

BUDI RAHARDJO

# KEAMANAN INFORMASI



# *Contents*

<i>Pengantar</i>	5
<i>Pendahuluan</i>	7
<i>Prinsip-prinsip Keamanan Informasi</i>	15
<i>Kriptografi</i>	19
<i>PGP / Gnu Privacy Guard</i>	27
<i>Keamanan Sistem Email</i>	35
<i>Bibliography</i>	41



# Pengantar

Buku ini muncul karena kebutuhan buku teks untuk kuliah keamanan informasi (*information security*). Jenis buku seperti ini agak langka. Bahkan dahulu ilmu yang terkait dengan keamanan - misalnya kriptografi - dianggap tidak boleh diajarkan sehingga referensi untuk hal itu sangat langka. Buku yang pertama kali terbit mengenai kriptografi adalah “Codebreakers” karangan David Kahn <sup>1</sup>, yang diterbitkan tahun 1969. Sejak saat ini, ilmu tentang keamanan (*security*) mulai terbuka untuk umum.

<sup>1</sup> David Kahn. *Codebreakers*. Scribner, 1967

Buku teks berbeda dengan buku *how to* yang banyak beredar di toko buku. Buku tersebut biasanya hanya menjelaskan bagaimana menggunakan sebuah program tertentu, atau melakukan hal tertentu. Sementara itu buku teks digunakan untuk memberikan landasan teori sehingga pemahaman tidak bergantung kepada *tools* tertentu saja. Meskipun demikian, penggunaan *tools* sebagai contoh akan juga disampaikan dalam buku ini. Semoga dengan demikian, buku ini dapat bertahan lebih lama. (Meskipun saya agak ragu setelah melihat pesatnya perkembangan teknologi informasi.)

Urutan pembahasan juga membuat saya merenung cukup panjang. Ada beberapa hal yang disinggung di depan, tetapi pembahasan teorinya di belakang. Sementara itu kalau teorinya diletakkan di depan, maka siswa akan bosan karena terlalu banyak teori. Seharusnya memang buku ini dipaketkan dengan materi presentasi (slide) yang saya gunakan untuk mengajar. Yang itu belum saya benahi. Masih menunggu waktu.

Sebelumnya saya pernah membuat buku yang sejenis, tetapi kode sumber dari buku tersebut sudah hilang. Maklum, saya membuatnya di tahun 1990-an dengan menggunakan program FrameMaker, yang sudah tidak saya miliki lagi. Sekarang saya buat dari awal dengan menggunakan L<sup>A</sup>T<sub>E</sub>X agar lebih bisa bebas.

Ada yang mengatakan bahwa *security* itu seperti pisau yang bermata dua. Dia dapat digunakan untuk kebaikan atau kejahatan, tergantung kepada penggunaannya. Saya berharap agar ilmu yang diperoleh dari membaca buku ini dapat digunakan untuk kebaikan, bukan kejahatan. Saat ini masih dibutuhkan banyak tenaga kerja

yang menguasai security (security professionals). Sayang sekali kalau lowongan ini tidak dapat dipenuhi dan malah banyak yang memilih untuk menjadi perusak.

Bagi Anda yang mengajarkan kuliah *security* dan ingin menggunakan buku ini sebagai buku teks, silahkan digunakan. Bagi para mahasiswa dan peneliti yang membutuhkan referensi untuk makalah Anda, semoga buku ini dapat membantu. Lebih baik lagi apabila Anda dapat menemukan guru yang dapat membantu Anda dalam memahami isi buku ini. Selain buku ini, saya juga menulis buku lain yang dapat diunduh juga: “Keamanan Perangkat Lunak”<sup>2</sup>. Yang ini saya gunakan untuk kuliah saya yang lainnya.

<sup>2</sup> Budi Rahardjo. *Keamanan Perangkat Lunak*. PT Insan Infonesia, 2016

Dikarenakan buku ini masih dalam pengembangan, maka ada banyak bagian yang masih kosong atau meloncat. Mohon dimaafkan. Dalam penulisan selanjutnya, bagian-bagian tersebut akan diisi dan dilengkapi. Mohon masukkan jika hal ini terjadi.

Selamat menikmati versi 0.2 dari buku ini. Semoga bermanfaat.

Bandung, 2017

Budi Rahardjo, peneliti

twitter: @rahard

blog: <http://rahard.wordpress.com>

web: <http://budi.rahardjo.id>

Penulisan referensi:

Budi Rahardjo, “*Keamanan Informasi*”, PT Insan Infonesia, 2017.

This work is licensed under a Creative Commons “Attribution-NonCommercial-ShareAlike 3.0 Unported” license.



# *Pendahuluan*

Selalu ada aspek negatif dari sebuah pemanfaatan teknologi. Teknologi informasi tidak lepas dari masalah ini. Ada banyak manfaat dari teknologi informasi. Sayangnya salah satu aspek negatifnya adalah masalah keamanan (*security*).

Banyak tulisan dan buku yang mengajarkan cara merusak sebuah sistem informasi. Sementara itu buku yang mengajarkan cara pengamanannya agak minim. Demikian pula, ilmu untuk mengamankan sistem berbasis teknologi informasi juga harus lebih banyak diajarkan.

## *Keamanan Informasi*

Ketika kita berbicara tentang *security*, yang muncul dalam benak kebanyakan orang adalah *network security*, keamanan jaringan. Padahal sesungguhnya yang ingin kita amankan adalah **informasi**. Bahwa informasi tersebut dikirimkan melalui jaringan adalah benar, tetapi tetap yang ingin kita amankan adalah informasinya. Nanti akan kita bahas lebih lanjut mengapa demikian. Maka judul dari buku ini adalah “Keamanan Informasi”.

## *Beberapa Contoh Kasus*

Untuk menunjukkan betapa banyaknya masalah keamanan informasi, berikut ini ada beberapa contoh kasus-kasus. Contoh ini bukanlah daftar yang komplis, melainkan hanya sampel dari kondisi yang ada. Bahkan, kemungkinan kondisi yang ada lebih parah daripada contoh-contoh ini.

Beberapa contoh kasus di luar negeri (diurutkan berdasarkan tahun terjadinya) antara lain dapat dilihat dari daftar berikut.

1. 2006-2008. Tahun-tahun ini ditandai dengan mulai masuknya aspek manajemen ke dalam bidang keamanan informasi. Standar ISO (mulai dari 17799 dan kemudian menjadi seri 27000) mulai digunakan di berbagai instansi. Adanya bencana alam (tsunami,

banjir, gempa, dan sejenisnya) membuat orang mulai memikirkan keberlangsungannya sistem IT. Perangkat IT semakin mengecil dalam ukuran sehingga mulai dibawa pengguna ke kantor. Misalnya pengguna membawa sendiri akses internet dengan menggunakan handphone 3G. Penggunaan kartu sebagai pengganti uang juga mulai populer. (Less cash society.)

2. 2013. Virus masih tetap mendominasi masalah. Pencurian identitas (*identity theft*) mulai marak. Cyber war mulai menjadi bagian dari diskusi.
3. 2014. Heartbleed dan Bash Bug. (Yang ini lebih mudah dijelaskan dengan menggunakan gambar. Sayangnya saya tidak memiliki hak untuk memasukkan gambar tersebut ke dalam buku ini. Di kesempatan berikutnya akan saya usahakan memberi penjelasan dengan kata-kata dulu.)
4. 2014. Bursa Singapura terganggu karena masalah software. Perdagangan saham sempat terhenti.
5. 2016. Sebuah firma hukum di Panama bernama Mossack Fonseca (MF) mengalami kebocoran data. Data yang bocor berupa tabungan / investasi orang-orang terkenal dari beberapa negara (termasuk Indonesia). Kasus ini disebut *Panama Papers Breach*. Kebocoran ini diduga karena *Slider plugin* yang digunakan oleh situsnya (yang menggunakan Wordpress) sudah kadaluarsa dan memiliki kerentanan. Hasil eksploitasi memperkenalkan orang untuk mengambil berkas sesukanya.
6. 2016. CCTV digunakan sebagai bagian dari Distributed DoS attack. Ini menunjukkan bahwa perangkat yang menjadi bagian dari Internet of Things (IoT) dapat menjadi target serangan untuk kemudian dijadikan “anak buah” (zombie) untuk menyerang tempat lain. Kode sumber Mirai yang digunakan untuk melakukan penyerangan tersedia di internet. Jika kita tidak siap, ini dapat menjadi masalah yang berikutnya.
7. 2016. Serangan DDoS terhadap berbagai DNS (Domain Name System) servers. Serangan menggunakan bantuan *botnet* sehingga menghabiskan *bandwidth* jaringan dalam orde Gbps.
8. 2017. Sebuah kampus di Amerika Serikat mengalami masalah di jaringan internalnya. Ternyata ada banyak paket dari segmen mesin minuman (atau segmen IoT). Perangkat IoT ternyata diretas (melalui coba-coba password secara brute-force). Kemudian perangkat tersebut menyerang DNS dari kampus.



9. Februari 2017. Layanan cloud dari Amazon.com berhenti berfungsi (down) selama beberapa jam. Berbagai perusahaan yang menyediakan layanan untuk publik dan kebetulan menggunakan layanan cloud (S3) dari Amazon ikut berhenti. Setelah diteliti, penyebabnya adalah salah ketik (typo) dari salah seorang operator <sup>3</sup>.

<sup>3</sup> Catatan dari Amazon dapat dibaca di sini:  
<https://aws.amazon.com/message/41926/>

Selain contoh-contoh di atas, tentunya masih banyak kasus-kasus lain. Ada yang menganalogikan ini sebagai puncak dari *iceberg*. Di bawah laut lebih banyak lagi masalah yang tidak terlihat.

Beberapa contoh kasus yang terkait dengan Indonesia dapat dilihat dari daftar berikut.

