jika kedua sidik jari itu identik, maka Badu dapat yakin bahwa pesan itu utuh tidak diubah-ubah sejak dibuatkan sidik jari yang diterima Badu. Jika pesan pembayaran 1 juta rupiah itu diubah menjadi 10 juta rupiah, tentunya akan menghasilkan nilai *hash* yang berbeda.

|| = operator penyambungan (*concatenation*)
MSB = Most Significant Byte
LSB = Least Significant Byte

Jika $m = n$, maka mode *OFB n*-bit adalah seperti pada Gambar 6. *OFB* menggunakan skema umpan balik dengan mengaitkan blok plainteks bersama-sama sedemikian sehingga cipherteks bergantung pada semua blok plainteks sebelumnya. Skema enkripsi dan dekripsi dengan mode *OFB* dapat dilihat pada Gambar 7.

Stream-Cipher

Stream-cipher adalah algoritma sandi yang mengenkripsi data persatuan data, seperti bit, byte, nible atau per lima bit(saat data yang di enkripsi berupa data Boudout). Setiap mengenkripsi satu satuan data di gunakan kunci yang merupakan hasil pembangkitan dari kunci sebelum.

Algoritma-algoritma sandi kunci-simetris

Beberapa contoh algoritma yang menggunakan kunci-simetris:

- DES - Data Encryption Standard
- blowfish
- twofish
- MARS
- IDEA
- 3DES - DES diaplikasikan 3 kali
- AES - Advanced Encryption Standard, yang bernama asli rijndael

***Advanced Encryption Standard* (*AES*)**

Advanced Encryption Standard (*AES*) merupakan sebuah algoritma kriptografi simetri yang beroperasi dalam bentuk blok 128-bit. *AES* mendukung panjang kunci 128-bit, 192-bit, dan 256-bit.

Algoritma Sandi Kunci-Asimetris

Skema ini adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Skema ini disebut juga sebagai sistem kriptografi kunci publik karena kunci untuk enkripsi dibuat untuk diketahui oleh umum (public-key) atau dapat diketahui siapa saja, tapi untuk proses dekripsinya hanya dapat dilakukan oleh yang berwenang yang memiliki kunci rahasia untuk mendekripsinya, disebut private-key. Dapat dianalogikan seperti kotak pos yang hanya dapat dibuka oleh tukang pos yang memiliki kunci tapi setiap orang dapat memasukkan surat ke dalam kotak tersebut. Keuntungan algoritma model ini, untuk berkorespondensi secara rahasia dengan banyak pihak tidak diperlukan kunci rahasia sebanyak jumlah pihak tersebut, cukup membuat dua buah kunci, yaitu kunci publik bagi para korensponden untuk mengenkripsi pesan, dan kunci privat untuk mendekripsi pesan. Berbeda dengan skema kunci-simetris, jumlah kunci yang dibuat adalah sebanyak jumlah pihak yang diajak berkorespondensi.

Fungsi Enkripsi dan Dekripsi Algoritma Sandi Kunci-Asimetris

Apabila Ahmad dan Bejo hendak bertukar berkomunikasi, maka:

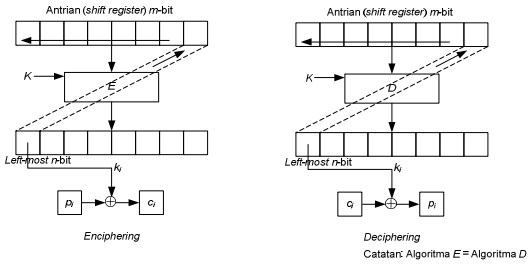
1. Ahmad dan Bejo masing-masing membuat 2 buah kunci
    1. Ahmad membuat dua buah kunci, kunci-publik $K_{publik[Ahmad]}$ dan kunci-privat $K_{privat[Ahmad]}$
    2. Bejo membuat dua buah kunci, kunci-publik $K_{publik[Bejo]}$ dan kunci-privat $K_{privat[Bejo]}$
2. Mereka berkomunikasi dengan cara:
    3. Ahmad dan Bejo saling bertukar kunci-publik. Bejo mendapatkan $K_{publik[Ahmad]}$ dari Ahmad, dan Ahmad mendapatkan $K_{publik[Bejo]}$ dari Bejo.
    4. Ahmad mengenkripsi teks-terang P ke Bejo dengan fungsi $C = E(P, K_{publik[Bejo]})$
    5. Ahmad mengirim teks-sandi C ke Bejo

