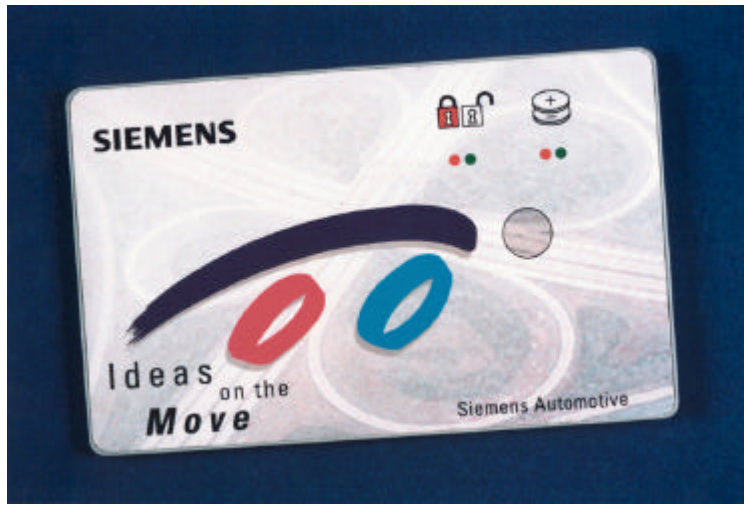
14

Kriptografi dalam Kehidupan Sehari-hari

Kehidupan kita saat ini dikelililingi oleh kriptografi. Kriptografi sudah digunakan dalam berbagai aplikasi, mulai dari penarikan uang di ATM, penggunaan kartu kredit, penggunaan kartu cerdas, percakapan dengan telepon genggam, *password* komputer, televisi, transaksi *e-commerce* di internet, gedung-gedung bisnis, sampai pada pengaktifan peluru kendali dan bom nuklir. Bab ini membahas secara ringkas penerapan kriptografi dalam kehidupan sehari-hari.

14.1 Kartu Cerdas

Salah satu aplikasi *PKI* yang tumbuh sangat pesat adalah kartu cerdas (*smart card*). Kartus cerdas yang mirip dengan kartu kredit dapat melayani banyak fungsi, mulai dari otentikasi sampai penyimpanan data (Gambar 14.1). Dengan menggunakan kartu cerdas, pengguna dapat mengakses informasi dari berbagai peralatan dengan kartu cerdas yang sama



Smart Card
The smart card completely replaces keys for functions like door locking, ignition switch, immobilization and exterior locks. In addition to these security features, it can performs tasks such as personalized seat, mirror and cilmate adjustments.

Gambar 4.1 Sebuah kartu cerdas dari Siemens

Kartu cerdas yang paling populer adalah *memory card* dan *microprocessor card*. *Memory card* mirip dengan *floppy disk*, sedangkan *microprocessor card* mirip dengan komputer kecil dengan sistem operasi, sekuriti, dan penyimpanan data. Kartu cerdas mempunyai beberapa jenis antarmuka (*interface*) yang berbeda. Jenis antarmuka yang umum adalah *contact interface*, yang dalam hal ini kartu cerdas dimasukkan ke dalam alat pembaca (*card reader*) dan secara fisik terjadi kontak fisik antara alat dan kartu (Gambar 14.2).

Gambar 14.2 Pembaca kartu cerdas

Kartu cerdas menyimpan kunci privat, sertifikat digital, dan informasi lainnya untuk mengimplementaiskan *PKI.* Kartu cerdas juga menyimpan nomor kartu kredit dan informasi kontak personal (no telpon). Sertifikat digital ditandatangani oleh *card issuer* (*CA*) untuk mensertifikasi kunci publik pemilik kartu.

Penggunaan kartu cerdas dikombinasikan dengan *PIN* (*Personal Identification Number*). Jadi, ada dua level yang harus dari penggunaan kartu cerdas, yaitu memiliki kartu cerdas itu sendiri dan mengetahui *PIN* yang mengakses informasi yang disimpan di dalam kartu. Komputer *server* mengotentikasi kartu dengan cara mengirimkan suatu nilai atau *string* (yang disebut *challenge*) ke kartu untuk ditandatangani dengan menggunakan kunci privat (yang tersimpan di dalam kartu), lalu tanda-tangan tersebut diverifikasi oleh mesin dengan menggunakan kunci publik pemilik kartu. Komputer *server* perlu menyimpan kunci publik *card issuer* untuk memvalidasi sertifikat digital.

Banyak peralatan *mobile* yang menggunakan kartu cerdas untuk otentikasi. Namun kartu cerdas masih tidak menjamin keamanan secara total. Jika peralatan *mobile* hilang atau dicuri, sertifikat digital dan kunci privat di dalam kartu cerdas (yang terdapat di dalam peralatan tersebut) berpotensi diakses oleh pencuri untuk mengakses informasi rahasia. Telpon seluler dengan teknologi *GSM* memiliki kartu cerdas yang terintegrasi di dalam *handphone*. Pemilik *handphone* memiliki opsi untuk men-set *PIN* untuk proteksi tambahan, sehingga jika *handphone* hilang atau dicuri, *handphone* tidak dapat digunakan tanpa mengetahui *PIN* tersebut.

Kartu cerdas *Wireless Identity Module* (*WIM*) termasuk di dalam *Wireless Application Protocol* (*WAP*). Kartu *WIM* memproteksi komunikasi dan transaksi *mobile* dengan tanda-tangan digital. Kartu *WIM* menyediakan keamanan untuk sertifikat digital, manajemen kode *PIN*, kunci, dan tanda-tangan digital. *WIM* menyimpan algoritma enkripsi yang diperlukan di daam kartu cerdas. Semua fungsi yang diperlukan untuk sistem *PKI* dimasukkan ke dalam karu cerdas.

14.2 Transaksi lewat Anjungan Tunai mandiri (ATM)

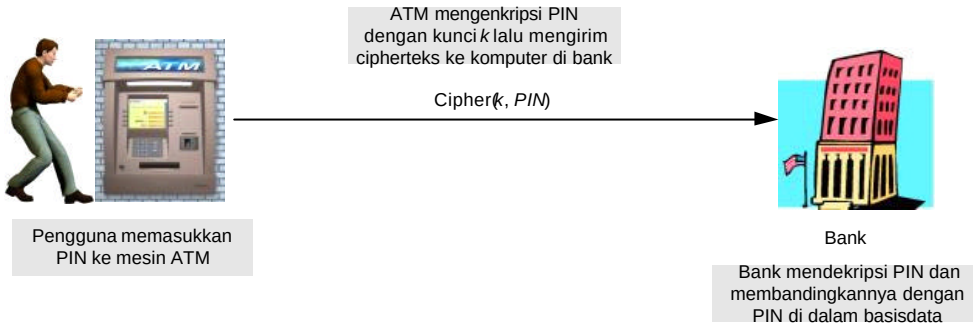
Anjungan Tunai Mandiri atau *Automatic Teller Machine* (*ATM*) digunakan nasabah bank untuk melakukan transaski perbankan. Utamanya, kegunaan *ATM* adalah untuk menarik uang secara tunai (*cash withdrawal*), namun saat ini *ATM* juga digunakan untuk transfer uang (pemindahbukuan), mengecek saldo, membayar tagihan kartu ponsel, membeli tiket kereta api, dan sebagainya.

Transaksi lewat *ATM* memerlukan kartu magnetik (disebut juga kartu *ATM*) yang terbuat dari plastik dan kode *PIN* (*Personal Information Number*) yang berasosiasi dengan kartu tersebut. *PIN* terdiri dari 4 angka yang harus dijaga kerahasiannya oleh pemilik kartu *ATM*, sebab orang lain yang mengetahui *PIN* dapat menggunakan kartu *ATM* yang dicuri atau hilang untuk melakukan penarikan uang.

PIN digunakan untuk memverifikasi kartu yang dimasukkan oleh nasabah di *ATM*. Proses verifikasi dilakukan di komputer pusat (*host*) bank, oleh karena itu harus ada komunikasi dua arah antara *ATM* dan komputer *host*. *ATM* mengirim *PIN* dan informasi tambahan pada

kartu ke komputer *host*, *host* melakukan verifikasi dengan cara membandingkan *PIN* yang di-*entry*-kan oleh nasabah dengan *PIN* yang disimpan di dalam basisdata komputer *host*, lalu mengirimkan pesan tanggapan ke *ATM* yang menyatakan apakah transaksi dapat dilanjutkan atau ditolak.

Selama transmisi dari *ATM* ke komputer *host*, *PIN* harus dilindungi dari penyadapan oleh orang yang tidak berhak. Bentuk perlindungan yang dilakukan selama transmisi adalah dengan mengenkripsikan *PIN*. Di sisi bank, *PIN* yang disimpan di dalam basisdata juga dienkripsi (lihat Gambar 14.3).



Gambar 14.3 Mekanisme enkripsi dan dekripsi *PIN* pada transaksi dengan mesin ATM

Algoritma enkripsi yang digunakan adalah *DES* dengan mode *ECB*. Karena *DES* bekerja dengan mengenkripsikan blok 64-bit, maka *PIN* yang hanya terdiri dari 4 angka (32 bit) harus ditambah dengan *padding bits* sehingga panjangnya menjadi 64 bit. *Padding bits* yang ditambahkan berbeda-beda untuk setiap *PIN*, bergantung pada informasi tambahan pada setiap kartu *ATM*-nya [PIN02].