1. 1999. Nama domain Timor Timur (.TP) diacak-acak. Diduga pelakunya adalah pemerintah Indonesia. Investigasi lebih lanjut menunjukkan bahwa ini tidak dilakukan oleh pemerintah Indonesia tetapi oleh seseorang (atau sekelompok) yang berada di Amerika Serikat.
2. 2011. Perusahaan Research in Motion (RIM) yang memproduksi *Blackberry* dipaksa untuk memiliki server di Indonesia. Alasan utama adalah agar dapat dilakukan *lawful interception*, yaitu penyadapan secara legal untuk kasus-kasus tertentu. Pihak RIM keberatan. Tidak ada server RIM di Indonesia.
3. 2015. Serangan man-in-the-browser (MITB) dilakukan terhadap berbagai layanan internet banking di Indonesia sehingga mengakibatkan hilangnya uang nasabah<sup>4</sup>
4. 2016. Aplikasi Pokemon Go mulai muncul dan ramai digunakan. Aplikasi ini menggunakan lokasi pengguna sebagai bagian dari permainannya, yaitu untuk menampilkan monster Pokemon sesuai dengan lokasi. Selain itu, foto dari lingkungan sekitarnya dapat juga kita ambil dan kita bagikan (share) dengan orang lain melalui media sosial. Aplikasi ini dilarang digunakan di lingkungan militer dan pemerintahan karena dikhawatirkan dapat membocorkan data rahasia. (Sebetulnya ada banyak aplikasi lain yang juga menggunakan data lokasi seperti *Waze* dan *Google Maps*, tetapi ini tidak “terlihat”. Bahkan lebih berbahaya lagi adalah penggunaan layanan email gratisan untuk akun resmi pemerintahan atau instansi lain di Indonesia.)
5. 2016. Berbagai *market place* (seperti Tokopedia, Bukalapak, dll.) dan aplikasi handphone (seperti Go-Jek) diserang oleh orang yang mencoba melakukan password cracking. Asumsinya adalah seseorang akan menggunakan userid (alamat email) dan password yang sama untuk situs-situs tersebut. Identitas yang bocor di

<sup>4</sup> <http://regional.kompas.com/read/2015/08/11/12185971/> Kronologi. Hilangnya. Uang. Nasabah. Bank. Mandiri. Versi. Korban

sebuah layanan (web site, application) dicoba digunakan di tempat lain.

6. 2016. Topik pembentukan “Badan Cyber Nasional (BCN)” mulai hangat dibicarakan.
7. 2017. Seorang (beberapa?) mahasiswa di sebuah perguruan tinggi di Indonesia meminta bantuan cracker untuk mengubah nilainya di sistem informasi kampusnya.
8. Maret 2017. Listrik dari tempat data center dari sebuah ISP mati sehingga layanannya terhenti. Beberapa *electronic market places* ikut terkena imbasnya karena menggunakan layanan ISP tersebut.

Saat ini semakin banyak lagi masalah keamanan yang ditemui. Hal ini disebabkan semakin banyak pemanfaatan teknologi informasi dan jaringan internet. Selain itu teknik untuk menemukan lubang keamanan juga semakin canggih sehingga lebih banyak ditemukan kelemahan-kelemahan tersebut.

Sebuah survey yang dilakukan oleh *Information Week* di Amerika Serikat (tahun?) menunjukkan bahwa hanya 22 persen manager yang menganggap keamanan sistem informasi sebagai hal yang penting. Bagaimana meyakinkan mereka untuk melakukan investasi di pengamanan?

Rendahnya kesadaran atas masalah keamanan (lack of security awareness) merupakan salah satu kunci utama munculnya masalah keamanan. Para praktisi juga masih menjalankan kebiasaan buruk, seperti misalnya berbagi password admin.

Masalah keamanan informasi yang biasanya berupa data teknis harus diterjemahkan ke angka finansial agar dapat dimengerti oleh pihak pimpinan. Sebagai contoh, di Inggris ada survey mengenai berapa biaya yang dikeluarkan perusahaan jika sistem mereka tidak dapat diakses (*down*).

### *Security Life Cycle*

Banyak orang yang beranggapan bahwa masalah keamanan informasi dapat dipecahkan dengan membeli produk keamanan, misalnya firewall, anti-virus, dan seterusnya. Keamanan informasi sebetulnya berupa sebuah siklus sebagaimana ditampilkan pada Gambar 1.

Sesuatu yang akan kita amankan disebut dengan “aset”. Untuk itu, langkah pertaman dalam pengamanan adalah menentukan aset yang ingin dilindungi. Apa saja yang dianggap sebagai aset harus ditentukan bersama dengan pemilik dari sistem (aplikasi, data, dsb.) sebab mereka yang mengetahui mana aset dan mana yang

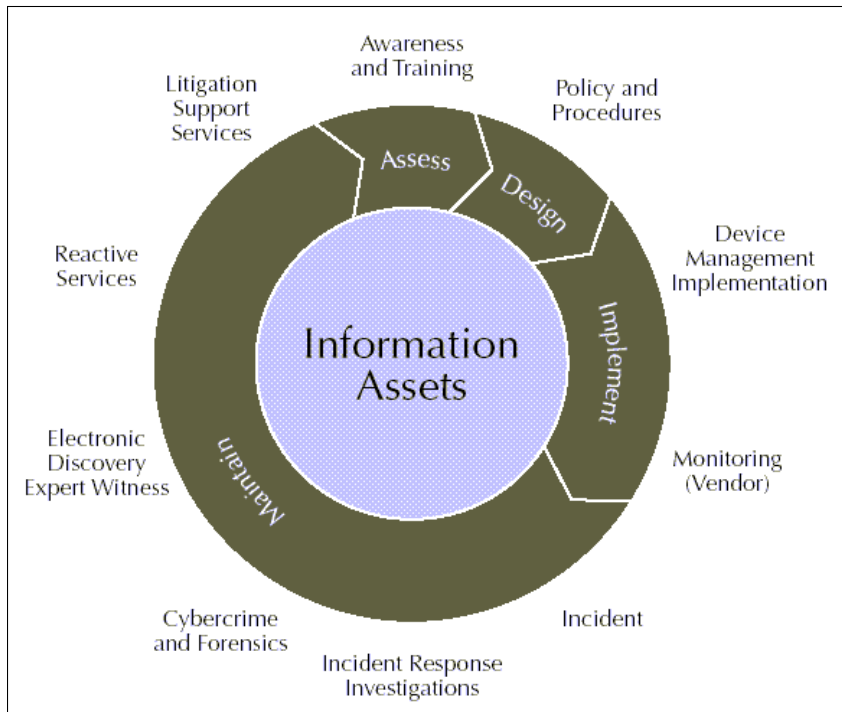


Figure 1: Security Life Cycle

bukan aset. Proses ini disebut *assessment* dan dapat dilakukan dengan melalui training atau *awareness* terhadap pihak-pihak terkait. Seringkali pemilik aplikasi memahami mana asetnya tetapi pihak operasional (orang-orang IT yang diberi tugas untuk mengamankan sistem) tidak tahu.

Setelah mengetahui aset yang ingin diamankan, aset tersebut harus kita beri harga (value). Pengamanan nantinya akan disesuaikan dengan nilai dari aset tersebut. Akan sulit kita melakukan investasi pengamanan yang biasanya lebih mahal dari nilai asetnya. Sebagai contoh, jika kita memiliki sebuah komputer yang harganya Rp. 5.000.000,- maka akan sulit untuk menerima biaya pengamanan yang harganya Rp. 100.000.000,- (lebih mahal). Biaya pengamanan harus lebih murah daripada nilai asetnya. Jika biaya pengamanan lebih mahal, mungkin lebih baik membeli barang sejenis saja sebagai duplikat.

Untuk hal-hal yang terkait dengan teknologi informasi, pendaftaran aset-aset ini tidak mudah karena ada hal-hal yang tidak terlihat secara kasat mata. Aset ini dapat kita bagi menjadi tiga (3) jenis; *hardware*, *software*, dan data. Mari kita mulai mencoba mendata.

Aset yang berbentuk perangkat keras (*hardware*) agak “mudah” didata karena terlihat oleh mata, tetapi ada beberapa hal yang membuatnya menjadi susah. Salah satunya adalah nilai dari aset tersebut.

Harga komputer cenderung jatuh dengan cepat. Berapa depresiasi dari sebuah server? Contoh-contoh aset perangkat keras antara lain komputer, router, perangkat jaringan, disk, dan seterusnya. Apakah notebook termasuk aset atau barang habis? Bagaimana dengan USB flashdisk? Apakah itu termasuk aset juga?

Beberapa kejadian terkait dengan kesulitan mendata perangkat keras antara lain tidak diketahuinya pemilik dari perangkat tersebut. Database perangkat keras sering tidak tersedia. Sebagai contoh, sering kali tidak diketahui harddisk dan lokasi server yang menggunakan disk tersebut.

Jika pendataan perangkat keras sudah susah, maka pendataan perangkat lunak lebih susah lagi. Masalahnya, perangkat lunak tidak terlihat secara kasat mata sehingga pendataannya harus melalui pemilik layanan / pemilik aplikasi. Sebagai contoh, sebuah layanan online memiliki aplikasi di server web dan juga database di server database. Aplikasi-aplikasi tersebut berbentuk beberapa perangkat lunak yang tentunya memiliki harga.

Penentuan harga (nilai) dari perangkat lunak cukup rumit. Untuk aplikasi yang dibeli, dapat digunakan harga pembelian tersebut. Bagaimana menentukan harga aplikasi yang dikembangkan sendiri? Ada yang menggunakan jumlah waktu pengembang (dalam *man days*) yang kemudian dikalikan dengan honor (gaji) orang tersebut. Itulah harga dari aplikasi tersebut. Lantas bagaimana dengan produk *free software* atau (sebagian dari) *open source* yang kebanyakan dapat diperoleh secara gratis? Bagaimana menentukan harga mereka? Ini masih menjadi pertanyaan. Hal lain yang menyulitkan adalah berapa depresiasi dari perangkat lunak?

Bagian selanjutnya dari aset teknologi informasi adalah data. Jika pendaftaran aset hardware dan software sudah sukar, pendaftaran data lebih sukar lagi. Data apa saja yang dimiliki oleh sistem? Pada umumnya data apa saja yang tersedia tidak terdaftar. Masing-masing aplikasi hanya memiliki data tersendiri.

Penentuan harga dari data lebih sukar lagi. Sebagai contoh, berapa harga data transkrip mahasiswa? Bagi mahasiswa, data tersebut sangat berharga sehingga harus dilindungi. Bagi orang lain, data tersebut mungkin tidak ada nilainya. Maukah Anda membeli data transkrip mahasiswa ITB seharga Rp. 30.000.000,- ? Saya yakin tidak ada yang mau. Bagaimana jika Rp. 3.000.000,- saja? Mungkin masih tidak. Bagaimana jika Rp. 3.000,- ? Mungkin mau. Apakah nilai dari data transkrip tersebut hanya tiga ribu rupiah? Jika iya, bagaimana nilai perlindungan yang akan kita berikan?