***Output-Feedback* (*OFB*)**

Pada mode *OFB*, data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit, dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode *OFB*-nya disebut *OFB* 8-bit. Secara umum *OFB n*-bit mengenkripsi plainteks sebanyak *n* bit setiap kalinya, yang mana $n \leq m$ (m = ukuran blok). Mode *OFB* membutuhkan sebuah antrian (*queue*) yang berukuran sama dengan ukuran blok masukan.

Tinjau mode *OFB n*-bit yang bekerja pada blok berukuran *m*-bit. Algoritma enkripsi dengan mode *OFB* adalah sebagai berikut (lihat Gambar 6):

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Enkripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil enkripsi dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan *m-n* bit lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan. *n* bit paling kiri dari hasil enkripsi juga berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari plainteks menjadi *n*-bit pertama dari cipherteks.
3. *m-n* bit plainteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.

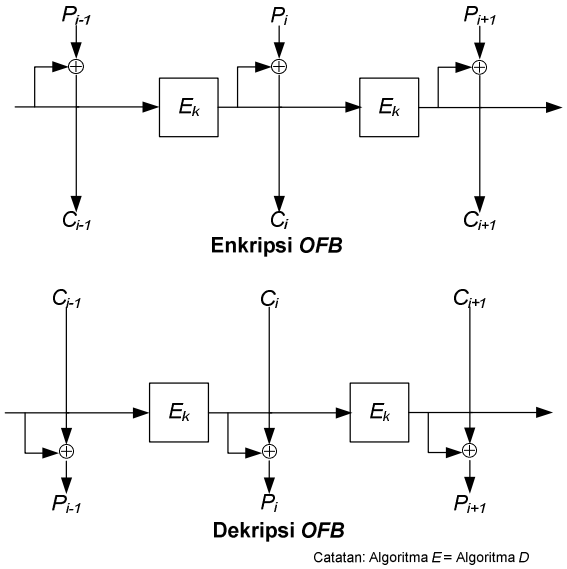


Gambar 7. Mode *OFB* n-bit

Sedangkan, algoritma dekripsi dengan mode *OFB* adalah sebagai berikut (lihat Gambar 6):

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Dekripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil dekripsi dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan *m-n* bit lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan. *n* bit paling kiri dari hasil dekripsi juga berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari cipherteks menjadi *n*-bit pertama dari plainteks.
3. *m-n* bit cipherteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.

Baik enkripsi maupun dekripsi, algoritma *E* dan *D* yang digunakan sama. Mode *OFB n*-bit yang bekerja pada blok berukuran *m*-bit dapat dilihat pada Gambar 6.



Gambar 8. Enkripsi dan Dekripsi *OFB n*-bit untuk blok n-bit

Secara formal, mode *OFB n*-bit dapat dinyatakan sebagai:

Proses Enkripsi:

$$C_i = P_i \oplus MSB_m(E_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, LSB_n(E_k(X_i))$$

Proses Dekripsi:

$$P_i = C_i \oplus MSB_m(D_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \,||\, LSB_n(E_k(X_i))$$

yang dalam hal ini:

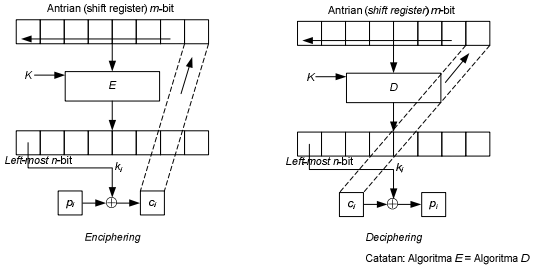
X_i = isi antrian dengan X_i adalah *IV*
E = fungsi enkripsi dengan algoritma *cipher* blok
D = fungsi dekripsi dengan algoritma *cipher* blok
K = kunci
m = panjang blok enkripsi/dekripsi
n = panjang unit enkripsi/dekripsi

menjadi *n*-bit pertama dari cipherteks. Salinan (*copy*) *n*-bit dari cipherteks ini dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan semua *m-n* bit lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan.

3. *m-n* bit plainteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.

Sedangkan, algoritma dekripsi dengan mode *CFB* adalah sebagai berikut:

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Dekripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil dekripsi berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari cipherteks menjadi *n*-bit pertama dari plainteks. Salinan (*copy*) *n*-bit dari cipherteks dimasukkan ke dalam antrian (menempati *n* posisi bit paling kanan antrian), dan semua *m-n* lainnya di dalam antrian digeser ke kiri menggantikan *n* bit pertama yang sudah digunakan.
3. *m-n* bit cipherteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.



Gambar 5. Mode *CFB* n-bit

Baik enkripsi maupun dekripsi, algoritma *E* dan *D* yang digunakan sama. Mode *CFB n*-bit yang bekerja pada blok berukuran *m*-bit dapat dilihat pada Gambar 4.

Secara formal, mode *CFB n*-bit dapat dinyatakan sebagai:

Proses Enkripsi:

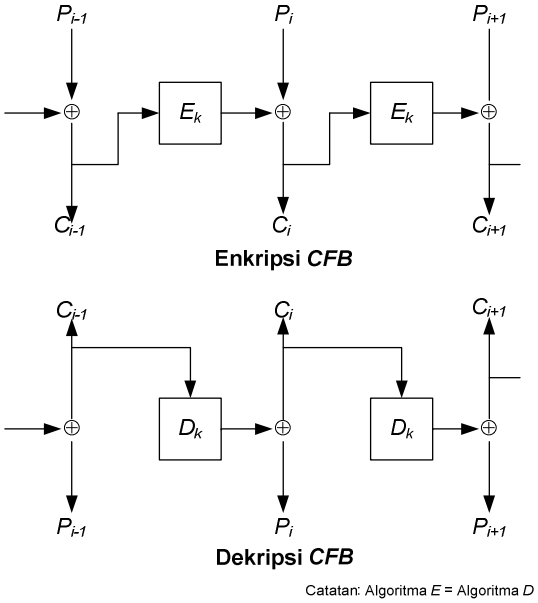
$$C_i = P_i \oplus MSB_m(E_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \parallel C_i$$

Proses Dekripsi:

$$P_i = C_i \oplus MSB_m(D_k(X_i))$$
$$X_{i+1} = LSB_{m-n}(X_i) \parallel C_i$$

yang dalam hal ini:

X_i = isi antrian dengan X_i adalah *IV*
E = fungsi enkripsi dengan algoritma *cipher* blok
D = fungsi dekripsi dengan algoritma *cipher* blok
K = kunci
m = panjang blok enkripsi/dekripsi
n = panjang unit enkripsi/dekripsi
$\parallel$ = operator penyambungan (*concatenation*)
MSB = Most Significant Byte
LSB = Least Significant Byte