Karena panjang *PIN* hanya 4 angka, maka peluang ditebak sangat besar. Seseorang yang memperoleh kartu *ATM* curian atau hilang dapat mencoba semua kemungkinan kode *PIN* yang mungkin, sebab hanya ada $10 \times 10 \times 10 \times 10 = 10.000$ kemungkinan kode *PIN* 4-angka. Untuk mengatasi masalah ini, maka kebanyakan *ATM* hanya membolehkan peng-*entry*-an *PIN* maksimum 3 kali, jika 3 kali tetap salah maka *ATM* akan 'menelan' kartu *ATM*. Masalah ini juga menunjukkan bahwa kriptografi tidak selalu dapat menyelesaikan masalah keamanan data.

Beberapa jaringan *ATM* sekarang menggunakan kartu cerdas sehingga memungkinkan penggunaan kriptografi kunci publik. Kartu *ATM* pengguna mengandung kunci privat dan sertifikat digital yang ditandatangani oleh *card issuer* (*CA*) untuk mensertifikasi kunci publiknya. *ATM* mengotentikasi kartu dengan cara mengirimkan suatu *string* ke kartu untuk ditandatangani dengan menggunakan kunci privat, lalu tanda-tangan tersebut diverifikasi oleh *ATM* dengan menggunakan kunci publik pemilik kartu.

Seperti semua sistem yang berbasis sertifikat digital, terminal *ATM* perlu memiliki salinan kunci publik *card issuer* dengan maksud untuk memvalidasi sertifikat digital. Hal ini direalisasikan dengan menginstalasi kunci publik tersebut ke dalam mesin *ATM*.

14.3 *Pay TV*

Pay TV adalah siaran TV yang hanya dapat dinikmati oleh pelanggan yang membayar saja, sedangkan pemilik TV yang tidak berlangganan tidak dapat menikmati siarannya. Kebanyakan *Pay TV* menyiarkan acara olahraga dan hiburan (Gambar 14.4). Siaran *Pay TV* dipancarkan secara *broadcast*, namun hanya sejumlah pesawat TV yang berhasil menangkap siaran tersebut yang dapat 'mengerti' isinya. Pada sistem *Pay TV*, sinyal *broadcast* dienkripsi dengan kunci yang unik. Orang-orang yang berlangganan *Pay* TV pada dasarnya membayar untuk mengetahui kunci tersebut.

Gambar 14.4 Contoh iklan acara *Pay TV*

Bagaimana mengetahui bahwa kunci tersebut dimiliki oleh pelanggan yang sah, dan bukan orang yang mengetahui kunci tersebut dari pelanggan lainnya? Solusi yang umum adalah setiap pelanggan diberikan kartu cerdas (*smart card*) yang mengandung kunci privat (*private key*) yang unik dalam konteks algoritma kriptografi kunci-publik. Kartu cerdas dimasukkan ke dalam *card reader* yang dipasang pada pesawat TV. Selanjutnya, pelanggan *Pay* TV dikirimi kunci simetri yang digunakan untuk mengenkripsi siaran. Kunci simetri ini dikirim dalam bentuk terenkripsi dengan menggunakan kunci publik pelanggan. *Smart card* kemudian mendekripsi kunci simetri ini dengan kunci privat pelanggan. Selanjutnya, kunci simetri digunakan untuk mendekripsi siaran TV.

14.4 Komunikasi dengan Telepon Seluler

Penggunaan telepon seluler (ponsel) atau lebih dieknal dengan nama telepon genggam (*handphone*) yang bersifat *mobile* memungkinkan orang berkomunikasi dari tempat mana saja. Telepon seluler bersifat nirkabel (*wireless*), sehingga pesan yang dikirim dari ponsel ditransmisikan melalui gelombang mikro (*microwave*) atau radio sampai ia mencapai *base station* (*BST*) terdekat, selanjutnya ditransfer ke ponsel penerim. *GSM* merupakan teknologi telepon seluler yang paling banyak digunakan di seluruh dunia.

Karena menyadap sinyal radio jauh lebih mudah daripada menyadap sinyal pada saluran kabel, maka ini berarti *GSM* tidak lebih aman daripada telepon *fixed* konvensional. Untuk membuat komunikasi lewat ponsel aman, maka pesan dienkripsi selama transmisi dari ponsel ke *BST* terdekat. Metode enkripsi yang digunakan adalah metode *cipher* aliran (*stream cipher*).

Masalah keamanan lain adalah identitas penelpon. Operator seluler harus dapat mengidentifikasi suatu panggilan (*call*) dan mengetahui identitas penelpon (apakah penelpon merupakan pengguna/pelanggan dari operator seluler tersebut atau pengguna/ pelanggan dari operator lain).

Jadi, pada *GSM* diperlukan dua kebutuhan keamanan lainnya, yaitu:

1. otentikasi penelpon (*user authentication*), yang merupakan kebutuhan bagi sistem,
2. kerahasiaan (*confidentiality*) pesan (data atau suara), yang merupakan kebutuhan bagi pelanggan,

Dua kebutuhan ini dipenuhi dengan penggunaan kartu cerdas (*smart card*) personal yang disebut kartu *SIM* (*Subscriber Identity Module card*). Kartu *SIM* berisi:

1. identitas pelanggan/pengguna operator seluler berupa *IMSI* (*International Mobile Subscriber Identity*) yang unik nilainya,
2. kunci otentikasi rahasia sepanjang 128-bit yang diketahui hanya oleh operator. Nilai ini digunakan sebagai kunci pada protokol otentikasi dengan menggunakan program enkripsi yang dipilih oleh operator (algoritma *A2*, *A3*, atau *A5*),
3. *PIN* (jika di-set oleh pengguna).
4. Program enkripsi.

Secara keseluruhan, sistem keamanan *GSM* terdiri atas dalam 3 komponen, yaitu:

1. kartu *SIM*,
2. *handset* (pesawat telepon seluler),
3. jaringan *GSM* (seperti jaringan *ProXL*, *Simpati*, *IM3*). Setiap jaringan dioperasikan oleh operatornya masing-masing (*Excelcomindo*, *Telkomsel*, *Satelindo*). Komputer operator (*host*) memiliki basisdata yang berisi identitas (*IMSI*) dan kunci otentikasi rahasia semua pelanggan/pengguna *GSM*.

Otentikasi Penelpon

Otentikasi penelpon dilakukan melalui protokol otentikasi dengan mekanisme *challenge – response*. Ketika pengguna ponsel melakukan panggilan (*call*), identitasnya dikirim ke komputer operator via *BST* untuk keperluan otentikasi. Karena *BST* tidak mengetahui kunci otentikasi kartu *SIM*, dan bahkan tidak mengetahui algoritma otentikasi, maka komputer operator melakukan verifikasi pengguna dengan cara mengirimkan suatu nilai acak (128 bit) yang disebut *challenge* ke *SIM card* penelpon. Kartu *SIM* mengeluarkan *response* dengan cara mengenkripsi *challenge* 128-bit tersebut dengan menggunakan kunci otentikasi yang terdapat di dalam kartu.

Enkripsi terhadap *challenge* menghasilkan keluaran 128-bit; dari 128-bit keluaran ini hanya 32 bit yang dikirim dari kartu *SIM* ke *BST* sebagai *response*. *BST* meneruskan *response* ke komputer operator. Ketika *response* sampai di komputer operator, komputer operator melakukan perhitungan yang sama dengan yang dilakukan oleh kartu *SIM*; yang dalam hal ini komputer mengenkripsi *challenge* yang dikirim tadi dengan menggunakan kunci otentikasi penelpon (ingat, komputer operator mengetahui kunci otentikasi semua kartu *SIM*), lalu membandingkan hasil enkripsi ini (yang diambil hanya 32 bit) dengan *response* yang ia terima. Jika sama, maka otentikasi berhasil, dan penelpon dapat melakukan percakapan.

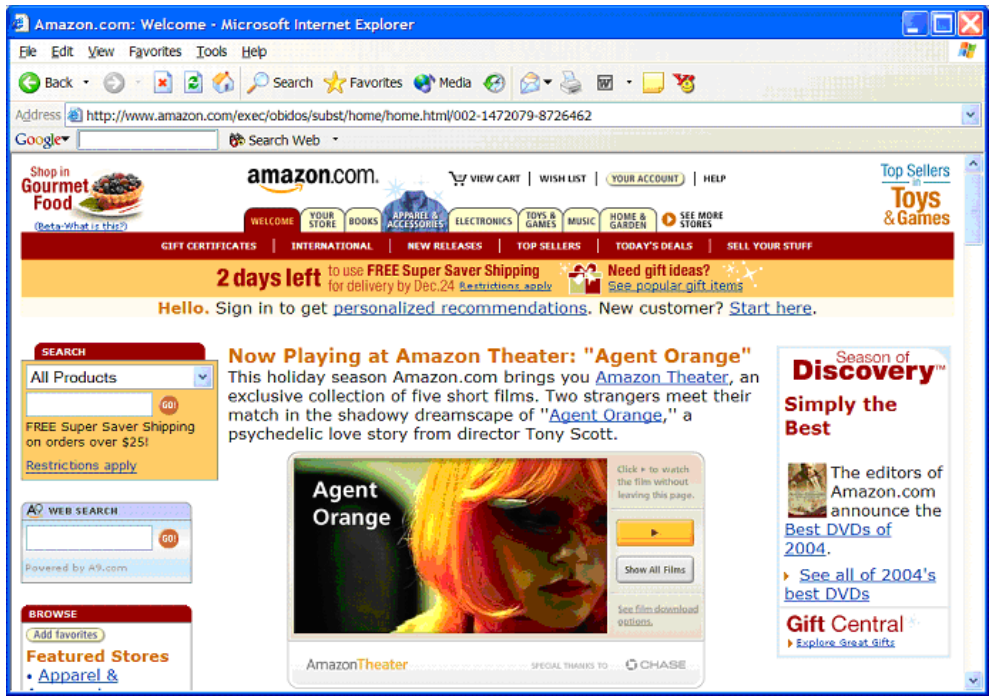
Sebagaimana dijelaskan di atas, dari 128-bit hasil enkripsi, hanya 32 bit yang dikirim sebagai *response*. Jadi, masih ada 96 bit sisanya yang hanya diketahui oleh kartu *SIM*, *BST*, dan komputer operator.