Setelah mengetahui aset yang akan dilindungi, langkah pertama yang dilakukan adalah melakukan desain pengamanan. Hal ini dilakukan dengan mengembangkan kebijakan dan prosedur (*policies*

*and procedures*). Banyak yang melupakan langkah ini dan langsung melakukan implementasi, tetapi tanpa PnP ini akan sulit. Sebagai contoh, siapa yang boleh melakukan akses kepada data transkrip tersebut? Ini dituangkan dalam kebijakan. Tanpa kebijakan ini, perangkat pengamanan yang ada (*authorization* dan *access control*) akan sulit diterapkan. Rules apa yang akan dipakai? Banyak kejadian sistem pengamanan diterapkan dengan salah karena tidak memiliki desain yang benar.

Desain pengamanan ini kemudian diterapkan secara teknis melalui perangkat pengamanan (*security devices*). Penerapan ini dapat meminta bantuan vendor.

Meskipun sudah diterapkan pengamanan, insiden keamanan akan tetap dapat terjadi. Ketika insiden ini terjadi, maka harus dilakukan investigasi terlebih dahulu. Apakah insiden yang terjadi tersebut benar-benar insiden ataukah kejadian biasa saja? Jika memang itu adalah insiden, maka akan diproses lebih lanjut sesuai dengan kebijakan dan prosedur yang ada.

Ini kemudian menjadi siklus; *security life cycle*. Banyak orang yang masih menganggap *security* adalah sebuah produk sehingga mereka lebih fokus kepada pembelian produk pengamanan tertentu tanpa memperhatikan faktor lain (misalnya aset mana yang akan dilindungi). Ini seperti mengatasi sakit kepala dengan menggunakan obat penghilang rasa sakit tanpa perlu mencari tahu apa sumber permasalahan sesungguhnya.

### *Keamanan Elemen Sistem*

Dahulu, ketika kita berbicara tentang *security*, maka yang ada dalam kepala sebagian besar orang adalah *network security* (keamanan jaringan komputer). Padahal sistem teknologi informasi tidak hanya jaringan saja.

Sebuah sistem berbasis teknologi informasi memiliki beberapa elemen (komponen), yaitu komputer (*host*, *server*, *workstation*), jaringan (berserta perangkatnya), dan aplikasi (*software*, *database*). Keamanan dari masing-masing elemen tersebut akan berbeda penanganannya.

### *Computer Security*

*Computer security* (keamanan komputer) terkait dengan keamanan komputer, yang boleh jadi merupakan *server*, *workstation*, *notebook*, dan sejenisnya. Seringkali ini disebut juga sebagai *host security*.

Permasalahan yang muncul pada keamanan komputer terkait dengan sistem operasi yang digunakan, *patch* yang sudah dipasang, konfigurasi yang digunakan, serta keberadaan aplikasi yang ada.

Seringkali sistem operasi yang digunakan sudah kadaluwarsa dan sudah ditemukan beberapa kerentanan (vulnerability) pada sistem operasinya.

### *Network Security*

Saat ini sebagian besar sistem terhubung dengan jaringan (apapun jenis jaringannya). Ada beberapa masalah terkait dengan keamanan jaringan, seperti misalnya penyadapan data, modifikasi data, dan juga serangan *Denial of Service* terhadap jaringan dengan membanjiri jaringan dengan paket yang sangat banyak (sangat besar).

### *Application Security*

Boleh jadi komputer (server) yang digunakan sudah aman dan jaringan yang digunakan sudah aman, tetapi aplikasi (software) yang digunakan memiliki kerentanan sehingga data dapat diambil (atau diubah) oleh orang yang tidak berhak. Contoh yang sering terjadi saat ini adalah *SQL injection*.

Terkait dengan application security adalah keamanan sistem database. Ternyata penanganan masalah keamanan database mirip dengan penanganan keamanan jaringan (bukan aplikasi).

Topik keamanan aplikasi atau software mulai menjadi penting dan ini menjadi topik buku tersendiri.

# *Prinsip-prinsip Keamanan Informasi*

Ada beberapa prinsip utama dalam keamanan informasi. Bab ini akan membahas prinsip-prinsip tersebut secara singkat. Hal-hal yang lebih rinci dan teknis, misalnya bagaimana mengimplementasikan aspek keamanan, akan dibahas pada bagian terpisah.

## *Aspek Keamanan*

Ketika kita berbicara tentang keamanan informasi, maka yang kita bicarakan adalah tiga hal; *confidentiality*, *integrity*, dan *availability*. Ketiganya sering disebut dengan istilah **CIA**, yang merupakan gabungan huruf depan dari kata-kata tersebut. Selain ketiga hal tersebut, masih ada aspek keamanan lainnya.

Ketika kita berbicara tentang keamanan sebuah sistem - jaringan, aplikasi, atau apa pun - yang kita lakukan adalah mengevaluasi aspek C, I, dan A dari sistem tersebut. Prioritas dari ketiga aspek tersebut berbeda-beda untuk jenis sistem dan organisasi yang menggunakannya. Ada sistem yang aspek *integrity* lebih penting daripada kerahasiaannya (*confidentiality*). Untuk itu, pahami ketiga aspek ini. Ini adalah prinsip utama dari keamanan.

## *Confidentiality*

*Confidentiality* atau kerahasiaan adalah aspek yang biasa dipahami tentang keamanan. Aspek *confidentiality* menyatakan bahwa data hanya dapat diakses atau dilihat oleh orang yang berhak. Biasanya aspek ini yang paling mudah dipahami oleh orang. Jika terkait dengan data pribadi, aspek ini juga dikenal dengan istilah *Privacy*.

Serangan terhadap aspek *confidentiality* dapat berupa penyadapan data (melalui jaringan), memasang *keylogger* untuk menyadap apa-apa yang diketikkan di keyboard, dan pencurian fisik mesin / disk yang digunakan untuk menyimpan data.

Perlindungan terhadap aspek *confidentiality* dapat dilakukan dengan menggunakan kriptografi, dan membatasi akses (segmentasi jaringan)

### *Integrity*

Aspek *integrity* mengatakan bahwa data tidak boleh berubah tanpa ijin dari yang berhak. Sebagai contoh, jika kita memiliki sebuah pesan atau data transaksi di bawah ini (transfer dari rekening 12345 ke rekening 6789 dengan nilai transaksi tertentu), maka data transaksi tersebut tidak dapat diubah seenaknya.

TRANSFER 12345 KE 6789 100000

Serangan terhadap aspek *integrity* dapat dilakukan oleh *man-in-the-middle*, yaitu menangkap data di tengah jalan kemudian mengubahnya dan meneruskannya ke tujuan. Data yang sampai di tujuan (misal aplikasi di web server) tidak tahu bahwa data sudah diubah di tengah jalan.

Perlindungan untuk aspek *integrity* dapat dilakukan dengan menggunakan *message authentication code*.

### *Availability*

Ketergantungan kepada sistem yang berbasis teknologi informasi menyebabkan sistem (beserta datanya) harus dapat diakses ketika dibutuhkan. Jika sistem tidak tersedia, *not available*, maka dapat terjadi masalah yang menimbulkan kerugian finansial atau bahkan nyawa. Itulah sebabnya aspek *availability* menjadi bagian dari keamanan.

Serangan terhadap aspek *availability* dilakukan dengan tujuan untuk meniadakan layanan atau membuat layanan menjadi sangat lambat sehingga sama dengan tidak berfungsi. Serangannya disebut *Denial of Service* (DOS).

Perlindungan terhadap aspek *availability* dapat dilakukan dengan menyediakan redundansi. Sebagai contoh, jaringan komputer dapat menggunakan layanan dari dua penyedia jasa yang berbeda. Jika salah satu penyedia jasa jaringan mendapat serangan (atau rusak), maka masih ada satu jalur lagi yang dapat digunakan.

### *Aspek Keamanan Lainnya*

Selain ketiga aspek utama (CIA), yang sudah dibahas pada bagian sebelumnya, ada aspek keamanan lainnya. Yang ini sifatnya tambahan, meskipun kadang menjadi bagian yang cukup signifikan juga.

### *Non-repudiation*

Aspek *non-repudiation* atau nir-sangkal digunakan untuk membuat para pelaku tidak dapat menyangkal telah melakukan sesu-



atu. Aspek ini biasanya kental di dalam sistem yang terkait dengan transaksi. Contoh penggunaannya adalah dalam sistem lelang elektronik.

Implementasi dari aspek ini dapat dilakukan dengan menggunakan *message authentication code* (dengan menggunakan fungsi *hash*) dan pencatatan (logging).

### *Authentication*

Proses *Authentication* digunakan untuk membuktikan klaim bahwa seseorang itu adalah benar-benar yang diklaim (bagaimana membuktikan bahwa saya adalah pengguna dengan nama “budi”).

Proses pembuktian seseorang ini lebih mudah dilakukan di dunia nyata dibandingkan dengan di dunia maya (siber, *cyber*). Di dunia nyata akan sulit bagi saya untuk membuat klaim palsu bahwa saya seorang wanita. (Saya memiliki kumis dan jenggot.) Namun di dunia maya, saya dapat membuat klaim bahwa saya seorang wanita dengan hanya memilih nama wanita dan memasang foto wanita.

Proses *authentication* ini dapat dilakukan dengan bantuan hal lain, yang sering disebut “faktor”. (Sehingga ada istilah *two-factor authentication*.) Faktor-faktor tersebut adalah sebagai berikut.

1. Sesuatu yang diketahui. Contoh dari faktor ini adalah nama, userid, password, dan PIN.
2. Sesuatu yang dimiliki. Contoh dari faktor ini adalah kartu, kunci, dan token.
3. Sesuatu yang menjadi bagian dari fisik pengguna. Contoh dari faktor ini adalah sidik jari, retina mata, dan *biometric* lainnya.

Selain faktor-faktor di atas, ada juga yang menambahkan faktor lain seperti berikut ini:

1. orang tersebut berada di tempat tertentu. (Proximity);
2. authentication dengan menggunakan bantuan pihak lain, pihak ketiga yang terpercaya (trusted third party).

### *Authorization*

Pada aspek sebelumnya, *authentication*, kita dapat mengetahui siapa pengguna dan *roles* dari pengguna tersebut. Selanjutnya hak akses akan diberikan kepada pengguna sesuai dengan *roles* yang dimilikinya. Inilah aspek *authorization*. Perlu diingatkan kembali bahwa aspek *authorization* ini membutuhkan *authentication*, sehingga dia letaknya setelah *authentication*.



