# Analisis Keamanan pada *Pretty Good Privacy* (PGP)

Andri Tanoto – NIM: 13503078

*Program Studi Teknik Informatika, Institut Teknologi Bandung*
*Jl. Ganesha 10, Bandung*
E-mail : if13078@students.if.itb.ac.id

## Abstrak

*Pretty Good Privacy* (PGP) adalah suatu pogram komputer yang dikembangkan oleh Phil Zimmermann pada pertengahan tahun 1980 yang memungkinkan seseorang untuk saling bertukar pesan melalui email dan juga *file* dengan memberikan perlindungan kerahasiaan berupa enkripsi dan otentikasi berupa *digital signature* (tanda tangan digital). PGP menggunakan kriptografi kunci simetri dan juga kriptografi kunci publik. Oleh karena itu, PGP mempunyai dua tingkatan kunci, yaitu kunci rahasia (simetri), yang disebut juga *session key*, untuk melakukan enkripsi data dan pasangan kunci privat dan kunci publik untuk memberikan *digital signature* dan sekaligus melindungi kunci simetri. Fungsi-fungsi utama pada PGP antara lain untuk melakukan enkripsi dan membuat *digital signature* pada *file*, melakukan dekripsi dan verifikasi pada *file* yang memiliki *digital signature*, dan mengelola koleksi kunci PGP yang dimiliki.

Penggunaan PGP ditujukan untuk melindungi tiga hal sebagai berikut, pertama, privasi, kerahasiaan pada penyimpanan dan transmisi data akan dijamin sehingga hanya orang-orang yang berhaklah yang dapat mengaksesnya, kedua, integritas, jaminan terhadap data agar tidak dimodifikasi tanpa sepengetahuan pemiliknya, dan ketiga, otentikasi, jaminan kepemilikan terhadap data.

Pada makalah ini akan dibahas mengenai *Pretty Good Privacy* (PGP) yang meliputi sejarah perkembangan PGP, cara kerja PGP, eksperimen dengan PGP, algoritma pada PGP, dan analisis keamanan pada PGP.

***Kata kunci**: Pretty Good Privacy* (PGP), enkripsi, kriptografi kunci publik, *digital signature*.

## 1. Pendahuluan

*Pretty Good Privacy* (PGP) dikembangkan pertama kali oleh Phil Zimmermann pada akhir tahun 1980. Pada mulanya, PGP digunakan untuk melindungi surat elektronik atau email dengan memberikan perlindungan kerahasiaan dengan metode enkripsi dan otentikasi berupa tanda tangan digital. Saat ini, PGP tidak hanya ditujukan untuk keamanan email tetapi juga untuk keamanan berbagai file dan program pada komputer personal (PC).

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

PGP tersedia sebagai *freeware* maupun sebagai paket komersil dalam berbagai versi yang dapat beroperasi pada berbagai sistem operasi (Windows, Linux, Mac). Program PGP dapat di-download melalui internet pada situs http://www.pgp.org. PGP terbaru adalah PGP versi 9, tetapi pada makalah ini akan digunakan PGP versi 8 untuk melakukan analisis keamanannya. PGP versi-versi awal menggunakan IDEA sebagai algoritma kunci simetri dan RSA sebagai algoritma kunci publik, sedangkan pada versi-versi terakhir menggunakan algoritma CAST sebagai algoritma kunci simetri serta algoritma Diffie-Hellman sebagai algoritma kunci publik.

PGP pertama kali muncul sebagai aplikasi sederhana untuk melindungi komunikasi antarkomputer dengan menggunakan algoritma RSA dari serangan penyadap. Seiring dengan perkembangan internet, gangguan yang terjadi tidak hanya pada jalur komunikasi saja tetapi juga pada dokumen-dokumen yang terdapat pada komputer pribadi. Oleh karena itu, PGP menambahkan fitur-fitur baru pada aplikasinya, antara lain fitur untuk menghapus dokumen secara aman, enkripsi pada *hard disk*, dan enkripsi pada jaringan. Selain itu, dikembangkan juga antarmuka yang *user-friendly* untuk memudahkan penggunaan PGP. Perubahan juga terjadi pada target pasar pengguna PGP yang semula individu menjadi peusahaan karena hanya perusahaan yang bersedia untuk membayar program pengamanan seperti PGP.

PGP menjadi terkenal dan banyak digunakan karena merupakan program pertama yang menawarkan penggunaan kriptografi secara kuat. Pada awalnya PGP merupakan program yang gratis, pengguna dapar melihat *source code* (kode sumber) dan

mengembangkannya jika diinginkan, akan tetapi pada akhirnya PGP dibeli oleh perusahaan perangkat lunak dan menjadi tidak gratis lagi.