Kerahasiaan Pesan

SIM card juga berisi program *stream cipher* (algoritma *A5*) untuk mengenkripsi pesan dari ponsel ke *BST*. Kunci enkripsi panjangnya 64 bit, yang diambil dari 96 bit sisa dari *response SIM card*. Perhatikan bahwa kunci enkripsi 64-bit ini berbeda setiap kali proses otentikasi dilakukan (mengapa?). Hal ini memenuhi prinsip algoritma *OTP* (*one-time pad*).

14.5 *E-commerce* di Internet dan *SSL*

E-commerce adalah transaksi barang dan jasa dengan menggunakan internet sebagai medianya. Di dalam dunia *e-commerce* ada pedagang (*merchant*) dan pembeli (*customer*). Padagang menawarkan barang/jasa melalui situs *web* yang dapat diakses oleh pembeli dari lokasi manapun di muka bumi. Dalam hal ini, situs *web* pedagang disebut *server* sedangkan pembeli disebut *client*. Pembeli mengakses *web* dengan program *browser* seperrti *Internet Explorer*. Situs *web Amazon.com* merupakan situs *e-commerce* yang terkenal (Gambar 14.5.



Gambar 14.5 Situs *amazon.com*

Pembayaran barang umunya dilakukan dengan menggunakan kartu kredit, yang berarti bahwa pembeli harus mengirimkan nomor kartu kredit dan informasi lainnya melalui internet. Karena alasan keamanan yang menyangkut informasi kartu kredit maka transaksi barang lewat internet tidak terlalu populer. Banyak orang yang masih beranggapan *e-commerce* tidak aman; kekhawatiran yang wajar, namun masalah ini sebnarnya sudah dipikirkan solusinya. *Browsing web* secara aman adalah fitur paling penting pada *e-commerce*.

Secure Socket Layer (*SSL*)

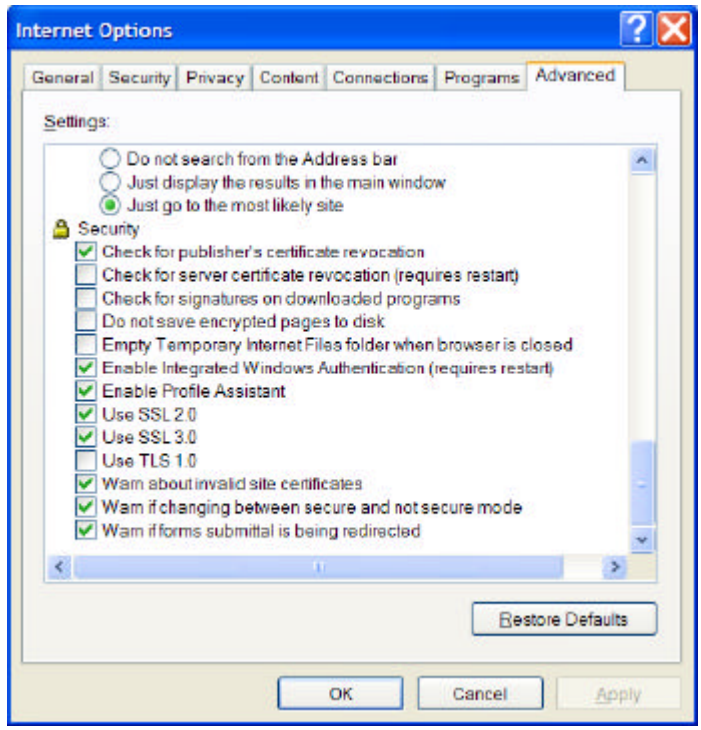
Secure Socket Layer (*SSL*) adalah protokol yang digunakan untuk *browsing web* secara aman. Dalam hal ini, *SSL* bertindak sebagai protokol yang mengamankan komunikasi antara *client* dan *server*. Protokol ini memfasilitasi penggunaan enkripsi untuk data yang rahasia dan membantu menjamin integritas informasi yang dipertukarkan antara *website* dan *web browser*.

SSL dikembangkan oleh *Netscape Communations* pada tahun 1994, dan menjadi protokol yang umum digunakan untuk komunikasi aman antara dua komputer pada Internet. *SSL* dibangun ke dalam banyak *web browser* (termasuk *Netscape Communicator* dan *Internet Explorer*). Ada beberapa versi *SSL*, versi 2 dan versi 3, tetapi versi 3 paling banyak digunakan saat ini.

Untuk memastikan apakah *Internet Explorer* sudah siap menjalankan protokol *SSL*, klik dari *IE*:

Tools → *Internet Options* → *Advanced*

lalu cari pilihan Security, kemudian periksa apakah SSL versi 2.0 atau SSL versi 3.0 telah diberi tanda √ (Gambar 14.6).



Gambar 14.6 Opsi penggunaan *SSL* pada fitur *security* di dalam *Internet Explorer*.

SSL beroperasi antara protokol komunikasi *TCP/IP* (*Transmission Control Protocol*/*Internet Protocol*) dan aplikasi (Gambar 14.7). *SSL* seolah-olah berlaku sebagai lapisan(*layer*) baru antara lapisasan transpor (*TCP*) dan lapisan aplikasi. *TCP/IP* adalah standard protokol yang digunakan untuk menghubungkan komputer dan jaringan dengan jaringan dari jaringan yang lebih besar, yaitu Internet.

Application (*HTTP, FTP, Telnet*)
Security (*SSL*)
Transport (*TCP*)
Network (*IP*)
Data link (*PPP*)
Phyiscal (*modem, ADSL, cable* TV)

Gambar 14.7 Lapisan (dan protokol) untuk *browsing* dengan *SSL*

Di dalam standar komunikasi di Internet, pesan dari pengirim dilewatkan melalui *socket* (*port* khusus yang menerima dan mengirim informasi dari jaringan dengan *mode byte stream*). *Socket* kemudian menerjemahkan pesan tersebut melalui protokol *TCP/IP* (*Transmission Control Protocol*/*Internet Protocol*).

Cara kerja *TCP/IP* (Tanpa *SSL*)

Kebanyakan transmisi pesan di Internet dikirim sebagai kumpulan potongan pesan yang disebut **paket**. Pada sisi pengiriman, paket-paket dari sebuah pesan diberi nomor secara sekuensial. *IP* bertanggung jawab untuk merutekan paket (lintasan yang dilalui oleh paket), dan setiap paket mungkin menempuh rute yang berbeda di dalam Internet. Tujuan sebuah paket ditentukan oleh *IP address*, yaitu nomor yang digunakan untuk mengidentifikasi sebuah komputer pada sebuah jaringan.

Pada sisi penerima, *TCP* memastikan bahwa suatu paket sudah sampai, menyusunnya sesuai nomor urut, dan menentukan apakah paket tiba tanpa mengalami perubahan (misalnya berubah karena *physical error* selama transmisi). Jika paket mengalami perubahan atau ada data yang hilang, *TCP* meminta pengiriman ulang. Bila semua paket dari pesan berhasil mencapai *TCP/IP*, pesan tersebut kemudian dilewatkan ke *socket* penerima. *Socket* tersebut menerjemahkan pesan kembali menjadi bentuk yang dibaca oleh aplikasi penerima (contoh aplikasi adalah *HTTP*, *FTP*, *Telnet*).

Cara kerja TCP/IP (tanpa SSL)

Dari penjelasan di atas dapat dilihat bahwa pada dasarnya *TCP/IP* tidak memiliki pengamanan komunikasi yang bagus. Bahkan, *TCP* tidak cukup canggih menentukan bilamana suatu paket berubah karena diubah oleh pihak ketiga (musuh), karena paket yang diubah tersebut dapat dianggap oleh *TCP* sebagai paket yang benar. Pada transaksi yang menggunakan *SSL*, *SSL* membangun hubungan (*connection*) yang aman antara dua *socket*, sehingga pengiriman pesan antara dua entitas dapat dijamin keamanannya.