# Kriptografi

Ada dua cara untuk mengamankan data, yaitu menyembunyikan data atau menyandikan data. Cara pertama menggunakan steganografi, sementara cara kedua menggunakan kriptografi. Bab ini akan membahas lebih banyak tentang kriptografi, meskipun steganografi akan disinggung secara singkat.

Ilmu ini pada awalnya dianggap terlarang untuk diajarkan sehingga tidak ada bahan bacaan untuk mempelajarinya. Setelah David Kahn membuat bukunya di tahun 1969, maka ilmu pengamanan data ini menjadi lebih terbuka untuk dipelajari. Saat ini sudah sangat banyak buku yang membahas mengenai hal ini, mulai dari yang umum <sup>5</sup> (tidak teknis) sampai ke yang teknis.

<sup>5</sup> Steven Levy. *Crypto: How the Code Rebels Beat the Government Saving Privacy in the Digital Age*. Penguin Books, 2001

## Steganografi

Steganografi (*steganography*) adalah ilmu untuk menyembunyikan pesan sehingga tidak terlihat dengan mudah. Mekanisme penyembunyian (*hide, concealment*) dilakukan dengan menggunakan media lain. Sebagai contoh, kita dapat menyembunyikan pesan dalam gambar (*image, foto*), audio, atau video. Dalam sejarahnya, penyembunyian pesan dapat dilakukan dengan menggunakan meja yang dilapisi lilin (jaman perang antara Yunani dan Persia).

Saat ini steganografi digunakan sebagai bagian dari *Digital Rights Management* (DRM), misalnya dengan menyisipkan informasi mengenai HaKI dari produk digital (musik, ebook, foto, dan sejenisnya).

Ada banyak cara untuk menyisipkan informasi ke dalam berkas digital. Sebagai contoh, kita dapat menyisipkan pesan di dalam gambar (foto) digital seperti ditampilkan pada Gambar 2. Cara yang digunakan adalah dengan menggunakan *least significant bit* (LSB) dari data *pixel* gambar. Misalnya, sebuah *pixel* (titik) di gambar direpresentasikan oleh 8-bit. Dalam hal ini ada 256 kombinasi (katakanlah skala abu-abu / grey scale). Maka kita dapat mengorbankan bit yang ke-8 tersebut dan hanya menggunakan 7-bit saja dalam pewarnaan. Bit ke-8 dapat kita gunakan sebagai bagian dari data. Jika ada 8 bit yang berurutan kita perlakukan seperti itu, maka akan ada tempat



Figure 2: Contoh Watermark

untuk 8-bit data (1 byte). Jika ini kita teruskan lagi, maka kita dapat menyimpan beberapa (banyak) bytes di dalam gambar itu. Data yang kita simpan di dalam berkas (gambar) tersebut dapat berupa kepemilikan gambar, *copyright*, misalnya.

Tentu saja contoh di atas agak sedikit menyederhanakan permasalahan. Steganografi yang bagus akan tahan bantingan terhadap proses-proses tertentu. Sebagai contoh, jika gambar tersebut diubah ukurannya (*resize*) maka informasi yang ditanamkan tersebut akan hilang. Algoritma steganografi yang bagus akan memiliki kekuatan yang lebih dari itu.

### Kriptografi

Berbeda dengan steganografi, kriptografi tidak menyembunyikan pesan tetapi mengubah pesan sehingga sulit diperoleh pesan aslinya. Pesan diubah dengan cara **transposisi** (mengubah letak dari huruf) dan **substitusi** (mengganti huruf/kata dengan huruf/kata lainnya). Pesan yang sudah diubah terlihat seperti sampah, tetapi tetap terlihat oleh penyerang atau orang yang tidak berhak.

Proses transposisi dapat dilakukan dengan berbagai cara. Sebagai contoh, kita dapat menulis pesan menjadi dua baris secara bergantian. Huruf pertama diletakkan di baris pertama, huruf kedua di baris kedua, huruf ketiga di baris pertama lagi, huruf keempat di baris kedua lagi, dan seterusnya. Mari kita ambil contoh kalimat “selamat datang di kota bandung”. Kalimat ini akan kita tuliskan secara

bergantian. (Dalam contoh ini, spasi kita buang saja.)

```
slmtaagioadn
eaadtndktbnug
```

Kalimat yang akan dikirimkan menjadi “slmtaagioadneaadtndktbnug”. Kalimat ini yang kita kirimkan. Dapat dilihat bahwa kalimat yang dikirimkan sudah sulit dibaca isinya. Di sisi penerima akan dilakukan proses sebaliknya sehingga didapat kalimat aslinya. Ini adalah salah satu contoh proses transposisi.

Proses substitusi dalam kriptografi dilakukan dengan menukarkan simbol dengan simbol lain. Sebagai contoh kita dapat menukarkan huruf dengan huruf lain sesuai dengan sebuah aturan.

**Caesar cipher** merupakan salah satu contoh kriptografi dengan menggunakan metoda substitusi. Cara kerjanya adalah sebagai berikut. Urutkan huruf abjad dari “A” sampai ke “Z”. Di bawahnya kita geser huruf-huruf tersebut sebanyak tiga lokasi. Hasilnya seperti terlihat di bawah ini. (Penggunaan huruf besar dan kecil hanya untuk memperjelas saja. Seharusnya kedua baris memiliki huruf yang sama.)

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
d e f g h i j k l m n o p q r s t u v w x y z a b c
```

Katakan kita ingin mengirim pesan yang berisi kata “BUDI”. Yang kita lakukan adalah mengambil huruf-huruf di bawah masing-masing huruf sebagai penggantinya. Sebagai contoh, huruf “B” akan digantikan dengan “e”, “U” digantikan “x”, “D” digantikan “g”, dan “I” digantikan “l”. Sehingga “BUDI” akan menjadi “EXGL”.

Jumlah pergeseran huruf - apakah 3 atau 7 - dapat ditentukan bersama. Namun ada yang menarik jika kita gunakan 13 sebagai jumlah pergeseran. Jumlah huruf kita ada 26, sehingga 13 merupakan angka “ajaib”. Sebuah pesan dapat kita geser 13 sehingga menjadi tersembunyi dan kalau kita geser 13 lagi (dengan proses yang sama) akan menghasilkan teks aslinya. Proses ini dikenal dengan istilah **ROT13** (atau rotate 13)<sup>6</sup>.