Gambar 6. Enkripsi dan Dekripsi Mode *CFB n*-bit untuk blok *n*-bit

Jika *m* = *n*, maka mode *CFB n*-bit adalah seperti pada Gambar 5. *CFB* menggunakan skema umpan balik dengan mengaitkan blok plainteks bersama-sama sedemikian sehingga cipherteks bergantung pada semua blok plainteks sebelumnya. Skema enkripsi dan dekripsi dengan mode *CFB* dapat dilihat pada Gambar 5.
Dari Gambar 5 dapat dilihat bahwa:

$$C_i = P_i \oplus E_k(C_{i-1})$$
$$P_i = C_i \oplus D_k(C_{i-1})$$

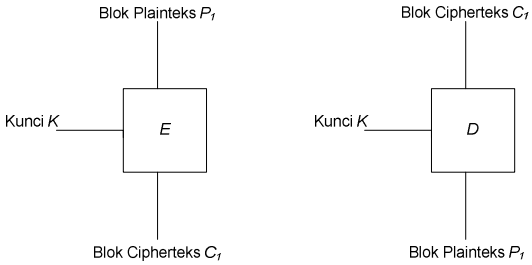
IV pada *CFB* tidak perlu dirahasiakan. *IV* harus unik untuk setiap pesan, sebab *IV* yang sama untuk setiap pesan yang berbeda akan menghasilkan *keystream* k_i yang sama.

$$C_i = E_k(P_i)$$

dan dekripsi sebagai

$$P_i = D_k(C_i)$$

yang dalam hal ini, P_i dan C_i masing-masing blok plainteks dan cipherteks ke-*i*. Skema enkripsi dan dekripsi dengan mode *ECB* dapat dilihat pada Gambar 2.



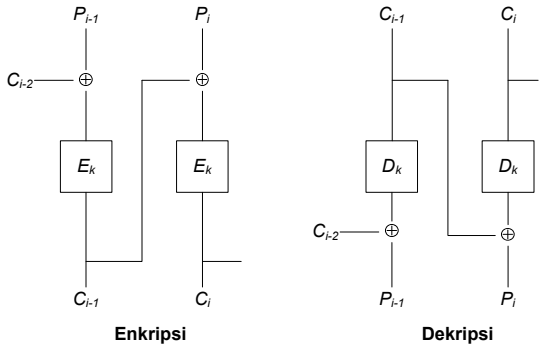
Gambar 3. Skema Enkripsi dan Dekripsi dengan Mode *ECB*

Ada kemungkinan panjang plainteks tidak habis dibagi dengan panjang ukuran blok yang ditetapkan. Hal ini mengakibatkan blok terakhir berukuran lebih pendek daripada blok-blok lainnya. Satu cara untuk mengatasi hal ini adalah dengan *padding*, yaitu menambahkan blok terakhir dengan pola bit yang teratur agar panjangnya sama dengan ukuran blok yang ditetapkan.

Cipher Block Chaining (*CBC*)

Mode ini menerapkan mekanisme umpan balik (*feedback*) pada sebuah blok, yang dalam hal ini hasil enkripsi blok sebelumnya diumpanbalikkan ke dalam enkripsi blok yang *current*. Caranya, blok plainteks yang *current* di-*XOR*-kan terlebih dahulu dengan blok cipherteks hasil enkripsi sebelumnya, selanjutnya hasil peng-*XOR*-an ini masuk ke dalam fungsi enkripsi. Dengan mode *CBC*, setiap blok cipherteks bergantung tidak hanya pada blok plainteksnya tetapi juga pada seluruh blok plainteks sebelumnya.

Dekripsi dilakukan dengan memasukkan blok cipherteks yang *current* ke fungsi dekripsi, kemudia meng-*XOR*-kan hasilnya dengan blok cipherteks sebelumnya. Dalam hal ini, blok cipherteks sebelumnya berfungsi sebagai umpan maju (*feedforward*) pada akhir proses dekripsi. Skema enkripsi dan dekripsi dengan mode *CBC* dapat dilihat pada Gambar 3.