SSL disusun oleh dua sub-protokol:

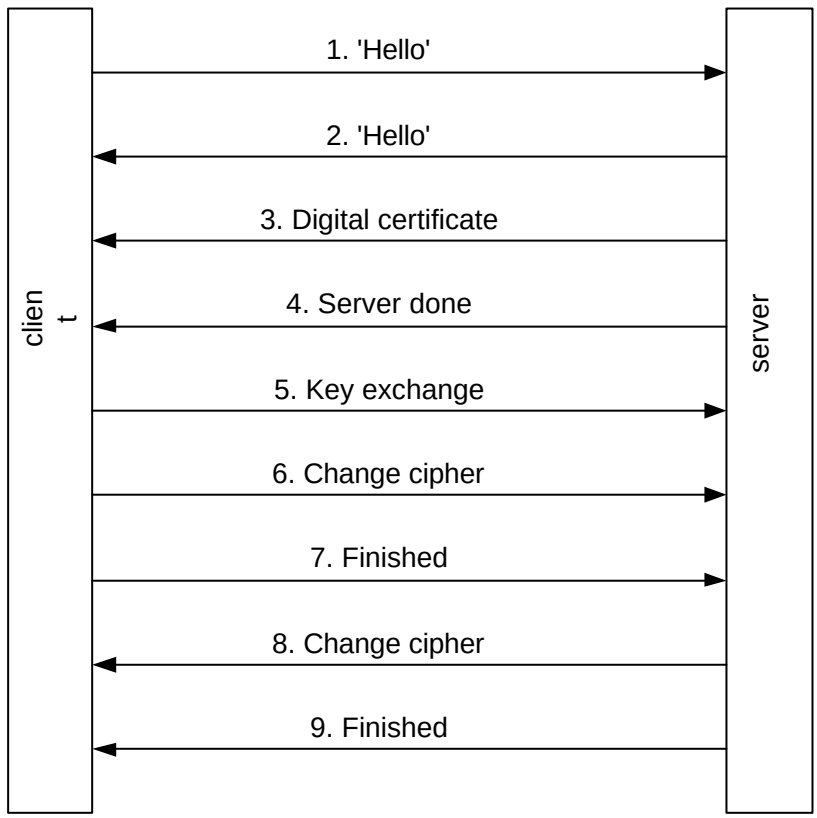
1. *SSL handshaking*, yaitu sub-protokol untuk membangun koneksi (kanal) yang aman untuk berkomunikasi,
2. *SSL record*, yaitu sub-protokol yang menggunakan kanal yang sudah aman. *SSL Record* membungkus seluruh data yang dikirim selama koneksi.

SSL mengimplementasikan kriptografi kunci-publik dengan menggunakan algoritma *RSA* dan sertifikat digital untuk mengotentikasi *server* di dalam trasnsaksi dan untuk melindungi informasi rahasia yang dikirim antara dua buah *socket*. *Server* selalu diotentikasi, sedangkan *client* tidak harus diotentikasi oleh *server*. *Server* diotentikasi agar *client* yakin bahwa ia mengakses situs *web* yang sah (dan bukan situs *web* palsu yang menyamar seolah-olah benar ia adalah *server* yang asli). *Client* tidak harus diotentikasi oleh *server* karena kebanyakan *server* menganggap nomor kartu kredit sudah cukup untuk mengotentikasi *client*.

Perlu dicatat bahwa *SSL* adalah protokol *client-server*, yang dalam hal ini *web browser* adalah *client* dan *website* adalah *server*. *Client* yang memulai komunikasi, sedangkan *server* memberi respon terhadap permintaan *client*. Protokol *SSL* tidak bekerja kalau tidak diaktifkan terlebih dahulu (biasanya dengan meng-klik tombol yang disediakan di dalam *web server*)

1. Sub-protokol handshaking

Sub-protokol *handshaking* diperlihatkan pada Gambar 14.8. Dari gambar tersebut terlihat bahwa *SSL* dimulai dengan pengiriman pesan *Hello* dari *client* ke *server* (1). *Server* merespon dengan mengirim pesan *Hello* (2) dan sertifikat digital ke *client* untuk otentikasi (3).

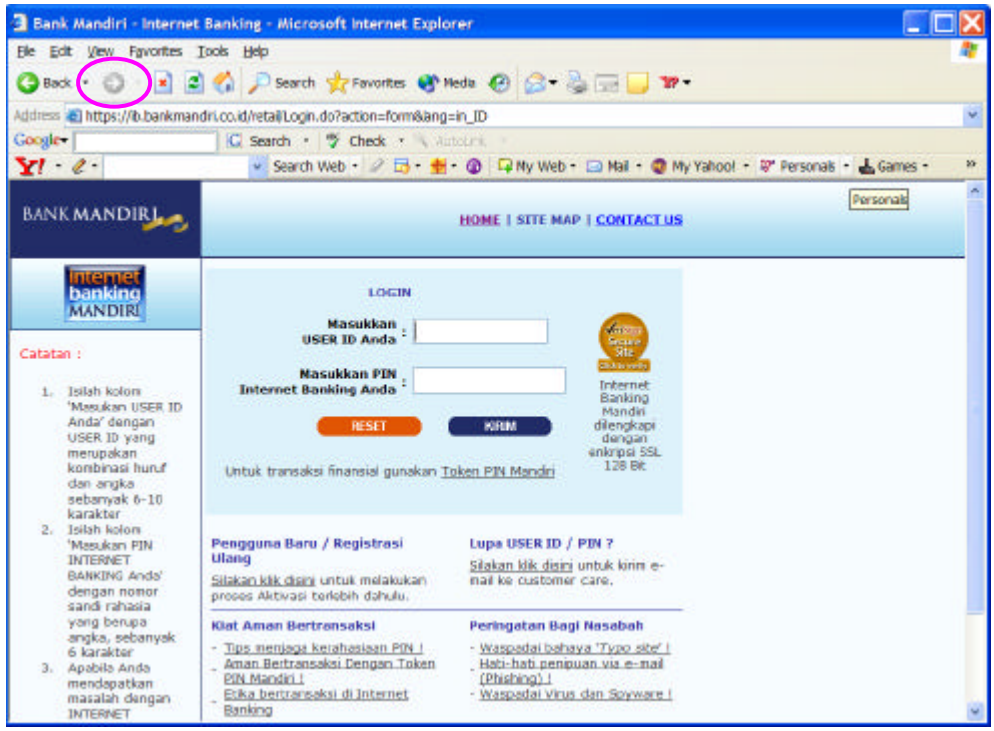


Gambar 14.8. Sub-protokol *handshaking* untuk membangun koneksi yang aman

Sertifikat digital berisi kunci publik *server*. Di dalam *browser client* terdapat daftar *CA* yang dipercaya. Jika sertifikat digital ditandatangi oleh salah satu *CA* di dalam daftar tersebut, maka *client* dapat memverifikasi kunci publik *server*. Setelah proses otentikasi selesai, *server* mengirimkan pesan *server done* (4) kepada *client*.

Selanjutnya, *client* dan *server* menyepakati *session key* untuk melanjutkan transaksi melalui proses yang disebut *key exchange* (5). *Session key* adalah kunci rahasia yang digunakan selama transaksi. Nantinya, komunikasi antara *client* dan *server* dilakukan dengan menggunakan *session key* ini. Data yang akan ditansmisikan dienkripsi terlebih dahulu dengan *session key* melalui protokol *TCP/IP*. Proses *exchange key* diawali dengan *client* mengirim nilai acak 384-bit yang disebut *premater key* kepada *server*. Nilai acak ini dikirim dalam bentuk terenkripsi (dienkripsi dengan kunci publik server). Melalui perhitungan yang cukup kompleks, *client* dan *server* menghitung *session key* yang diturunkan dari *premaster key*. Setelah pertukaran kunci, *client* dan *server* menyepakati algoritma enkripsi (6). *SSL* mendukung banyak algoritma enkripsi, antara lain *DES*, *IDEA*, *RC2*, dan *RC4*. Sedangkan untuk fungsi *hash*, *SSL* mendukung algoritma *SHA* dan *MD5*.

Client mengirim pesan bahwa bahwa ia sudah selesai membangun sub-protokol (pesan 7). *Server* merespon *client* dengan mengirim pesan 8 dan 9. Sampai di sini, proses pembentukan kanal yang aman sudah selesai. Bila sub-protokol ini sudah terbentuk, maka *http://* pada *URL* berubah menjadi *https://* (*http secure*). Gambar 14.9 memperlihatkan *https* pada situs web Bank Mandiri. Proses *SSL* yang cukup panjang ini mengakibatkan sistem menjadi lambat. Oleh karena itu, *SSL* diaktifkan hanya jika *client* memerlukan transmisi pesan yang benar-benar aman.