Secara garis besar PGP memiliki tiga fitur utama, yaitu:

1. Fitur untuk melakukan enkripsi dan menandatangani dokumen.
2. Fitur untuk melakukan dekripsi dan verifikasi tanda tangan.
3. Fitur untuk mengelola kunci PGP yang dimiliki oleh pengguna.

Setiap orang yang menggunakan PGP harus menerima kunci publik terlebih dahulu. Kunci publik tersebut didapatkan dengan cara mengirim email kepada rekan yang akan diajak berkomunikasi dengan memanfaatkan program PGP atau dengan terhubung secara langsung ke server yang memegang kunci publik. Setelah kunci publik didapat maka langkah berikutnya adalah melakukan verifikasi terhadap kunci tersebut. Proses verifikasi tersebut dapat dilakukan secara tidak langsung, hal ini merupakan fitur utama dari PGP.

PGP dan kriptografi tentu memiliki beberapa keterbatasan, berikut ini adalah ancaman yang harus diperhatikan oleh pengguna PGP:

1. *Dictionary attack*
   Seringkali pengguna PGP memilih *password* untuk melakukan enkripsi kunci privat yang mudah ditebak, seperti nama kerabat, tanggal lahir, nomor telepon, atau kata umum lainnya. Seorang hacker dapat dengan mudah menebak *password* tersebut dengan cara melakukan *dictionary attack*, yaitu menebak *password* dengan cara mencoba semua kata yang umum digunakan oleh pengguna PGP. Oleh karena itu, dibutuhkan *password* yang sulit ditebak seperti gabungan karakter antara alfabet dan numerik.
2. Penghapusan dokumen secara tidak "bersih"
   Proses menghapus dokumen dari komputer pribadi tidaklah semudah yang dikira. Sistem operasi hanya melakukan penghapusan indeks *file* dari *hard disk*, sedangkan *file* tersebut sebenarnya masih terdapat di sana. Seseorang dapat melakukan pemulihan (*recovery*) terhadap *file* tersebut dengan menggunakan berbagai aplikasi tertentu, salah satunya dengan Norton Disk Doctor. PGP menyediakan subrutin untuk melakukan *overwrite* tempat kosong pada *hard disk* sehingga *file-file* yang sudah dihapus tidak dapat di-*recover* lagi.
3. Virus dan *trojan horse*
   Seseorang dapat menyebarkan virus yang mencatat semua tombol *keyboard* yang ditekan oleh pengguna, program seperti ini biasa disebut *keylogger*. *Keylogger* dapat digunakan untukmencuri *password* kunci privat pengguna PGP. Selain itu, terdapat pula program PGP palsu yang berupa *trojan horse* dan dapat menyebarkan kunci rahasia pengguna. Oleh karena itu, pengguna perlu melengkapi komputer pribadinya dengan program *antivirus* dan juga *firewall*.
4. Ancaman keamanan fisik
   Selain ancaman pada sistem komputer, ancaman fisik seperti pencurian komputer dan data juga perlu diperhatikan. Oleh karena itu diperlukan juga keamanan secara fisik pada komputer milik pribadi dan juga perusahaan.
5. Analisis jaringan
   PGP dapat mencegah seseorang membaca pesan yang dikirimkan, tetapi tidak dapat mencegah seseorang yang melakukan *monitoring* terhadap jaringan komputer. Serangan seperti ini dilakukan dengan menangkap paket-paket data yang dikirim melalui jaringan kemudian melakukan analisis terhadap paket-paket data tersebut.
6. TEMPEST
   TEMPEST merupakan metode untuk mengumpulkan informasi rahasia dengan cara "mendengarkan" pada radiasi yang dipancarkan peralatan elektronik. Pada tahun 1960-an, perlengkapan militer dilengkapi alat untuk melindungi jenis serangan seperti ini. PGP juga menyediakan fitur untuk memilih jenis huruf khusus dengan ujung yang *smooth* dan karakter yang tidak biasa agar radiasi yang dipancarkan melalui monitor komputer tidak terdeteksi oleh serangan TEMPEST.

Bagian berikutya pada makalah ini akan membahas mengenai *Pretty Good Privacy* (PGP) yang meliputi sejarah perkembangan PGP, cara kerja PGP, eksperimen dengan PGP, algoritma pada PGP, dan analisis keamanan pada PGP.