Gambar 4. Enkripsi dan Dekripsi dengan Mode *CBC*

Secara matematis, enkripsi dengan mode *CBC* dinyatakan sebagai

$$C_i = E_k(P_1 \oplus C_{i1})$$

dan dekripsi sebagai

$$P_i = D_k(C_i) \oplus C_{i-1}$$

Yang dalam hal ini, C_0 = *IV* (*initialization vector*). *IV* dapat diberikan oleh pengguna atau dibangkitkan secara acak oleh program. Jadi, untuk menghasilkan blok cipherteks pertama (C_1), *IV* digunakan untuk menggantikan blok cipherteks sebelumnya, C_0. Sebaliknya pada dekripsi, blok plainteks diperoleh dengan cara meng-*XOR*-kan *IV* dengan hasil dekripsi terhadap blok cipherteks pertama.

Pada mode *CBC*, blok plainteks yang sama menghasilkan blok cipherteks yang berbeda hanya jika blok-blok plainteks sebelumnya berbeda.

Cipher-Feedback (*CFB*)

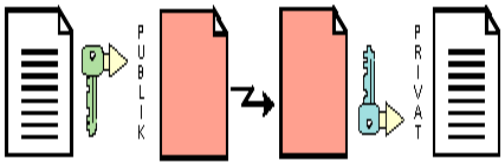
Pada mode *CFB*, data dienkripsikan dalam unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan dapat berupa bit per bit, 2 bit, 3 bit, dan seterusnya. Bila unit yang dienkripsikan satu karakter setiap kalinya, maka mode *CFB*-nya disebut *CFB* 8-bit. Secara umum *CFB* *n*-bit mengenkripsi plainteks sebanyak *n* bit setiap kalinya, yang mana $n \leq m$ (m = ukuran blok). Mode *CFB* membutuhkan sebuah antrian (*queue*) yang berukuran sama dengan ukuran blok masukan.

Tinjau mode *CFB* *n*-bit yang bekerja pada blok berukuran *m*-bit. Algoritma enkripsi dengan mode *CFB* adalah sebagai berikut:

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Enkripsikan antrian dengan kunci *K*. *n* bit paling kiri dari hasil enkripsi berlaku sebagai *keystream* (k_i) yang kemudian di-*XOR*-kan dengan *n*-bit dari plainteks

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.

Dengan cara seperti ini, jika Anto mengirim pesan untuk Badu, Anto dapat merasa yakin bahwa pesan tersebut hanya dapat dibaca oleh Badu, karena hanya Badu yang bisa melakukan dekripsi dengan kunci privatnya. Tentunya Anto harus memiliki kunci publik Badu untuk melakukan enkripsi. Anto bisa mendapatkannya dari Badu, ataupun dari pihak ketiga seperti Tari.



Gambar 2. Penggunaan kunci asimetris

Teknik enkripsi asimetris ini jauh lebih lambat ketimbang 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).

Berdasarkan arah implementasi dan pembabakan jamannya dibedakan menjadi :

- algoritma sandi klasik classic cryptography
- algoritma sandi modern modern cryptography

Berdasarkan kerahasiaan kuncinya dibedakan menjadi :

- algoritma sandi kunci rahasia secret-key
- algoritma sandi kunci publik publik-key

Pada skema kunci-simetris, digunakan sebuah kunci rahasia yang sama untuk melakukan proses enkripsi dan dekripsinya. Sedangkan pada sistem kunci-asimentris digunakan sepasang kunci yang berbeda, umumnya disebut kunci publik(public key) dan kunci pribadi (private key), digunakan untuk proses enkripsi dan proses dekripsinya. Bila elemen teks terang dienkripsi dengan menggunakan kunci pribadi maka elemen teks sandi yang dihasilkannya hanya bisa didekripsikan dengan menggunakan pasangan kunci pribadinya. Begitu juga sebaliknya, jika kunci pribadi digunakan untuk proses enkripsi maka proses dekripsi harus menggunakan kunci publik pasangannya.

algoritma sandi kunci-simetris

Skema algoritma sandi akan disebut kunci-simetris apabila untuk setiap proses enkripsi maupun dekripsi data secara keseluruhan digunakan kunci yang sama. Skema ini berdasarkan jumlah data per proses dan alur pengolahan data didalamnya dibedakan menjadi dua kelas, yaitu block-cipher dan stream-cipher.