```
FHQNUXNU NAQN ZRAPBON?
```

Cara substitusi seperti yang digunakan oleh *Caesar cipher* tersebut menggunakan satu tabel sehingga sebuah huruf akan disubstitusi oleh huruf yang sama. Dalam contoh pertama, huruf “B” akan disubstitusi dengan huruf “E”. Hal ini disebut *monoalphabetic cipher*.

Persandian dengan menggunakan *Caesar cipher* ini bertahan cukup lama. Dapatkah Anda membayangkan cara memecahkan persandian ini? Serangan (attack) apa yang dapat Anda lakukan? Ternyata Al Kindi menemukan cara untuk memecahkan *Caesar cipher* ini.

<sup>6</sup> Ada situs [www.rot13.com](http://www.rot13.com) yang dapat kita gunakan untuk melakukan enkripsi dan dekripsi dengan pergeseran 13 huruf itu.

Kelemahan dari *monoalphabetic cipher* adalah huruf yang sama digantikan oleh huruf pasangannya dan tetap seperti itu. Serangan yang dilakukan oleh Al Kindi adalah membuat statistik dari kemunculan huruf. Dalam sebuah bahasa tertentu, katakan Bahasa Inggris, ada statistik kemunculan setiap huruf. Dalam Bahasa Inggris, huruf yang paling sering muncul adalah huruf "A". Jika hasil statistik dari teks yang sudah tersandikan huruf yang paling sering muncul adalah huruf "J", maka kita tinggal geser huruf "J" tersebut di bawah huruf "A" dan sisanya tinggal menyesuaikan. Ternyata tidak susah untuk melakukan serangan bukan?

- Ex. 1 — Kemunculan huruf**
1. Buat sebuah program yang membuat statistik kemunculan huruf-huruf untuk bahasa Indonesia, bahasa Inggris, bahasa Jawa, dan bahasa Sunda. Urutkan lima terbesar di setiap bahasa. Catatan: Semakin banyak Anda memberi masukan kepada program statistik Anda, semakin akurat hasil statistiknya.
  2. Apakah metoda ini dapat digunakan untuk bahasa-bahasa yang tidak menggunakan karakter Roman (seperti bahasa Arab, Cina, India, Thailand, dan sejenisnya)?

Dalam sejarah persandian, keberadaan orang yang membuat algoritma sandi (*code maker*) akan berseteru dengan orang yang berusaha untuk memecahkannya (*code breaker*). Dalam contoh di atas, Al Kindi menjadi *code breaker* yang memecahkan kode *Caesar cipher*.

### Struktur Sistem Kriptografi

Secara umum ada tiga komponen utama dari kriptografi, yaitu *plain text*, *ciphertext*, dan algoritma serta kunci yang digunakan<sup>7</sup>.

- *Plain text* adalah data (teks, pesan, *message*) asli yang belum diproses. Meskipun disebut *plain text*, sesungguhnya data asli tidak harus berupa teks (ASCII). *Plain text* dapat juga berupa berkas biner.
- *Ciphertext* adalah data yang dihasilkan dari proses enkripsi. *Ciphertext* dapat berbentuk berkas biner atau ASCII. Perlu diasumsikan bahwa penyerang kemungkinan dapat mengakses *ciphertext*.
- Algoritma dan kunci merupakan *black box* yang memproses *plain text* menjadi *ciphertext*. Algoritma diasumsikan diketahui oleh penyerang, tetapi kunci tidak diketahui.

Hubungan antara ketiga komponen tersebut dapat dirumuskan sebagai berikut. *Ciphertext c* merupakan hasil operasi enkripsi (*en-*

<sup>7</sup> Pada awalnya, kunci melekat dengan algoritma. Namun kemudian, kunci dipisahkan dari algoritma. Kekuatan dari sebuah sistem kriptografi bergantung kepada kerahasiaan dari kuncinya, bukan dari kerahasiaan algoritmanya.

*crypton*)  $E$  dengan kunci  $k$  terhadap pesan  $m$ . *Ciphertext* ini yang nanti dikirimkan kepada pihak yang dituju.

$$c = E_k(m) \quad (1)$$

Penulisan proses enkripsi dapat juga dilakukan seperti berikut.

$$c = \text{ENKRIP}(k, m) \quad (2)$$

Di sisi sebaliknya, yaitu di sisi penerima, pesan (*plain text*)  $m$  diperoleh dari hasil proses dekripsi (*decryption*)  $D$  dengan kunci  $k$  terhadap *ciphertext*  $c$ .

$$m = D_k(c) \quad (3)$$

Penulisan dapat dilakukan dengan notasi berikut.

$$m = \text{DEKRIP}(k, c) \quad (4)$$

Jika kita menggunakan algoritma kriptografi kunci privat, kunci ( $k$ ) yang sama digunakan untuk proses enkripsi dan dekripsi. Untuk algoritma kriptografi yang berbasis kunci publik, kunci yang digunakan untuk proses enkripsi dan dekripsi berbeda bergantung kepada proses yang dilakukan (apakah untuk enkripsi atau penandatanganan).

### *Kriptografi Kunci Privat*

Kriptografi kunci privat adalah jenis kriptografi yang paling banyak dikenal. Pada sistem kriptografi ini, ada satu kunci yang digunakan untuk mengunci dan membuka. Itulah sebabnya sistem ini dikenal juga dengan istilah kriptografi **simetrik**. Kunci yang digunakan harus dirahasiakan sehingga kriptografi ini disebut kriptografi kunci privat.

Ada banyak algoritma yang mengimplementasikan kriptografi kunci privat, antara lain DES, Blowfish, dan AES. (Penjelasan rinci mengenai algoritma-algoritma ini akan dibahas secara terpisah.) Algoritma-algoritma ini umumnya sangat cepat dalam operasinya.

Salah satu kesulitan dari pengoperasian sistem kriptografi kunci privat adalah dalam hal distribusi kunci (*key distribution*). Sebagai contoh, jika *Alice* ingin mengirim pesan kepada *Bob*, maka mereka berdua memiliki sebuah kunci yang sama. Jika *Alice* ingin mengirim pesan ke orang lain, katakan *Charlie*, maka mereka berdua memiliki kunci sendiri (*Alice-Charlie*) yang berbeda dengan kunci *Alice-Bob*. Demikian pula jika *Bob* dan *Charlie* ingin berkomunikasi maka mereka memiliki kunci sendiri (*Bob-Charlie*). Jika kita teruskan dengan pihak-pihak lain, maka jumlah kunci yang dibutuhkan akan

meledak secara eksponensial sesuai dengan penambahan jumlah pengguna ( $n$ ).

$$\text{numkeys} = \frac{(n)(n-1)}{2} \quad (5)$$

Untuk jumlah pengguna yang sedikit, misal puluhan orang, maka jumlah kunci yang beredar tidak terlalu banyak. Begitu jumlah pengguna sangat banyak, maka jumlah kunci menjadi sangat besar seperti dapat dilihat pada tabel 1. Bayangkan jumlah pengguna internet di dunia ini. Berapa banyak kunci yang harus dipersiapkan jika semuanya akan saling berkomunikasi satu dengan lainnya?

$n$	Jumlah kunci
10	45
100	4.950
1000	499.500
10.000	49.995.000
100.000	4.999.950.000

Table 1: Jumlah Kunci

Meskipun ada masalah distribusi kunci, sistem kriptografi kunci privat ini yang paling baik kinerjanya (performance) sehingga dia sangat dibutuhkan.

### *Kriptografi Kunci Publik*

Pada sistem kriptografi kunci publik, ada dua kunci yang akan digunakan. Setiap pelaku akan memiliki sepasang kunci (**kunci publik** dan **kunci privat**) yang saling berhubungan. Jika sebuah pesan dikunci dengan kunci publik, maka dia hanya dapat dibuka oleh kunci privat pasangannya. Demikian pula jika sebuah pesan dikunci oleh kunci privat, maka dia hanya dapat dibuka oleh kunci publik pasangannya. (Jangan bingung. Baca berulang kali. Nanti akan dijelaskan dengan cara lain lagi.)

Sistem ini sering juga disebut **kriptografi asimetrik**, karena kunci yang dipakai untuk mengunci berbeda dengan kunci untuk membuka. Asimetrik.

Sesuai dengan namanya, kunci publik boleh diketahui oleh umum dan disimpan di tempat publik. Sementara itu, kunci privat hanya boleh diakses oleh pemiliknya. Selama-lamanya kunci privat ini tidak boleh terbuka. Jika kunci privat ini tercuri, maka identitas kita juga tercuri.

Mari kembali kita ambil contoh komunikasi antara *Alice* dan *Bob*. Masing-masing pelaku memiliki sepasang kunci. *Alice* memiliki kunci publik  $KA_{pub}$  dan kunci privat  $KA_{priv}$ . Demikian pula *Bob* memiliki kunci publik  $KB_{pub}$  dan kunci privat  $KB_{priv}$ . Jika *Alice* ingin



mengirimkan pesan  $m$  kepada *Bob*, maka *ciphertext*  $c$  merupakan hasil enkripsi dengan menggunakan kunci publik *Bob*.

$$c = E_{KB_{pub}}(m) \quad (6)$$

Di sisi penerima, *Bob*, akan menerima cipertext  $c$ . Untuk mengembalikannya ke pesan semula, dilakukan proses dekripsi ( $D$ ) dengan kunci privatnya sebagai berikut. Perlu diingat bahwa hanya *Bob* yang dapat melakukan hal ini karena hanya dia yang memiliki kunci privatnya. *Alice* sebagai pengirim pun sudah tidak dapat membuka kembali pesan yang dia kirimkan.

$$m = D_{KB_{priv}}(c) \quad (7)$$

Yang menarik dari proses ini adalah untuk mengirimkan pesan bersandi ke *Bob*, sang pengirim (*Alice*) hanya perlu mencari kunci publik *Bob*. Dia tidak perlu bertukar kunci sebelumnya. Masalah *key distribution* terpecahkan dengan cara ini.

Jumlah kunci yang beredar di sistem juga tidak meledak sebagai mana terjadi pada kriptografi kunci privat. Jika jumlah pengguna adalah  $n$ , maka jumlah kunci adalah:

$$numkeys = 2n \quad (8)$$

Beberapa contoh algoritma kriptografi kunci publik yang terkenal antara lain RSA<sup>8</sup> dan Elliptic Curve Cryptosystem (ECC). Sayangnya algoritma-algoritma ini memiliki komputasi yang cukup tinggi sehingga membutuhkan waktu yang jauh lebih lama dalam memproses data. (Akan dibahas kemudian.) Akibatnya, algoritma kriptografi kunci publik kurang disukai atau digunakan secukupnya saja.

<sup>8</sup> Nama RSA ini berasal dari singkatan nama penemunya, yaitu Ron Rivest, Adi Shamir, dan Len Adleman

### *Kriptografi Hybrid*

Seperti dikemukakan sebelumnya, kriptografi kunci privat memiliki algoritma yang relatif cepat tetapi ada masalah pada distribusi kunci. Sementara itu kriptografi kunci publik memiliki kelemahan komputasinya yang tinggi. Salah satu solusi yang menarik adalah menggabungkan kedua sistem sehingga menjadi kriptografi (kunci) hybrid.

Pada pendekatan ini, enkripsi (dan dekripsi) akan menggunakan kriptografi kunci privat tetapi dengan kunci yang dibuat sesaat (*session key*) dan kunci sesi inilah yang dipertukarkan dengan menggunakan algoritma kriptografi kunci publik. Ukuran kunci sangat kecil sehingga biaya (*cost*) untuk melakukan enkripsi dengan kunci publik menjadi kecil.

*Kualitas Sistem Kriptografi*

Kebagusan sebuah sistem kriptografi bergantung kepada kerahasiaan kuncinya bukan kepada kerahasiaan dari algoritmanya. Algoritma yang baik adalah algoritma yang diterbitkan untuk umum dan dievaluasi bersama-sama. Banyak orang yang heran atau tidak percaya kepada hal ini. Ambil contoh algoritma RSA yang dibuat pada tahun 1970-an. Algoritma ini tersedia dan dapat dievaluasi. Sampai sekarang belum ditemukan kelemahan dari algoritma ini. Jika ada seseorang atau sebuah vendor yang tidak mau membuka algoritmanya, maka tingkat keamanannya pantas untuk diragukan.

# *PGP / Gnu Privacy Guard*

Pretty Good Privacy (PGP) pada awalnya adalah aplikasi yang dapat digunakan pengguna untuk menggunakan kriptografi di berbagai aplikasi dengan lebih mudah. Pengembangan selanjutnya PGP menjadi bagian dari *public key infrastructure*.

## *Sejarah*

[... more to be written ...]

Gnu Privacy Guard (GPG) merupakan implementasi dari PGP yang bersifat terbuka. (Catatan: Singkatan dari GPG ini merupakan guyonan terhadap PGP.) Bab ini akan membahas lebih banyak tentang GPG, meskipun konsep yang sama dapat juga diterapkan pada PGP jika Anda menggunakan produk PGP yang komersial.

Dalam buku ini, kita akan menggunakan GPG versi *command line interface*, yaitu dengan mengetikkan perintah “gpg” di program terminal atau CMD.exe. Ada banyak program *GUI* dari GPG ini. Silahkan gunakan manual terkait dengan program-program tersebut. Prinsipnya masih tetap sama.

## *Menggunakan Gnu Privacy Guard, gpg*

Awal dari penggunaan GPG adalah membuat pasangan kunci publik dan privat. Hal ini dapat dilakukan dengan menggunakan perintah berikut.