## 2. Sejarah *Pretty Good Privacy* (PGP)

PGP versi 1 pertama kali dibuat pada tahun 1991 oleh Phil Zimmermann, walaupun telah dikembangkan sejak pertengahan tahun 1980-an. Pada mulanya PGP dibuat dengan tujuan untuk mengamankan komunikasi pada *Buletin Board System* dan juga melindungi dokumen dan pesan pribadi. Pada saat itu, PGP merupakan program yang gratis jika digunakan untuk alasan pribadi dan bukan komersial. Program PGP ini didapatkan dengan cara

di-*download* langsung melalui internet berikut dengan *source code* lengkapnya.

Nama "Pretty Good Privacy" diambil dari nama toko kelontong yang terdapat pada suatu kota fiksi dalam cerita radio yang bernama "Ralph's Pretty Good Grocery"

Enkripsi pesan dengan menggunakan PGP menjadi populer di seluruh dunia setelah diluncurkan pertama kalo melalui internet. Pengguna PGP banyak juga yang berasal dari para aktivis sosial dari berbagai belahan dunia yang membutuhkan keamanan saat berkomunikasi.

Dalam waktu yang cukup singkat setelah diluncurkan melalui internet, PGP telah digunakan sampai keluar Amerika Serikat. Oleh karena itu, Phil Zimmermann menjadi target pencarian oleh pemerintah Amerika Serikat karena dianggap telah mengekspor "alat pertahanan" secara ilegal. Sistem kriptografi di Amerika Serikat hanya membolehkan kunci dengan panjang 40 bit, sedangkan panjang kunci PGP selalu lebih besar daripada 128 bit. Oleh karena itu, algoritma enkripsi pada PGP dianggap melanggar aturan tersebut. Akan tetapi, investigasi terhadap Zimmermann pada akhirnya dihentikan tanpa adanya hukuman apapun.

Saat ini, enkripsi PGP tidak lagi dianggap sebagai "alat pertahanan" yang berbahaya dan dilarang untuk diekspor keluar Amerika Serikat kecuali jika diekspor ke 7 negara tertentu serta digunakan oleh sejumlah kelompok dan individu yang berada pada daftar hitam *(blacklist)*.

Pada saat pertentangan tentang PGP sedang berlangsung, Zimmermann mulai mengembangkan versi baru untuk PGP, yang disebut PGP 3. Versi baru ini menawarkan peningkatan keamanan, termasuk struktur sertifikat yang baru dengan sedikit perbaikan pada sertifikat yang digunakan pada PGP 2.x. PGP 3 menggunakan algoritma CAST5 sebagai algoritma kunci simetri dan algoritma DSA dan ElGamal sebagai algoritma kunci asimetri.

Setelah investigasi oleh pemerinah Amerika Serikat terhadap Zimmermann berakhir pada tahun 1996, ia mendirikan perusahaan untuk memproduksi versi baru PGP. Perusahaan baru ini bergabung dengan perusahaan Viacrypt kemudian berganti nama menjadi PGP Incorporated. Perusahaan ini memulai pekerjaannya dengan mengembangkan versi baru PGP berdasarkan pada sistem PGP 3. Lain halnya dengan PGP 2 yang hanya dijalankan melalui baris perintah (*command line*), PGP 3 telah memiliki *library* perangkat lunak yang memungkinkan pengguna untuk menjalankan PGP melalui antarmuka grafis atau GUI (*Graphical User Interface*). Perjanjian awal antara Viacrypt dan Zimmermann adalah sebagai berikut, Viacrypt akan memiliki hak atas versi genap dari PGP dan Zimmermann memiliki versi ganjilnya. Oleh karena itu, Viacrypt membuat versi baru PGP berdasarkan pada PGP 2 yang disebut PGP 4. Untuk menghilangkan kebingungan tentang PGP 3 yang sebenarnya adalah penerus dari PGP 4 maka PGP 3 diubah namanya menjadi PGP5 dan diluncurkan pada 5 Mei 1997.

Enkripsi dengan menggunakan PGP telah menjadi bagian penting di seluruh dunia dan dianggap sebagai sistem kriptografi yang sangat berkualitas. Oleh karena itu, banyak perusahaan ataupun individu yang ingin membuat perangkat lunak yang dapat berjalan bersamaan dengan PGP 5. Zimmermann pun menyadari bahwa diperlukan standar terbuka (*open standard*) untuk enkripsi PGP bagi komunitas kriptografi secara keseluruhan. Pada bulan Juli 1997, PGP Incorporated membuat proposal yang disebut OpenPGP dan mengajukannya kepada IETF. Kemudian tidak lama setelah itu, IETF menerima proposal tersebut dan membuat kelompok kerja (*working group*) OpenPGP. OpenPGP termasuk ke dalam *Internet Standard Track* dan spesifikasi lengkapnya berada pada RFC2440.