Block-Cipher

Block-cipher adalah skema algoritma sandi yang akan membagi-bagi teks terang yang akan dikirimkan dengan ukuran tertentu (disebut blok) dengan panjang t, dan setiap blok dienkripsi dengan menggunakan kunci yang sama. Pada umumnya, block-cipher memproses teks terang dengan blok yang relatif panjang lebih dari 64 bit, untuk mempersulit penggunaan pola-pola serangan yang ada untuk membongkar kunci. Untuk menambah kehandalan model algoritma sandi ini, dikembangkan pula beberapa tipe proses enkripsi, yaitu :

- ECB, Electronic Code Book
- CBC, Cipher Block Chaining
- OFB, Output Feed Back
- CFB, Cipher Feed Back

***Electronic Code Book* (*ECB*)**

Pada mode ini, setiap blok plainteks P_i dienkripsi secara individual dan independen menjadi blok cipherteks C_i. Secara matematis, enkripsi dengan mode *ECB* dinyatakan sebagai

penyerang bahkan dapat memilih penggalan mana dari pesan asli yang akan disandikan.

Berdasarkan bagaimana cara dan posisi seseorang mendapatkan pesan-pesan dalam saluran komunikasi, penyerangan dapat dikategorikan menjadi:

1. *Sniffing*: secara harafiah berarti mengendus, tentunya dalam hal ini yang diendus adalah pesan (baik yang belum ataupun sudah dienkripsi) dalam suatu saluran komunikasi. Hal ini umum terjadi pada saluran publik yang tidak aman. Sang pengendus dapat merekam pembicaraan yang terjadi.
2. *Replay attack* [DHMM 96]: Jika seseorang bisa merekam pesan-pesan *handshake* (persiapan komunikasi), ia mungkin dapat mengulang pesan-pesan yang telah direkamnya untuk menipu salah satu pihak.
3. *Spoofing* [DHMM 96]: Penyerang – misalnya Maman – bisa menyamar menjadi Anto. Semua orang dibuat percaya bahwa Maman adalah Anto. Penyerang berusaha meyakinkan pihak-pihak lain bahwa tak ada salah dengan komunikasi yang dilakukan, padahal komunikasi itu dilakukan dengan sang penipu/penyerang. Contohnya jika orang memasukkan PIN ke dalam mesin ATM palsu – yang benar-benar dibuat seperti ATM asli – tentu sang penipu bisa mendapatkan PIN-nya dan copy pita magentik kartu ATM milik sang nasabah. Pihak bank tidak tahu bahwa telah terjadi kejahatan.
4. *Man-in-the-middle* [Schn 96]: Jika *spoofing* terkadang hanya menipu satu pihak, maka dalam skenario ini, saat Anto hendak berkomunikasi dengan Badu, Maman di mata Anto seolah-olah adalah Badu, dan Maman dapat pula menipu Badu sehingga Maman seolah-olah adalah Anto. Maman dapat berkuasa penuh atas jalur komunikas ini, dan bisa membuat berita fitnah.

Kabel koaksial yang sering dipergunakan pada jaringan sangat rentan terhadap serangan *vampire tap* [Tane 89], yakni perangkat keras sederhana yang bisa menembus bagian dalam kabel koaksial sehingga dapat mengambil data yang mengalir tanpa perlu memutuskan komunikasi data yang sedang berjalan. Seseorang dengan *vampire tap* dan komputer jinjing dapat melakukan serangan pada bagian apa saja dari kabel koaksial.