Gambar 14.9 *http* pada situs web Bank Mndiri berubah menjadi *hhtps* setelah pengaktifan *SSL*

2. Sub-protokol SSL record

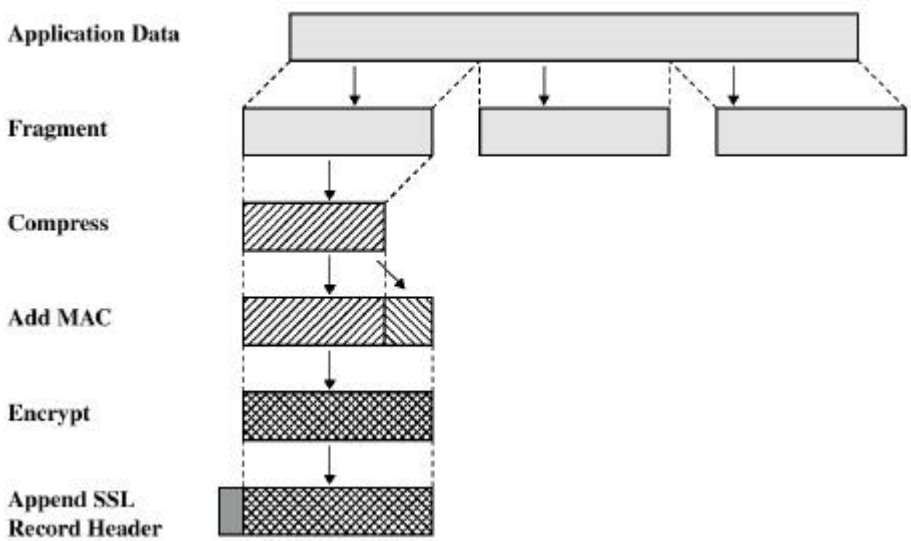
Setelah kanal yang aman terbentuk, *client* dan *server* menggunakannya untuk menjalankan sub-protokol kedua (*SSL record*) untuk saling berkirim pesan. Misalnya *client* mengirim *HTTP request* ke *server*, dan *server* menjawab dengan mengirim *HTTP response*.

Pesan dari *client* ke *server* (dan sebaliknya) dikirim dalam bentuk terenkripsi (pesan dienkripsi dengan menggunakan *session key*). Tetapi, sebelum pesan dikirim dengan *TCP/IP*, protokol *SSL* melakukan proses pemb ungkusan data sebagai berikut:

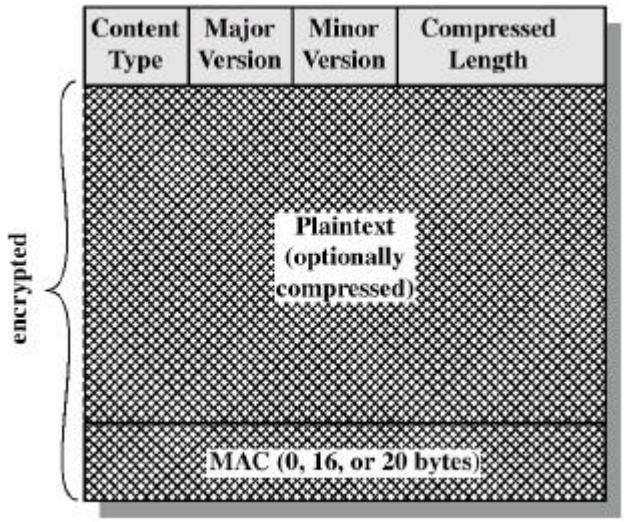
1. Pesan dipecah menjadi sejumlah blok (*fragment*) yang masing-masing panjangnya 16 KB; setiap blok diberi nomor urut sekuensial.
2. Setiap blok kemudian dikompresi, lalu hasil kompresi disambung (*concate*) dengan *session key*;
3. Kemudian, hasil dari langkah 2 di atas di-*hash* dengan algoritma *MD5* (atau algoritma *hash* lain yang disepakati). Nilai *hash* ini ditambahkan ke setiap blok sebagai *MAC* (*Message Authentication Code*). Jadi, *MAC* dihitung sebagai berikut:

   MAC = Hash(session key, compressed data block)
4. Hasil dari langkah 3 kemudian dienkripsi dengan algoritma kriptografi simetri (misalnya *RC4*).
5. Terakhir, hasil dari langkah 4 diberi *header* (2 atau 3 *byte*), baru kemudian dikirim melalui koneksi TCP/IP aman yang terbentuk sebelumnya.

Proses pembungkusan pesan oleh sub-protokol *SSL record* diperlihatkan pada Gambar 14.10. Format SSL record ditunjukkan pada gambar 14.11.



Gambar 14.10. Pembungkusan pesan oleh *SSL record*



Gambar 14.11. Format data *SSL record*

Setelah data sampai di tempat penerima, sub-protokol *SSL* ini melakukan proses berkebalikan: mendekripsi data yang diterima, mengotentikasinya (dengan *MAC*), men-dekompresinya, lalu merakitnya.

Meskipun *SSL* melindungi informasi yang dikirim melalui internet, tetapi ia tidak melindungi informasi yang sudah disimpan di dalam *server* pedagang (*merchant*). Bila pedagang *online* menerima informasi kartu kredit atas suatu pesanan barang, informasi tersebut mungkin di-dekripsi dan disimpan di dalam *server* pedagang sampai pesanan barang diantar. Jika *server* tidak aman dan data di dalamnya tidak dienkripsi, pihak yang tidak berhak dapat aja mengakses informasi rahasia tesrebut.

Piranti keras, seperti kartu *peripheral component interconnect* (*PCI*) yang dirancang untuk digunakan di dalam transaksi *SSL*, dapat dipasang ke dalam *web server* untuk memproses transaksi *SSL*, sehingga mengurangi waktu pemrosesan dan memungkinkan server bebas mengerjakan tugas-tugas lain.

Informasi lebih lanjut mengenai *SSL* dapat diperoleh dari tutorial *SSL* di `www.netscape.com/security/index.html`.

Pada Tahun 1996, *Netscape Communications Corp*. mengajukan *SSL* ke *IETF* (*Internet Engineering Task Force*) untuk standardisasi. Hasilnya adalah *TLS* (*Transport Layer Security*). *TLS* dijelaskan di dalam *RFC* 2246 (untuk informasi lebih lanjut perihal *TLS*, kunjungi situs *IETF* di `www.ietf.org/rfc/rfc2246.txt`). *TLS* dapat dianggap sebagai *SSL* versi 3.1, dan implementasi pertamanya adalah pada Tahun 1999, tetapi belum jelas apakah *TSL* akan menggantikan *SSL*.

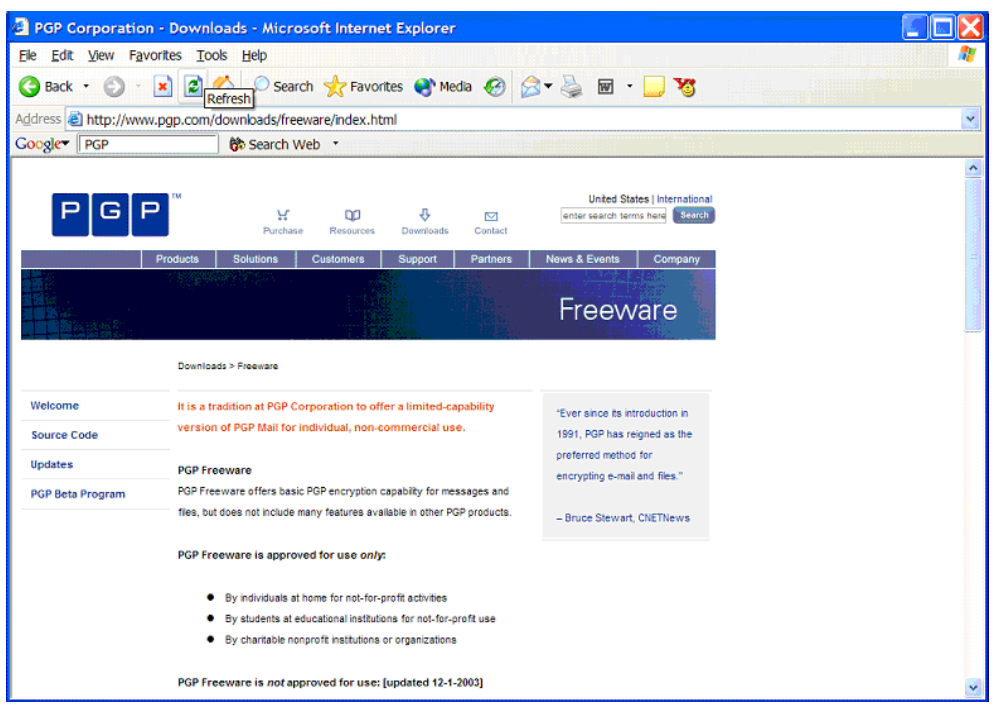
Wireless Transport Layer Security (*WTLS*) adalah protokol keamanan data untuk *Wireless Application Protocol* (*WAP*). *WAP* adalah standard untuk komunikasi nirkabel (*wireless*) pada telepon *mobile* dan peralatan nirkabel lainnya. *WTLS* mengamankan kanal untuk komunikasi antara peralatan nirkabel dan *server* aplikasi.

14.6 Pengamanan *E-mail* dengan *PGP* (*Pretty Good Privacy*)

Pretty Good Privacy atau *PGP* dikembangkan oleh Phil Zimmermann pada akhir tahu 1980. Pada mulanya, *PGP* digunakan untuk melindungi surat elektronik (*e-mail*) dengan memberi perlindungan kerahasiaan (enkripsi) dan otentikasi (tanda-tangan digital). Saat ini *PGP* tidak hanya ditujukan untuk keamanan *e-mail*, tetapi juga untuk keamanan berbagai file dan program pada komputer personal (*PC*).