Pada bulan Desember 1997, PGP Incorporated diakuisisi oleh Network Associates, Inc. yang kemudian menjadi perusahaan pertama yang secara legal melakukan strategi ekspor dengan mempublikasikan *source code*. Kemudian PGP pun berkembang dengan adanya fitur tambahan seperti enkripsi *disk*, *desktop firewall*, pendeteksian intrusi, dan Ipsec VPN ke komunitas keluarga PGP.

Pada awal tahun 2001, Zimmermann meninggalkan Network Associates, Inc. dan menjadi kriptografer utama untuk Hush Communications yang menyediakan layanan email berbasis OpenPGP, yang disebut Hushmail. Pada bulan Oktober 2001, Network Associates, Inc. menjual seluruh aset PGP, kecuali PGP yang dijalankan dengan *command line*, dan melakukan penundaan untuk pengembangan enkripsi PGP berikutnya.

Pada bulan Agustus 2002, beberapa mantan anggota tim pengembang PGP mendirikan perusahaan dengan nama PGP Corporation dan membeli aset PGP dari Network Associates, Inc. Saat ini Zimmermann menjadi penasihat khusus dan konsultan untuk PGP Corporation. Pada pertengahan tahun 2004 PGP Corporation meluncurkan versi baru PGP yang disebut PGP Command Line yang terintegrasi dengan aplikasi PGP Encryption Platform. Pada tahun 2005 PGP Corporation melakukan akuisisi terhadap perusahaan perangkat lunak Jerman bernama Glueck and Kanja Technology.

Berikut ini adalah produk-produk PGP yang diluncurkan pada rentang waktu antara tahun 2002 sampai dengan 2006:

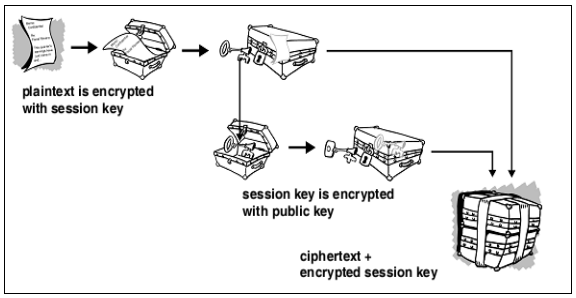
- Tahun 2002
  - PGP 7.2 untuk Mac OS 9
  - PGP Personal dan PGP Freeware
  - PGP 8.0 untuk Macintosh dan Windows
  - *Source code* PGP
- Tahun 2003
  - PGP Desktop 8.0.1DE untuk Windows
  - PGP Desktop 8.0.2
  - PGP Desktop 8.0.3
  - PGP Universal 1.0
  - PGP Universal 1.1
- Tahun 2004
  - PGP Universal 1.2
  - PGP Desktop 8.1
  - PGP Command Line 8.5
  - PGP Universal yang terintegrasi dengan email
  - PGP Software Development Kit (SDK)
- Tahun 2005
  - PGP Universal 2.0
  - PGP Desktop 9.0
  - PGP 9.0.1 Freeware
  - PGP Whole Disk Encryption
  - PGP 9.0.2 untuk negara Jerman
  - PGP 9.0.2 untuk negara Jepang
- Tahun 2006
  - PGP Desktop Home 9.5
  - PGP Desktop Email 9.5
  - PGP Desktop Storage 9.5
  - PGP Desktop Professional 9.5
  - PGP Desktop Enterprise 9.5
  - PGP Command Line 9.5
  - PGP Whole Disk Encryption 9.5
  - PGP Universal Server 9.5
  - PGP Netshare 9.5

## 3. Cara Kerja PGP

PGP mengkombinasikan fitur-fitur terbaik yang terdapat pada kriptografi konvensional dengan kriptografi kunci publik. PGP merupakan sistem kriptografi *hybrid*.

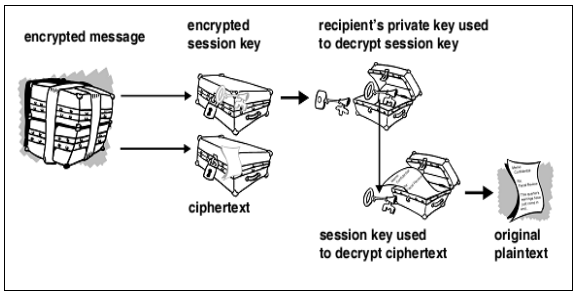
Enkripsi pada PGP menggunakan kriptografi kunci publik dan juga sistem yang menggabungkan kunci publik tersebut dengan identitas pengguna. Versi pertama dari sistem ini memperkenalkan skema *web of trust* yang berbeda dengan sistem X.509 yang menggunakan pendekatan berdasarkan otoritas sertifikat (*authority certificate*). Versi terbaru dari PGP menyediakan kedua alternatif tersebut melalui manajemen server secara otomatis.