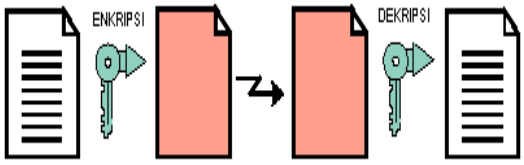
Penyerang juga bisa mendapatkan kunci dengan cara yang lebih tradisional, yakni dengan melakukan penyiksaan, pemerasan, ancaman, atau bisa juga dengan menyogok seseorang yang memiliki kunci itu. Ini adalah cara yang paling ampuh untuk mendapat kunci.

Secara umum berdasarkan kesamaan kuncinya, algoritma sandi dibedakan menjadi :

- kunci-simetris/symetric-key, sering disebut juga algoritma sandi konvensional karena umumnya diterapkan pada algoritma sandi klasik
- kunci-asimetris/asymetric-key

Kunci Simetris

Ini adalah jenis kriptografi yang paling umum dipergunakan. Kunci untuk membuat pesan yang disandikan sama dengan kunci untuk membuka pesan yang disandikan itu. Jadi pembuat pesan dan penerimanya harus memiliki kunci yang sama persis. Siapapun yang memiliki kunci tersebut – termasuk pihak-pihak yang tidak diinginkan – dapat membuat dan membongkar rahasia *ciphertext*. Problem yang paling jelas disini terkadang bukanlah masalah pengiriman *ciphertext*-nya, melainkan masalah bagaimana menyampaikan kunci simetris tersebut kepada pihak yang diinginkan. Contoh algoritma kunci simetris yang terkenal adalah DES (*Data Encryption Standard*) dan RC-4.



Gambar 1. Kunci simetris

Kunci Asimetris

Pada pertengahan tahun 70-an Whitfield Diffie dan Martin Hellman menemukan teknik enkripsi asimetris yang merevolusi dunia kriptografi.

adalah algoritma sandi yang kekuatannya terletak pada kunci, bukan pada kerahasiaan algoritma itu sendiri. Teknik dan metode untuk menguji kehandalan algoritma sandi adalah kriptanalisa.

Dasar matematis yang mendasari proses enkripsi dan dekripsi adalah relasi antara dua himpunan yaitu yang berisi elemen teks terang /plaintext dan yang berisi elemen teks sandi/ciphertext. Enkripsi dan dekripsi merupakan fungsi transformasi antara himpunan-himpunan tersebut. 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*).

Kode & nama	Penjelasan
A: Anto	Pihak pertama
B: Badu	Pihak kedua
C: Chandra	Pihak ketiga
E: Edi	Pihak penyadap informasi yang tidak diperuntukkan kepadanya (*eavesdropper*)
M: Maman	Pihak yang tidak hanya menyadap informasi, namun juga mengubah informasi yang disadap (*malacious person*)
T: Tari, Tata, Tania	Pihak yang dipercaya oleh pihak pertama, kedua dan ketiga (*trusted person*)

$$C = E(M)$$

dimana

M = pesan asli
E = proses enkripsi
C = pesan dalam bahasa sandi (untuk ringkasnya disebut sandi)

Sedangkan dekripsi adalah proses mengubah pesan dalam suatu bahasa sandi menjadi pesan asli kembali.

$$M = D(C)$$

D = proses dekripsi

Umumnya, selain menggunakan fungsi tertentu dalam melakukan enkripsi dan dekripsi, seringkali fungsi itu diberi parameter tambahan yang disebut dengan istilah kunci.

Untuk memudahkan penggambaran suatu skenario komunikasi dalam pembahasan selanjutnya, maka dipergunakan nama-nama orang yang relevan dengan peran yang dilakukannya dalam komunikasi itu.

Tabel 1 Nama-nama ganti untuk mempermudah penjelasan

Sekarang akan diuraikan mengenai beberapa jenis serangan, jenis-jenis kunci kriptografi, berbagai jenis perangkat dan protokol kriptografi, serta masalah panjang kunci kriptografi.