PGP menggunakan kriptografi simetri dan kriptografi kunci-publik. Oleh karena itu, *PGP* mempunyai dua tingkatan kunci, yaitu kunci rahasia (simetri) – yang disebut juga *session key* – untuk enkripsi data, dan pasangan kunci privat- kunci publik untuk pemberian tanda tangan dan melindungi kunci simetri. Kunci simetri hanya dipakai sekali (*one-time*) dan dibuat secara otomatis dari gerakan tetikus (*mouse*) atau ketikan tombol kunci.

PGP tersedia sebagai *freeware* maupun sebagai paket komersil dalam berbagai versi yang dapat dioperasikan dalam berbagai sistem operasi (*DOS*, *Windows*, *UNIX*, *Mac*). *Download* program *PGP* gratis dari situs www.pgp.org atau www.pgpi.org (lihat Gambar 14.12). Kode sumbernya juga dapat diakses dari Internet. *PGP* terbaru adalah *PGP* versi 8. *PGP* versi-versi awal menggunakan *IDEA* sebagai algoritma simetri dan *RSA* sebagai algoritma kunci-publik (asimetri), sedangkan versi-versi terakhir menggunakan algoritma *CAST* sebagai algoritma simetri dan algoritam *DH* (Diffie-Hellman) sebagai algoritma kunci-publik.

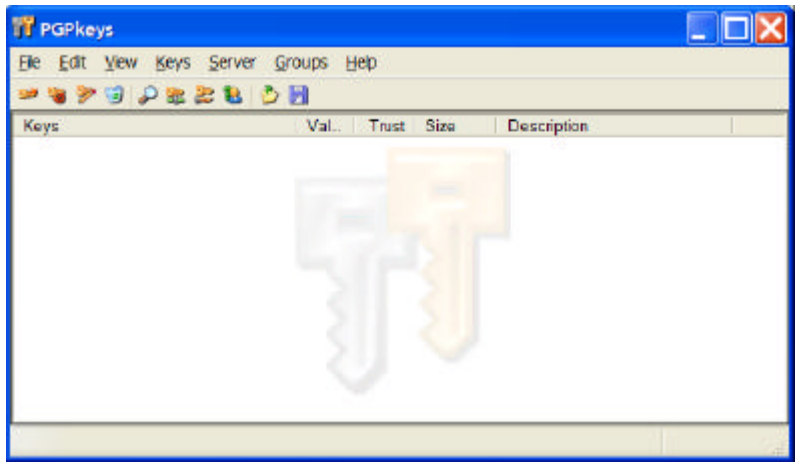


Gambar 14.12 Situs www.pgp.com

Download dari situs *PGP* program *PGP* versi 8.0 *for Windows*, lalu instalasi PGP 8.0 ke dalam komputer anda. Pada versi *freeware* ini, ada tiga program *PGP* yang tersedia: *PGPdisk*, *PGPkeys* (pembangkitan dan manajemen kunci), dan *PGPmail* (enkripsi dan tanda-tangan digital untuk *file* maupun *e-mail*).

Membuat Pasangan Kunci Privat-Kunci Publik Baru

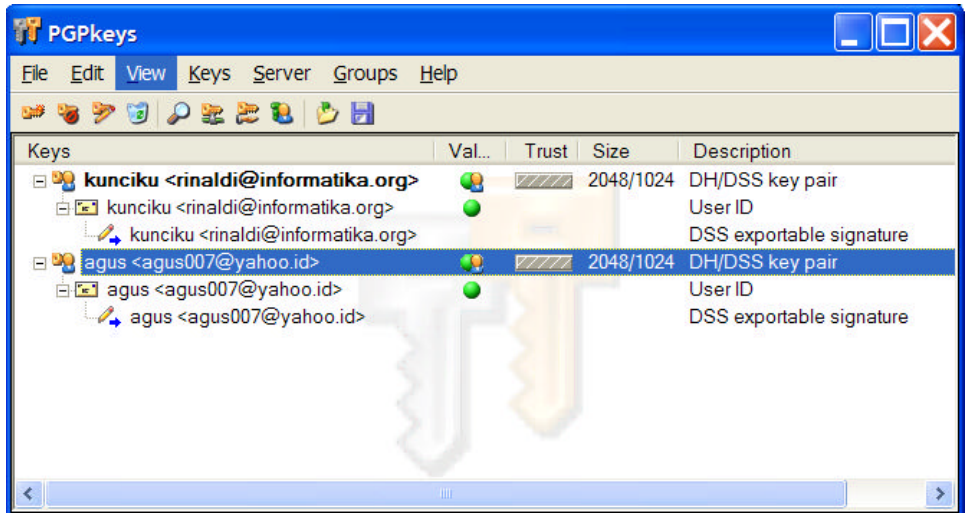
Aktifkan *PGPkeys* sehingga muncul tampilan berikut:



Pilih:

Keys → *New Key*

selanjutnya akan ditampilkan *wizard* untuk membangkitkan pasangan kunci. Isilah beberapa isian yang disediakan. Contoh hasil pembangkitan beberapa pasangan kunci :



Untuk melihat kunci publik, atau memberi kunci publik ke orang lain, ekspor kunci tersebut ke arsip (ekstensi arsip adalah `.asc`).

Contoh kunci publik:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: PGP 8.1 - not licensed for commercial use: www.pgp.com