```
gpg --gen-key
```

Perintah di atas akan menanyakan beberapa hal, seperti jenis algoritma yang digunakan (pilih RSA dan RSA), panjang kuncinya (pilih 2048), dan alamat email yang akan digunakan untuk kunci tersebut. Dalam contoh buku ini saya akan menggunakan alamat email “rahard2017@gmail.com”. Gunakan alamat email Anda sebagai penggantinya/

```
$ gpg --gen-key
```

```
gpg (GnuPG/MacGPG2) 2.0.30; Copyright (C) 2015 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
```

```
Please select what kind of key you want:
```

- (1) RSA and RSA (default)
- (2) DSA and Elgamal
- (3) DSA (sign only)
- (4) RSA (sign only)

```
Your selection? 1
```

```
RSA keys may be between 1024 and 4096 bits long.
```

```
What keysize do you want? (2048)
```

```
Requested keysize is 2048 bits
```

```
Please specify how long the key should be valid.
```

- 0 = key does not expire
- <n> = key expires in n days
- <n>w = key expires in n weeks
- <n>m = key expires in n months
- <n>y = key expires in n years

```
Key is valid for? (0) 3m
```

```
Key expires at Wed May 31 10:30:37 2017 WIB
```

```
Is this correct? (y/N) y
```

```
GnuPG needs to construct a user ID to identify your key.
```

```
Real name: Budi Rahardjo
```

```
Email address: rahard2017b@gmail.com
```

```
Comment:
```

```
You selected this USER-ID:
```

```
"Budi Rahardjo <rahard2017@gmail.com>"
```

```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit?
```

Setelah proses *key generation* selesai, kunci publik dapat diekspor dengan menggunakan perintah berikut. Gantikan “rahard2017@gmail.com” dengan alamat email Anda.

```
gpg --export --armor rahard2017@gmail.com > kunci-public.asc
```

Akan dihasilkan berkas “kunci-public.asc”. Tanpa perintah *redirect output* yang menggunakan “>” itu, hasilnya hanya akan ditampilkan di layar saja. Tentu saja Anda dapat menggunakan nama berkas lainnya. Berikut ini adalah contoh beberapa baris pertama dari berkas ASCII hasil ekspor kunci tersebut. (Tampilan bisa panjang sekali, bergantung dari panjang kunci yang digunakan.)

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
Comment: GPGTools - https://gpgtools.org
```

```
mQENBFiiijUBCAD3ZIxjUQyDtcGTmHs2iU+2aS+MMp2w+codVRrqLfVI+V8S+TmP
WGe2Gu4pTLovGzZ2NGWvKACZ0A+BL97e9K6QN6+UIYRrFdxfojQ4lbwH/W01YSn
J6BsCE1IfX8niRiskdibBhZZcrUCcDa/tgvgXTymoAeZEcMwr48hE0UWu0GoCE2g
TA/kD9yCyorFUKesbyKJMKPt1MJGICUwtggDdXT5arCCafpydLGxok+w4TJjADeX
jeTxDgYSNjCNar9WRhtnG+G/Irbfctj77/9Ppz8j8Dp1NkAwx5P+9AJRNNkwU88c
7FSEz0cpAzgC4VyaL/iX85G8wr6z5kaWNV3FABEBAAGOJEJ1ZGkgUmFoYXJkam8g
PHJhaGFyZDIwMTdAZ21haWwuy29tPokBPQQTaQoAJwUCWKKNQIbAwUJABuvgaUL
...
```

Kunci publik ini yang akan didistribusikan secara publik, misal disimpan di blog Anda. Kunci ini juga dapat disimpan di *key server*.

Untuk melihat informasi mengenai kunci Anda, dapat digunakan perintah berikut:

```
gpg --fingerprint rahard2017
```

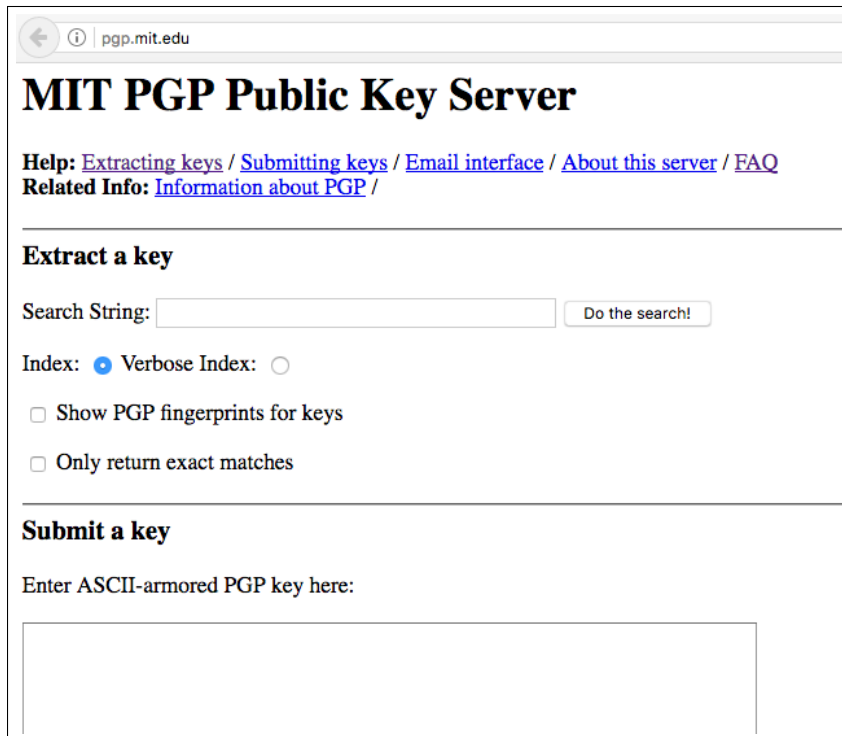
```
pub 2048R/EB6CEB46 2017-02-14 [expires: 2017-03-07]
    Key fingerprint = 810B 2149 3366 E699 F95E 9E49 07E1 BDC5 EB6C EB46
uid [ultimate] Budi Rahardjo <rahard2017@gmail.com>
sub 2048R/C2E83C60 2017-02-14 [expires: 2017-03-07]
```

Perhatikan bahwa kunci publik yang ini memiliki ID “EB6CEB46”. ID ini dapat digunakan untuk bertukar kunci, atau untuk melakukan proses-proses lainnya. Sebagai contoh, kita dapat mengirimkan kunci ini ke server agar dapat dilihat atau dicari oleh orang lain. Keyserver yang akan kita gunakan dalam contoh ini adalah `pgp.mit.edu`.

```
gpg --keyserver pgp.mit.edu --send-keys EB6CEB46
```

Pengiriman kunci dapat juga dilakukan dengan menggunakan web dari situs PGP MIT tersebut sebagaimana dicontohkan dalam gambar berikut. Masukkan teks ASCII kunci Anda ke kolom kunci dan upload.

- |                                     |   |
|-------------------------------------|---|
| <p><b>Ex. 2 — Membuat Kunci</b></p> | <ol style="list-style-type: none"> <li>1. Buat pasangan kunci publik dan kunci privat Anda. Gunakan parameter default saja (RSA dan RSA). Asosisasikan dia dengan alamat email Anda.</li> <li>2. Unggah kunci publik Anda ke salah satu keyserver yang tersedia (misal <code>pgp.mit.edu</code> atau <code>keyserver.cert.or.id</code>)</li> <li>3. Beritahukan teman-teman Anda tentang kunci Anda tersebut. Minta mereka untuk ikut menandatangani kunci publik Anda tersebut.</li> </ol> |
|-------------------------------------|---|



pgp.mit.edu

## MIT PGP Public Key Server

**Help:** [Extracting keys](#) / [Submitting keys](#) / [Email interface](#) / [About this server](#) / [FAQ](#)  
**Related Info:** [Information about PGP](#) /

---

### Extract a key

Search String:

Index: ☒ Verbose Index: ☐

☐ Show PGP fingerprints for keys  
☐ Only return exact matches

---

### Submit a key

Enter ASCII-armored PGP key here:

Figure 3: Key server pgp.mit.edu

### Enkrip Untuk Sebuah Alamat Email

Salah satu fungsi dari PGP/GPG adakan mengirim pesan secara rahasia. Pesan tersebut dienkrip dengan menggunakan kunci publik (alamat email) tujuan. Sebagai contoh, kita akan mencoba mengirimkan pesan rahasia ke rahard2017@gmail.com.

Langkah pertama tentunya kita harus mendapatkan kunci publik dari akun email rahard2017 tersebut. Cari di salah satu keyserver, unduh, dan masukkan ke gantungan kunci kita.

```
gpg --keyserver hkp://pgp.mit.edu --search-keys rahard2017
```

Dalam contoh ini, kita akan menggunakan berkas “dokumen.txt” yang berisi pesan dalam bentuk *plain text*. Tentu saja PGP/GPG tidak membatasi jenis pesan (berkas). Dalam contoh ini saja kita akan menggunakan berkas teks yang berisi teks ASCII. Secara umum berkas biner (MS Word, lagu MP3, film MP4, dan seterusnya) dapat juga diproses. Isi berkas dokumen.txt tersebut adalah sebagai berikut.

Ini adalah dokumen untuk percobaan menggunakan PGP/GPG.  
 Untuk tanda tangan, isinya tidak harus disembunyikan.  
 Selamat mencoba.

Perintah untuk mengenkripsi dengan menggunakan kunci publik tujuan (rahard2017@gmail.com) adalah seperti dicontohkan di bawah ini. Perintah tersebut akan menghasilkan berkas “dokumen.pgp”.

```
gpg --output dokumen.pgp --encrypt --recipient rahard2017@gmail.com dokumen.txt
```

Berkas “dokumen.pgp” yang dihasilkan dari proses enkripsi di atas berupa berkas biner. Jika kita ingin menghasilkan berkas dalam bentuk ASCII, maka gunakan opsi “-armor” (-a) seperti perintah di bawah ini.

```
gpg -a --output dokumen.pgp --encrypt --recipient rahard2017@gmail.com dokumen.txt
```

Jika kita lihat berkas “dokumen.pgp”, isinya adalah seperti berikut.

```
-----BEGIN PGP MESSAGE-----
Comment: GPGTools - https://gpgtools.org

hQEMay/vbaXC6DxgAQf/Zjk74t1FXTT3abmJUz/w5Z8iHIIIEWZlvMmBdMS7w/2U8
L2NrvvG6G0rsduLTlrybCIAQOGPU9Nq+YOMYJaY3BhiqkCSyYhHYRk04060S3GCA
ONnhiqKiVLJIyfNWdDBtB4k7s8pfM5ngxgeZ6/gH5TDspHrhjrLS65Stn7sr+Nlf
TSuG9p21vr19yL13KBkd2rI5WbNl68/3bRjNkT0JL1PLeMvQOeZiIRXcmropPXIs
rltFRflpd0H0LJn2/xQ5rXr13QRRjzo3SL4i/eYxPlEmpD164aicI8LC+qKgYcc
FShUuTxwxtu7tPYpqEH17jTTd29wZdrXDFHp5TGLTdKtAeMKeGupxvic0aNM46J
S3LBC9fU2bywvmvM77cCr97D0P1rA6WXR2xluRvLhms831NMcpSDuZCPpb8rxKmu
Xk2WFThfV5rq0I1kyP2Kc1g+OcgNjKazFXe5MQRIOpJ/MRwrekAo3Dvp3TefwMYu
R3LYwCi8VstPpnH9rcFyoyxsqMqLTptUnheISNUmVERtLm3rALtHjf2vyvkr/2JF
Vmu79aqbbFJPJZ0gNmK=
=D+3L
-----END PGP MESSAGE-----
```

Berkas “dokumen.pgp” tersebut dapat kita kirimkan (misal melalui email) ke rahard2017@gmail.com. Di sisi penerima (rahard2017), berkas tersebut dapat dia buka dengan menggunakan kunci privatnya. Ketika menggunakan kunci privat, biasanya pengguna ditanyakan *passphrase* untuk mengakses (membuka) kunci privat tersebut karena biasanya kunci privat dilindungi dengan *passphrase* tersebut.

```
gpg --decrypt dokumen.pgp

You need a passphrase to unlock the secret key for
user: "Budi Rahardjo <rahard2017@gmail.com>"
2048-bit RSA key, ID C2E83C60, created 2017-02-14 (main key ID EB6CEB46)

gpg: encrypted with 2048-bit RSA key, ID C2E83C60, created 2017-02-14
"Budi Rahardjo <rahard2017@gmail.com>"
```

Ini adalah dokumen untuk percobaan menggunakan PGP/GPG.  
Untuk tanda tangan, isinya tidak harus disembunyikan.  
Selamat mencoba.