Jenis Serangan

Selain ada pihak yang ingin menjaga agar pesan tetap aman, ada juga ternyata pihak-pihak yang ingin mengetahui pesan rahasia tersebut secara tidak sah. Bahkan ada pihak-pihak yang ingin agar dapat mengubah isi pesan tersebut. Ilmu untuk mendapatkan pesan yang asli dari pesan yang telah disandikan tanpa memiliki kunci untuk membuka pesan rahasia tersebut disebut kriptoanalisis. Sedangkan usaha untuk membongkar suatu pesan sandi tanpa mendapatkan kunci dengan cara yang sah dikenal dengan istilah serangan (*attack*).

Di bawah ini dijelaskan beberapa macam penyerangan terhadap pesan yang sudah dienkripsi:

1. *Ciphertext only attack*, penyerang hanya mendapatkan pesan yang sudah tersandikan saja.
2. *Known plaintext attack*, dimana penyerang selain mendapatkan sandi, juga mendapatkan pesan asli. Terkadang disebut pula *clear-text attack*.
3. *Choosen plaintext attack*, sama dengan *known plaintext attack*, namun

Studi Mengenai Kriptografi dan Masalah-Masalah Serta Penyerangan yang Sering Terjadi Terhadap Metode Kriptografi

Andika Pratama – NIM : 13505048
Sekolah Teknik Elektro & Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
E-mail : squall_pratama@yahoo.com

Abstraksi

Makalah berisi tentang mengenai salah satu sistem enkripsi yaitu kriptografi yang sering digunakan pada zaman perang dunia untuk menyamarkan pesan-pesan rahasia yang dikirim menjadi kode-kode yang kalaupun berhasil disadap oleh musuh, pesan yang terkirim tidak akan membahayakan pihak pengirim.

Namun seringkali kita temui penyerangan-penyerangan yang terjadi pada bahasa yang telah dienkripsi tersebut. Makalah ini ditujukan untuk menerangkan kepada para pembaca mengenai contoh-contoh penyerangan pada metode kriptografi dan beberapa cara penanggulangannya agar pembaca dapat lebih awas dan waspada terhadap bahaya-bahaya yang mengintai di sekitar kita.

Pendahuluan

Kriptografi, secara umum adalah ilmu dan seni untuk menjaga kerahasiaan berita. Selain pengertian tersebut terdapat pula pengertian ilmu yang mempelajari teknik-teknik matematika yang berhubungan dengan aspek keamanan informasi seperti kerahasiaan data, keabsahan data, integritas data, serta autentikasi data. Tidak semua aspek keamanan informasi dapat ditangani oleh kriptografi.

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/mengupas informasi yang telah disandi.
- Integritas data, adalah 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 pensubsitusian data lain kedalam data yang sebenarnya.
- Autentikasi, adalah 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.
- Non-repudiasi., atau nirpenyangkalan adalah usaha untuk mencegah terjadinya penyangkalan terhadap pengiriman/terciptanya suatu informasi oleh yang mengirimkan/membuat.

Algoritma Sandi

algoritma sandi adalah algoritma yang berfungsi untuk melakukan tujuan kriptografis. Algoritma tersebut harus memiliki kekuatan untuk melakukan (dikemukakan oleh Shannon):

- konfusi/pembingungan (confusion), dari teks terang sehingga sulit untuk direkonstruksikan secara langsung tanpa menggunakan algoritma dekripsinya
- difusi/peleburan (difusion), dari teks terang sehingga karakteristik dari teks terang tersebut hilang.

sehingga dapat digunakan untuk mengamankan informasi. Pada implementasinya sebuah algoritmas sandi harus memperhatikan kualitas layanan/Quality of Service atau QoS dari keseluruhan sistem dimana dia diimplementasikan. Algoritma sandi yang handal