mQGiBEHDmkIRBAD3p8b3phfk0FftdA2mRqEHLcg/iwF6VzcSde5ng9v86PeEB9xK
BMR9EiUjRdo1Us9YVi8awZ3iZG5EhX5sI/tbuBWJILCARhQzrn7Ww+sAuKrEPg4s
ggZtxYO1FsAbWhB/nKNqgDnYxxY3RbvOYlzH65Bk2xosR3H/YkHqc0L/SQCg/w0S
sh3fkWhymqao7rTJb4B/w2kD+QGWQlz81EkEbQaj3XeE4MdNMDjefKzxp/gP6I7Z
koJyiQxiIm1z4Q2R4iLniUX3hO7Vb9xre+J3s8D+rB0teJ70P7L2RNqK8QLVuqGh
lT1Yy4Okv5uuu0D4yTIOxB+vc3AlAoQwTVVSrKw5I8W7vaXvYBZd2m3w7ItDUjpP
uKhQA/9RjeQBq2QtBA2/7hLPP/NhSSfZ2C9A7rbN0ur3rG7mP0HB+hVFELR7tpW8
Mq+wPHP59qFlGWZpjR0E7svN96pLmQPW5x13Lc8Ip0D1z99o66vZ+UllRFNQR0kk
QO+V3kEIggWFpwOHi/Rz+vCVzrXRpR2CRSPinjxRswoC3vfnL7Qha3VuY2lrdSA8
cmluYWxkaUBpbmZvcm1hdGlrYS5vcmc+iQBdBBARAgAdBQJBw5pCBwsJCAcDAgoC
GQEFGwMAAAAFHgEAAAAACgkQFOUfEytY5dS/SwCg+wXNaoaVjnnMMsqUbF888cJF
WO0AoJWFiXP5yWfWqYSRXfqTAqYv0HGsuQINBEHDmkMQCAD2Qle3CH8IF3Kiutap
QvMF6PlTETlPtvFuuUs4INoBp1ajFOmPQFXz0AfGy0OplK33TGSGSfgMg71l6RfU
odNQ+PVZX9x2Uk89PY3bzpnhV5JZzf24rnRPxfx2vIPFRzBhznzJZv8V+bv9kV7H
AarTW56NoKVyOtQa8L9GAFgr5fSI/VhOSdvNILSd5JEHNmszbDgNRR0PfIizHHxb
LY7288kjwEPwpVsYjY67VYy4XTjTNP18F1dDox0YbN4zISy1Kv884bEpQBgRjXyE
pwpy1obEAxnIByl6ypUM2Zafq9AKUJsCRtMIPWakXUGfnHy9iUsiGSa6q6Jew1Xp
Mgs7AAICCADgUJgMdoFAmVvW3rwTmXtx78O6st/vPoUqMHlGcQeAJ6jFZNj9YzE6
Q5Z3rB6Prv41oTyGBTm/iHFKhluhuA5Zce66KpODlXEWWKesBETkdqMClrmXdBQY
Pff1+NDSpTFfEiJ8YtTz9h3qETCUKEe5u/9oh1e4xCPhjvDTbZKCLV9k7mFyw4Ma
hdRY3moH/3UkDQJD1pD0xdr60d52vMoW71tY2TQ/2tAEbVrRncp9dVXAoqSsOr+J
qRvc0KhP5/5P2u50BobzRJ1nGrlGfRhbI0gR18bZtNLfLDXpHGuMwreYeDxcnUUG
z1gmHb0Xbe/ymsBQoRPqPCdiYM0HDF//iQBMBBgRAgAMBQJBw5pDBRsMAAAAAAoJ
EBTlHxMrWOXUOlcAn3ehXWUDWkHSTW7q6gHpK44VMmpBAKClarHaLAUahiGhHNt2
AMaQYklN/Q==
=GLD1
-----END PGP PUBLIC KEY BLOCK---
```

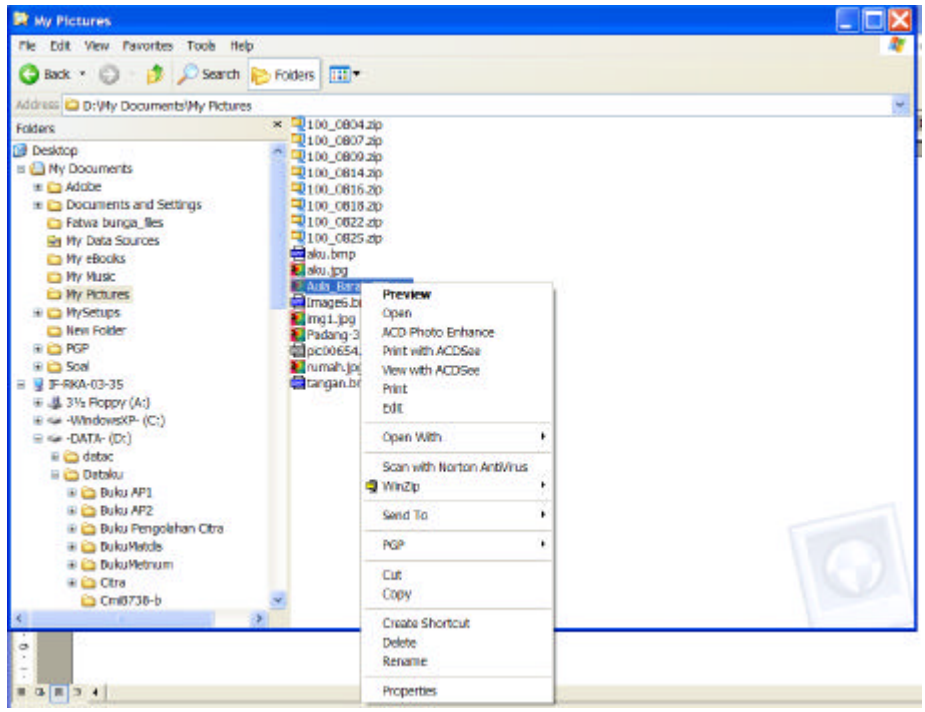
Kunci publik orang lain dapat dimasukkan ke dalam daftar kunci dengan cara memilih menu *Keys* → *Import*.

Mengenkripsi Arsip

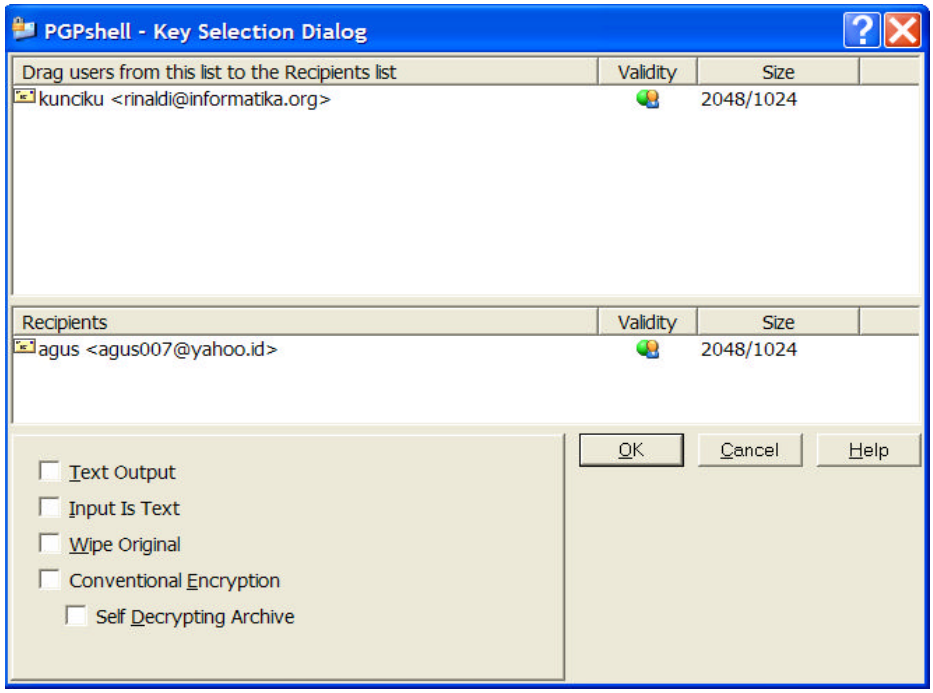
a. Mengenkripsi arsip yang akan dikirim

Ada dua cara mengenkripsi arsip (*file*) yang akan dikirim

1. Melalui *Windows Explorer*
   Pilih arsip yang akan dienkripsi, lalu klik kanan tetikus, dan pilih menu *PGP*.



Dari menu PGP, pilih *Encrypt*, sehingga muncul tampilan berikut:



Pilih *Recipents*(orang yang kita kirimi arsip), selanjutnya tekan *OK*. Arsip akan dienkripsi dengan kunci publik penerima. Hasil enkripsi dapat dipilih untuk disimpan sebagai arsip teks (*Text Output* √) – dengan ektesni nama arsip `.asc` – atau sebagai arsip biner – dengan ekstensi nama arsip `.pgp`.

Contoh arsip `aula-barat.jpg` sebelum dienkripsi:

Arsip `aula-barat.jpg.asc` (*Text output*):

```
-----BEGIN PGP MESSAGE-----
Version: PGP 8.1 - not licensed for commercial use: www.pgp.com

qANQR1DBwU4Dj04oYGwvm0sQCADmCsgqnbpme3mDmoLCAp0lbHmeCtTR9dVQ0CZT
0P43mMkj7iR3F8pEaGOzeAsgj4YXy1iaYkZeUkujWuHeEbtZcjID9cRdOFy/jgNk
zvlSccANEhUcNRBqKCiZM/mesfUTOecXdMJ17T2ApsaMqjVxbHtDDnHGWlsX1Q5U
jysmIyZ9XcaCq9kJYGSK5u5yKYHAe9SQK8/dezYgfLKql//eNYK8ycwSYCzaNNBa
/ZdhAoydKuVCRaQmOROJJXYcI9A2MncjSrYXnELnJVykFM9sjn5xPptrLpybpJ4i
OyxzMzpaDCKAJ5De0lJDKWqGTY+FKwHnk9xmghn/2d5gYu+zB/4+69+UqrOW3jXU
Fna42wWPyWsZ8T5eJ3KvS6OZdIoP5NtT27iSwpJQWWm9X0BLEtv9pyddUWld7QE8
fy3wv+IgGMKLkh7ZgfgwckpBGao06LOHsyLk2YB2Jh25HSiszqqr0N112dhJbmd/
63R8WDtcfgNEh3IirAR6atZsOUPUJKzv6RC7ulQqxZHc3vL/dl+ElpujFOs50qs+
DSnofXvYPZYzrcCi0hS9IjRmPAQo1MwBgmNNpI1Tnp8A7gg09auSQEH3F4DEyEUB
t5s4SaEcJ0cZlt3Ps/HX1z1PTvkWJQbuJDsTVaZbL0KQosEt68EZWagMDGd25kBj
2fALpzoC0uwBkUOpPXtnTYvw/jafWXjtfogXeqHlN0cOu2mNb64S85RgLbv3q6V4
a0OSaLE9qNpSOnyqAibTxz1sKlChPZWwfu/ORkFdsgu4kFzLwXDCSEktWz2a9xJC
Uy5ybGALmRvPxQNMhX8b2JPb3fxtnBDrqRsMXnlgjXMR+8nmk57f6MuSEebyYscg
8PNKkTqKOXOMBc6ZPNOh4ZxnPzHZrsgHvLgbdudie3p9uIFVTEoi5V2qj5O5/oy+
kYgJg+ix+R28zA33iFIFhN7PTfWwuFIlgOpk+7cLr+KDl8TKnzIgfrKdzXYNjErl
jrtZ8ws5JMPDwAgQ6677dUq1i1g3P2zXJwmgsdF9A4uC2JvpJeCbKu3Sy6ZXP4CX

... (deleted, because too long)