Seperti dilihat pada contoh di atas, teks aslinya ditampilkan di layar. Jika kita ingin menyimpannya ke dalam sebuah berkas, maka dapat kita gunakan opsi “-output namaberkas” ketika menjalankan perintah `gpg` tersebut di atas.

Perlu diingat kembali bahwa hanya orang yang memiliki kunci privat, yaitu `rahard2017@gmail.com`, yang dapat membuka berkas tersebut. Jika berkas tersebut dicoba untuk dibuka dengan kunci lain, maka dia tidak dapat menghasilkan isi yang sama.

### *Tanda Tangan Dokumen*

Salah satu manfaat penggunaan PGP/GPG adalah tanda tangan digital (sign). Dalam contoh berikut ini kita akan mendatangi berkas “dokumen.txt”. (Kita dalam contoh ini adalah “rahard2017@gmail.com”.)

```
gpg -u rahard2017@gmail.com --output dokumen.sig --sign dokumen.txt
```

```
You need a passphrase to unlock the secret key for
user: "Budi Rahardjo <rahard2017@gmail.com>"
2048-bit RSA key, ID EB6CEB46, created 2017-02-14
```

Seperti sebelumnya, berkas yang dihasilkan (`dokumen.sig`) dalam format biner. Untuk menghasilkan berkas ASCII, dapat digunakan “-a”.

```
gpg -a -u rahard2017@gmail.com --output dokumen.sig --sign dokumen.txt
```

Isi berkas “dokumen.sig” yang sudah di-armor-kan seperti ini. Sebagai catatan, isi dokumen dan tanda tangan tercampur dan terarmor-kan. Kerugian cara ini adalah isi dari dokumen tidak dapat terlihat secara langsung.

```
-----BEGIN PGP MESSAGE-----
```

```
Comment: GPGTools - https://gpgtools.org
```

```
owGbwMvMwMXI/nDv0dc5r90Y10xI4k7Jzy7NTc3TK6koidipudUzL1MhMSUxJzFD
ASqjUJpXUpqtUJBalJyflJiYpWAUS08vzUvMBrID3AP03QPc9bhCwYpKEvNSEkFk
emKejkJmcWZeJZCbmkZYrZCRWFRarJCSWZyam1SaV5kJ1K3HFZyak5ibWAIyEms4
HlcnowwLAYMXAxsre8g1DFycAjDXcoix//d7pfhQ+pbBxM4ilx/pi05400kFsfs2
TDDSVY/Zec58cp/zm9TgDczKn+71Kkqvs/4fNNvRuUr208VdwowLPTliFpR+02/L
0cR/KDTuie6h9Xlq3bXTSg6kpeoxblZ323ra4jDfvvs/9RjeSBYS0cp4+T/c81B6
YUepNt00hLjZZeGxR63yNqxbu7ZX9djt4p6a7/JTbJKepvo3XIItl08x0xVDt3LIv
oRc/tc7667Wqskac95zr1E6xREHL5eEKgnkByy7arEnL0Ge++bNd4ET4NOV3FYqP
```



```
L875kT8v8PeVPdqr+SrENG7YNL6+PWX0vNUR6ruMVdS0hzEoeyn4P2EWX6J1dVL/
n1dzXzdrAgA=
=VRfS
-----END PGP MESSAGE-----
```

Ada cara untuk memisahkan isi (teks) dan tanda tangannya. Perintah yang digunakan adalah “--clearsign”.

```
gpg -u rahard2017@gmail.com --output dokumen.sig --clearsign dokumen.txt
```

Hasilnya adalah sebuah berkas yang isinya adalah sebagai berikut. Perhatikan bahwa isi dan tanda tangan terpisah, meskipun masih dalam sebuah berkas.

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

Ini adalah dokumen untuk percobaan menggunakan PGP/GPG.
Untuk tanda tangan, isinya tidak harus disembunyikan.
Selamat mencoba.
-----BEGIN PGP SIGNATURE-----
Comment: GPGTools - https://gpgtools.org

iQEcbAEBCgAGBQJYUStHAAoJEafhvcXrb0tG1pkiALaFxiXK0bM7p/ZTMpznCIHN
GpyR9Q6QQEi+9+0loHdw1TqMEskJ3HxWOnegupMueiIp00IaEd8qCraLSfRF3G1f
gBxl837rXnTjDIYABN9T6f5GrZHyEWXehwDRDuLWsb9v85pNd/RUbQ8L7N/s4SU1
NgyPbHKhal+ob047mHE1zf9lvpuRfhWM3ztr80JN6MWTgb+NYpc0b9YuyqSRiDZO
y3fvY3iRexMjLI2dxhX0TWG4IM5ouwbFtheJML1Lssh/pn1KZwicnWGAouWhQT0Y
hGR8zSoy4oeS/hlUmdAu3bse73pcWeybCtYLwvXSBvg7W67sCOERmIfZxxxtq08=
=/uVR
-----END PGP SIGNATURE-----
```

Untuk melakukan verifikasi apakah dokumen tersebut terjamin integritasnya (isinya tidak berubah dan yang menandatangani benar), gunakan perintah “verify”.

```
gpg --verify dokumen.sig
```

Seringkali ada kebutuhan untuk memisahkan berkas dari tanda tangan dan dokumennya. Sebagai contoh, jika dokumen yang ingin kita tandatangani adalah berkas biner (dokumen *word processor*, audio, video, dan sejenisnya), maka tanda tangan dan dokumennya harus dipisah.

```
gpg -u rahard2017@gmail.com --output dokumen.sig --detach-sig dokumen.txt
```

Untuk melakukan verifikasi bahwa dokumen tersebut benar (isinya benar dan yang menandatangani benar juga) jika dokumen-nya dipisah adalah sebagai berikut.

```
gpg --verify dokumen.sig dokumen.txt
```

- Ex. 3 — Verifikasi**
1. Menurut Anda, apakah tanda tangan (signature) akan tetap sama untuk setiap orang meskipun dokumen yang ditandatangani berbeda-beda? Ataukah untuk dokumen yang berbeda, tanda tangan digital dari dokumen tersebut akan berbeda?
  2. Verifikasi dokumen dengan data yang benar. Maksudnya jangan ubah berkas “dokumen.sig” dan “dokumen.txt”.
  3. Verifikasi dokumen dengan data yang sudah tercemar. Edit berkas dokumen.txt sehingga berbeda dengan aslinya. Coba verifikasi. Tunjukkan bahwa proses verifikasi gagal.

### *Tools*

Perintah-perintah dalam contoh pada bagian sebelumnya dilakukan dengan menggunakan *command line*. Ada beberapa *tools* (biasanya memiliki *graphical user interface*) yang dapat membantu untuk mempermudah operasi. *Tools* yang tersedia bergantung kepada sistem operasi yang Anda gunakan.

### *Web of Trust*

Keamanan dari PGP/GPG ini berbasis *web of trust*. Bagaimana kita dapat mempercayai bahwa kunci publik tersebut milik dari “Budi Rahardjo” dengan alamat email tersebut (rahard2017@gmail.com)? Boleh jadi ada seseorang yang dengan sengaja memalsukan identitas tersebut.

Proses verifikasi dilakukan oleh orang lain dengan menandatangani kunci tersebut dan kemudian data tersebut - kunci yang sudah ditandatangani - diunggah kembali ke keyserver.

## Keamanan Sistem Email

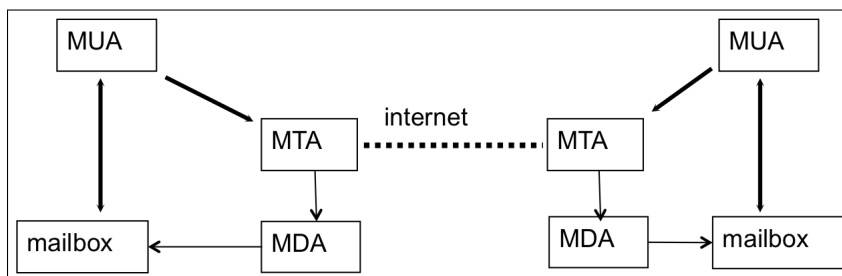
*Electronic mail* (email<sup>9</sup>) masih merupakan salah satu aplikasi yang paling populer di internet. Bahkan alamat email digunakan sebagai identitas pengguna di internet. Jika Anda mendaftar ke sebuah layanan, email digunakan sebagai identitas.

Beberapa masalah keamanan terkait dengan sistem email antara lain:

- disadap (*intercept*);
- dipalsukan (*forgery*);
- disusupi (*intrude*);
- digunakan untuk spamming;
- mailbomb;
- mail relay.

### Komponen Sistem Email

Sebelum membahas masalah keamanan tersebut ada baiknya kita melihat komponen dari sebuah sistem mail. Pemahaman tentang komponen ini dibutuhkan untuk memahami potensi sumber masalah keamanan email. Sebuah sistem email terdiri dari beberapa komponen; *mail user agent* (MUA), *mail transfer agent* (MTA), dan *mail delivery agent* (MDA). (Lihat Gambar 4.)



<sup>9</sup> Dalam bahasa Indonesia sudah ada istilah **surel** untuk email ini. Dalam buku ini saya masih menggunakan istilah email.

Figure 4: Topologi Sistem Email

MUA adalah komponen yang berhubungan dengan pengguna. Biasanya MUA adalah yang kita sebut program email. Contoh dari MUA antara lain adalah Thunderbird, Outlook, Mac Mail.app, mutt, UNIX mail, pine, dan masih banyak lagi. (Daftar ini sering berubah.) Pengguna menggunakan MUA untuk membuat (*compose*) dan membaca email.

MTA adalah komponen yang bertugas untuk mengirimkan dan menerima email. Dia adalah “pak pos”. MTA menerima berkas email dari MUA dan meneruskannya ke MTA lainnya dan seterusnya sampai ke MTA yang dituju. Contoh dari MTA antara lain adalah postfix, sendmail, qmail, Exchange, exim, dan sejenisnya. MTA biasanya adalah urusan dari administrator.

MDA adalah komponen yang bertugas untuk menyimpan email yang datang ke mailbox pengguna. Dahulu, MDA ini menjadi bagian dari MTA, tetapi kemudian dipisahkan karena pemisahan role agar lebih aman. MDA harus menambahkan email yang baru masuk ke mailbox pengguna. Untuk itu MDA harus memiliki hak untuk menulis ke mailbox tersebut, dengan kata lain MDA harus dijalankan dengan hak admin atau *super user / root*. Sementara itu MTA tidak harus dijalankan sebagai admin.

Skenario yang terjadi adalah sebagai berikut. Seorang pengguna (A) membuat email dengan menggunakan MUA. Setelah email selesai dibuat, email diberikan kepada MTA untuk disampaikan kepada MTA penerima (B). Kadang MTA yang dituju tidak langsung dapat diakses tetapi melalui MTA lainnya dahulu. Sesampainya di MTA tujuan, email diberikan ke MDA untuk ditambahkan ke mailbox penerima (B). Penerima (B) tidak harus online ketika email tersebut itu sampai. Ketika penerima (B) akan membaca email, maka dia akan menggunakan MUA untuk mengakses mailbox. Jika dia (B) akan membalas, maka digunakan MUA untuk menuliskan balasannya. Setelah selesai, email balasan diteruskan ke MTA untuk disampaikan ke MTA tujuan (A).

### *Standar Email*

Pengguna email memiliki sistem dan konfigurasi yang bervariasi. Masalah *interoperability* merupakan salah satu aspek yang sangat penting. Untuk itu digunakan RFC (Request For Comments) sebagai standar.

Standar format email didefinisikan oleh RFC 822, “Standard for the format of ARPA Internet text messages.” (RFC ini sudah digantikan oleh RFC 2822, “Internet Message Format.”) Pada prinsipnya email dibagi dua bagian; **header** dan **body**.

- *Header*. Seperti amplop, berisi informasi tentang alamat pengirim yang dituju. Header ini berisi *field* yang nantinya digunakan oleh MTA untuk mengirimkan ke tujuan.
- *Body*. Isi surat. Body dipisahkan dari header dengan satu baris kosong.

Contoh dari (format) email dapat dilihat sebagai berikut. Perhatikan bahwa header dan body dipisahkan oleh satu baris kosong. Contoh ini tentu saja merupakan simplifikasi dari format email sesungguhnya.

```
From: Budi Rahardjo <budi@cert.or.id>
To: br@paume.itb.ac.id
Subject: Kelas EL776 hari ini

Kelas hari ini dibatalkan dan akan digantikan dengan
hari lain.

-- budi
```

Ada standar *field* di header yang mudah terlihat oleh pengguna, yaitu *From*, *To*, *Subject*, *Cc*, dan *Bcc*. Padahal ada banyak *field-field* lain yang biasanya terdapat dalam email. Berikut ini contoh header yang lebih komplis lagi. (Mengenai masing-masing *field* di header tersebut akan dibahas lebih lanjut.)

```
Received: from nic.cafax.se (nic.cafax.se [192.71.228.17])
  by alliance.globalnetlink.com (8.9.1/8.9.1) with ESMTTP id QAA31830
  for <budi@alliance.globalnetlink.com>; Mon, 26 Mar 2001 16:18:01 -0600
Received: from localhost (localhost [[UNIX: localhost]])
  by nic.cafax.se (8.12.0.Beta6/8.12.0.Beta5) id f2QLSJVM018917
  for ietf-provreg-outgoing; Mon, 26 Mar 2001 23:28:19 +0200 (MEST)
Received: from is1-55.antd.nist.gov (is1-50.antd.nist.gov [129.6.50.251])
  by nic.cafax.se (8.12.0.Beta5/8.12.0.Beta5) with ESMTTP id f2QLSGiM018912
  for <ietf-provreg@cafax.se>; Mon, 26 Mar 2001 23:28:17 +0200 (MEST)
Received: from barnacle (barnacle.antd.nist.gov [129.6.55.185])
  by is1-55.antd.nist.gov (8.9.3/8.9.3) with SMTP id QAA07174
  for <ietf-provreg@cafax.se>; Mon, 26 Mar 2001 16:28:14 -0500 (EST)
Message-ID: <04f901c0b63b$16570020$b9370681@antd.nist.gov>
From: "Scott Rose" <scottr@antd.nist.gov>
To: <ietf-provreg@cafax.se>
Subject: confidentiality and transfers
Date: Mon, 26 Mar 2001 16:24:05 -0500
MIME-Version: 1.0
X-Mailer: Microsoft Outlook Express 5.50.4133.2400
```

Sender: owner-ietf-provreg@cafax.se  
 Precedence: bulk

Nama *field* biasanya berupa satu kata atau jika lebih dari satu kata disambungkan dengan tanda garis (dash) dan diakhiri dengan tanda titik dua (:). Isi dari *field* berupa teks. (Panjang dari teks ini biasanya kurang dari 80 karakter karena merupakan bawaan dari sistem-sistem jaman dahulu.) Jika teks membutuhkan lebih dari satu baris, maka lanjutannya dapat diletakkan di bawahnya dengan masuk (indent) menggunakan spasi atau tab.

Selain *field* yang sudah standar, kita juga dapat membuat *field* sendiri. Aturannya adalah nama *field* tersebut dimulai dengan "X-". Misalnya kita dapat membuat *field* "X-catatan:" yang dapat kita gunakan untuk menuliskan catatan kecil. Oleh MTA dan MUA, *field* tambahan ini akan diabaikan. Pada contoh sebelumnya dapat dilihat sebuah contoh, yaitu "X-Mailer:".

X-Mailer: Microsoft Outlook Express 5.50.4133.2400

Pemahaman fungsi dari masing-masing *field* itu dibutuhkan ketika kita akan melakukan *forensic* terhadap email, karena seringkali ada email palsu atau email-email ancaman. Sebagian besar orang memang tidak tahu keberadaan *field-field* tersebut.

Salah satu *field* yang penting adalah "Message-ID". Setiap email memiliki kode Message-ID yang berbeda-beda (unik). Message-ID ini dibuat oleh MTA ketika dia mengirimkan email (yang pertama kali). Identitas ini dapat membantu kita dalam melakukan pelacakan. Sebagai contoh, kita bisa mencari (bertanya) apakah di berkas catatan (log) di server mail pengirim ada Message-ID tersebut. Ini sangat bermanfaat dalam penyidikan.

### *Penyadapan Email*

Email sering dianalogikan dengan surat, padahal email lebih cocok dianalogikan dengan kartu pos karena dia terbuka dan dapat dibaca. Kartu pos dapat dibaca oleh orang-orang yang dilewati oleh kartu pos tersebut, misalnya oleh pak pos. Demikian pula dengan email. Email dapat dibaca pada setiap jalur yang dilewati oleh email tersebut, misalnya pada MTA.

Pengiriman email antar MTA pada awalnya menggunakan protokol SMTP, Simple Mail Transfer Protokol (RFC 821). Sesuai dengan namanya, SMTP merupakan protokol yang sederhana yang tidak memiliki pengamanan terhadap aspek penyadapan. SMTP menggunakan protokol TCP dan berjalan di atas port 25. Penyadapan port 25 ini dapat dilakukan dengan menggunakan program *sniffer* seperti *tcpdump* dan *wireshark*.

Penyadapan email ini dapat dilakukan pada jalur yang dilewati oleh email. Jadi kita tidak dapat menangkap email yang berada pada segmen jaringan yang berbeda.

Penggunaan *tcpdump* untuk menyadap email memang dapat dilakukan, tetapi data yang diperoleh masih mentah dan harus disambung-sambungkan lagi untuk mendapatkan email yang tersusun lengkap (dan berurutan). Ada program *mailsnarf* yang dapat melakukan hal tersebut<sup>10</sup>.

```
unix$ mailsnarf > sniffed-mails
```

Berkas hasil tangkapan *mailsnarf* dapat dibuka dengan menggunakan MUA (program email) yang biasa kita gunakan.

Ada beberapa cara untuk mengamankan email kita dari penyadapan. Yang pertama, dari sisi pengguna, kita dapat melakukan enkripsi terhadap body email sehingga meskipun email kita disadap tetapi sang penyadap tidak dapat memperoleh isinya. Cara ini dapat dilakukan dengan mudah jika kita menggunakan PGP atau GPG.

Cara yang lebih baik adalah dengan menggunakan protokol yang lebih aman. SMTPS, atau SMTP Secure, merupakan implementasi SMTP yang menggunakan enkripsi. Pada prinsipnya klien (MTA) tetap menggunakan protokol SMTP tetapi pada layer di bawahnya digunakan SSL atau TLS untuk menghindari penyadapan. SMTPS biasanya menggunakan port 587.

<sup>10</sup> Program mailsnarf terdapat dalam paket *dsniff*. Sayangnya paket program ini sudah tidak dikembangkan lagi. Di berbagai implementasi dari mailsnarf nampaknya tidak dapat menangkap email lagi. Untuk merakit ulang program ini dibutuhkan skill yang lumayan.





## *Bibliography*

- [1] David Kahn. *Codebreakers*. Scribner, 1967.
- [2] Steven Levy. *Crypto: How the Code Rebels Beat the Government Saving Privacy in the Digital Age*. Penguin Books, 2001.
- [3] Budi Rahardjo. *Keamanan Perangkat Lunak*. PT Insan Infonesia, 2016.