-----END PGP MESSAGE-----
```

Untuk mendekripsi arsip, klik arsip yang terenkripsi, selanjutnya klik kanan tetikus, dan pilih *Decrypt*. Program *PGP* meminta anda memasukkan *passphrase* untuk kunci privat (harus sama dengan passphrase yang diisikan pada waktu pembangkitan pasangan kunci).

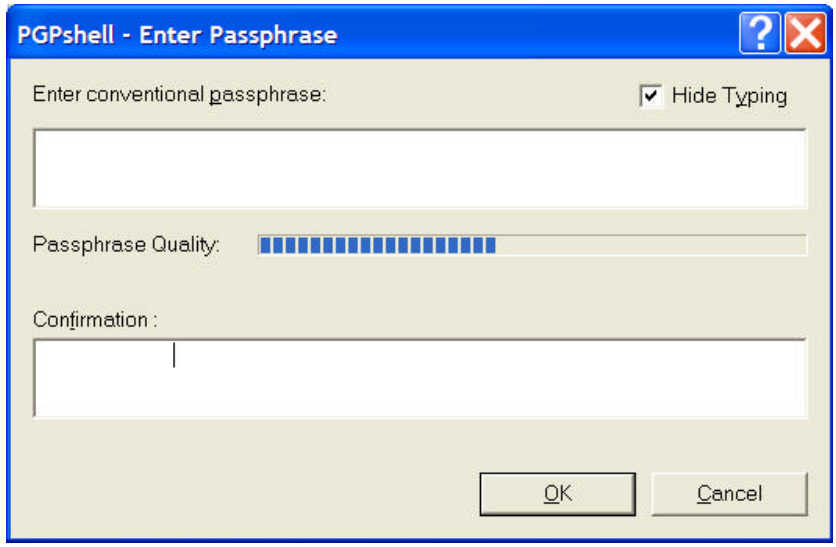
Jika *passphrase* benar, maka arsip akan didekripsi dengan menggunakan kunci privat yang berkoresponden dengan kunci publiknya.

2. Melalui program *PGPmail*
   Aktifkan program *PGPmail*, sehingga muncul tampilan berikut:

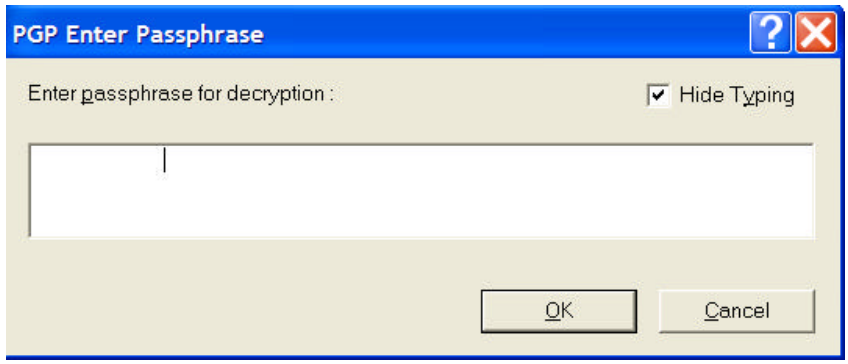
Pilih ikon surat+gembok, dan selanjutnya tahapan enkripsi sama seperti cara pertama.

b. Mengenkripsi arsip dengan algoritma simetri

Jika opsi *Conventional Encryption* dipilih, maka arsip akan dienkripsi dengan algoritma simetri. Di sini kunci simetri dibangkitkan dari *passphrase* yang diketikkan oleh pengguna:



Untuk mendekripsi arsip, klik arsip tersebut, lalu klik kanan tetikus, pilih menu *PGP*, lalu pilih *Decrypt*:



Ketikkan *passphrase* yang sama seperti waktu enkripsi. Hasil dekripsi dapat disimpan dengan nama lain.

Contoh enkripsi arsip `bandung.txt`.

(i) Arsip `bandung.txt` sebelum dienkripsi

```
Pada bulan Oktober 2004 ini, suhu udara kota Bandung terasa
lebih panas dari hari-hari biasanya. Menurut laporan Dinas
Meteorologi Kota Bandung, suhu tertinggi kota Bandung adalah
32 derajat Celcius pada Hari Rabu, 17 Oktober yang lalu.
Suhu terseut sudah menyamai suhu kota Jakarta pada hari-hari
biasa. Menurut Kepala Dinas Meteorologi, peningkatan suhu
tersebut terjadi karena posisi bumi sekarang ini lebih dekat
ke matahari daripada hari-hari biasa.

Sebutan Bandung sebagai kota sejuk dan dingin mungkin tidak
lama lagi akan tinggal kenangan. Disamping karena faktor
alam, jumlah penduduk yang padat, polusi dari pabrik di
sekita Bandung, asap knalpot kendaraan, ikut menambah
kenaikan suhu udara kota.
```

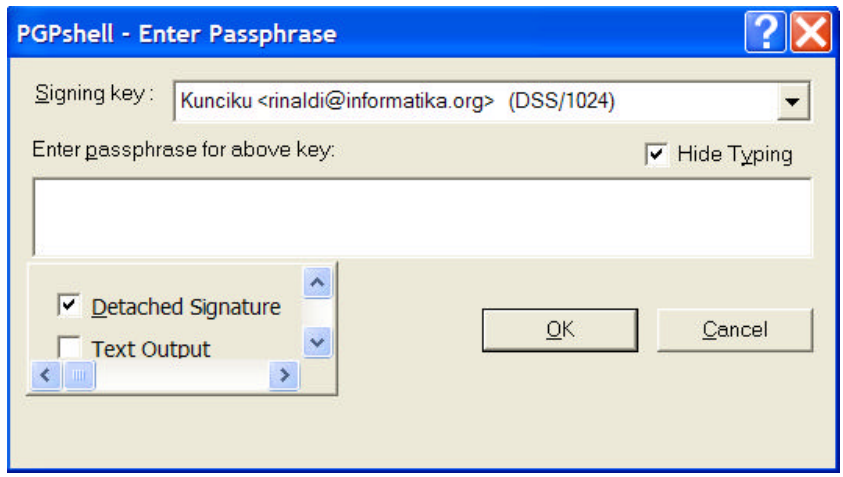
(ii) Arsip `bandung.txt.asc` setelah dienkripsi

```
-----BEGIN PGP MESSAGE-----
Version: PGP 8.1 - not licensed for commercial use: www.pgp.com

qANQR1DDDQQJAwKRt3ROh/zvkWDSwQ8BSzulggHt+bRY/Ma3X/0iEnhSh4xs/q14
m7KjXHi0c7EoQnGvfhZiEA5lzASdqVpUkdR0bRI4F/Vn8D4RWqmmca1qm7KskqRo
+wenFvfQYGBeagM1WOWTrWBKJAPdVG88oCcOE97Bf5YC+Z5f57PAjp5CgrHXj09N
4ElNR2EHohBzhOEAGYIzzzxNBS4kUD8XdThUBqlKSqRO8ZxZora20qYc1oHe79TC
+4T5BG+B+AUCQsTGx8zL2GwoCF/r1ed2SldTJou952gLMpMa6BPvn37VFs1s7EUG
zXa56peaq+bPMYZYW8J69OeoIdDjX6avrbsVOpk07mbOBQl2XbpteKFBz+fjldYE
8MrWblaGL26Q0feoHhckwVsa5uiUrFkjG6mQQubddibenWFMkp64jhDtgyXLJUUi
onFyQsQCFAxQUrcDRw4/0ggmq+qgBtSD4mg2AhhsQCleNbAqWO72yVJcX73eHZ1M
b8NTCMwrsKfMxEs35tY1OU1/SKvi4DlqOgNb5ye0oTzKcpzgJBuk82yJOJFnXkaW
NUysrLONu0kgC/UI3Ma4mtBOAxk4TjNfmEZWmHcI0cDTQ2/FFto8gNxP54qveFB0
IRj8qcpf
=6oEr
-----END PGP MESSAGE-----
```

Memberi Tanda-tangan

- Pilih arsip yang akan ditandatangani, lalu klik kanan tetikus, dan pilih menu *PGP*, kemudian pilih *Sign*.



Arsip akan ditandatangani dengan kunci privat. Masukkan *passphrase* untuk kunci privat. Tandatangan dapat disimpan terpisah menjadi arsip khusus (ekstensi `.sig`) atau digabung menjadi satu dengan arsip (*Detached Signature* √).

Contoh penandatanganan arsip `bandung.txt`.
(i) Arsip `bandung.txt` sebelum ditandatangani

```
Pada bulan Oktober 2004 ini, suhu udara kota Bandung terasa
lebih panas dari hari-hari biasanya. Menurut laporan Dinas
Meteorologi Kota Bandung, suhu tertinggi kota Bandung adalah
32 derajat Celcius pada Hari Rabu, 17 Oktober yang lalu.
Suhu terseut sudah menyamai suhu kota Jakarta pada hari-hari
biasa. Menurut Kepala Dinas Meteorologi, peningkatan suhu
tersebut terjadi karena posisi bumi sekarang ini lebih dekat
ke matahari daripada hari-hari biasa.

Sebutan Bandung sebagai kota sejuk dan dingin mungkin tidak
lama lagi akan tinggal kenangan. Disamping karena faktor
alam, jumlah penduduk yang padat, polusi dari pabrik di
sekita Bandung, asap knalpot kendaraan, ikut menambah
kenaikan suhu udara kota.
```

(ii) Arsip `bandung.txt` setelah ditandatangani

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Pada bulan Oktober 2004 ini, suhu udara kota Bandung terasa
lebih panas dari hari-hari biasanya. Menurut laporan Dinas
Meteorologi Kota Bandung, suhu tertinggi kota Bandung adalah 32
derajat Celcius pada Hari Rabu, 17 Oktober yang lalu. Suhu
terseut sudah menyamai suhu kota Jakarta pada
hari-hari biasa. Menurut Kepala Dinas Meteorologi, peningkatan
suhu tersebut terjadi karena posisi bumi sekarang ini lebih
dekat ke matahari daripada hari-hari biasa.

Sebutan Bandung sebagai kota sejuk dan dingin mungkin tidak lama
lagi akan tinggal kenangan. Disamping karena faktor alam, jumlah
penduduk yang padat, polusi dari pabrik di sekita Bandung, asap
knalpot kendaraan, ikut menambah kenaikan suhu udara kota.

-----BEGIN PGP SIGNATURE-----
Version: PGP 8.1 - not licensed for commercial use: www.pgp.com

iQA/AwUBQcOWFJTpyRekJ1FcEQI8xgCaAmBME/O/lIOfdvZZUfnHcgdhHPAAoPxJ
WPpSilIHIl63h3/iHoB9fIc2
=eKHn
-----END PGP SIGNATURE-----
```