Enkripsi email pada PGP menggunakan algoritma enkripsi kunci asimetri dengan pasangan kunci publik-kunci privat. Pengirim email menggunakan kunci publik penerima untuk melakukan enkipsi kunci rahasia yang digunakan pada algoritma *cipher* simetri. Pada akhirnya kunci akan digunakan untuk melakukan enkripsi plainteks. Hampir semua knci publik pengguna PGP tersimpan pada *server* kunci PGP yang tersebar di seluruh dunia. Berikut ini adalah skema enkripsi dengan menggunakan PGP:




**Gambar 1 Skema Enkripsi PGP**

Penerima email yang terenkripsi tersebut menggunakan kunci sesi (*session key*) untuk melakukan dekripsi terhadap email tersebut. Kunci sesi ini terdapat pada email yang terenkripsi tersebut dan diperoleh dengan cara mendekripsinya dengan menggunakan kunci privat. Berikut ini adalah skema dekripsi pada PGP:




**Gambar 2 Skema Dekripsi PGP**

Strategi yang sama digunakan untuk mendeteksi apakah suatu pesan sudah mengalami perubahan atau belum dan juga untuk menentukan apakah pesan berasal dari pengirim yang sebenarnya. Pengirim menggunakan enkripsi PGP untuk memberikan tanda tangan digital (*digital signature*) pada pesan dengan algoritma RSA atau DSA. Untuk melakukannya, PGP melakukan komputasi nilai *hash* (*message digest*) dari plainteks untuk kemudian membubuhkan tanda tangan digital dari nilai *hash* tersebut dengan menggunakan kunci privat pengirim. Penerima pesan melakukan perhitungan nilai hash terhadap plainteks semula dan menggunakan kunci publik penerima untuk memberikan tanda tangan digital pada pesan tersebut. Jika tanda tangan hasil komputasi ini sesuai dengan tanda tangan yang terdapat pada pesan yang diterima maka pesan dapat diterima dengan tingkat

kepercayaan tinggi sebagai pesan yang asli tanpa adanya perubahan.

Baik ketika melakukan enkripsi pesan maupun pada saat melakukan verifikasi tanda tangan, sangatlah penting bahwa kunci publik yang digunakan untuk mengirim pesan pada seseorang juga merupakan "milik" penerima pesan. Dengan men-*download* kunci publik dari *server* kunci PGP, ada kemungkinan terjadinya penyadapan oleh pihak yang tidak bertanggung jawab. Akan tetapi, PGP telah menyediakan cara untuk mendistribusikan kunci publik dengan menggunakan sertifikat identitas yang dibangkitkan dengan algoritma kriptografi. Sejak versi pertama, PGP telah memberikan suatu skema pembuatan sertifikat secara internal yang disebut *web of trust*. Kunci publik yang diberikan dapat ditandatangani secara digital oleh pihak ketiga untuk menguji asosiasi antara seseorang dengan kunci tersebut. Skema ini juga memiliki beberapa tingkat kepercayaan untuk hasil pengujian tanda tangan.

*Web of trust* adalah suatu model kepercayaan yang kumulatif. Sebuah sertifikat dapat dipercaya secara langsung atau dipercaya dengan melalui perantara sertifikat lainnya. PGP menggunakan tanda tangan digital sebagai bentuk perkenalan. Ketika pengguna menandatangani kunci lain maka pengguna tersebut akan menjadi pengenal (*introducer*) kunci tersebut. Dengan cara demikian maka akan terbentuk suatu jaringan kepercayaan yang disebut *web of trust*.

Pada lingkungan PGP, setiap pengguna dapat berperan sebagai pihak yang berwenang terhadap serifikat. Setiap pengguna dapat melakukan validasi terhadap sertifikat kunci publik milik pengguna lain. Akan tetapi, suatu sertifikat hanya akan dianggap valid jika pengguna tersebut percaya kepada pengguna lain yang berperan sebagai pengenal (*introducer*).

Pada spesifikasi OpenPGP, tanda tangan yang terpercaya dapat digunakan untuk mendukung pembuatan sertifikat otoritas (*certificate authorities*). Tanda tangan yang terpercaya ini menandakan bahwa suatu kunci memang merupakan milik pemiliknya dan pemilik kunci tersebut dipercaya juga untuk menandatangani kunci lain yang memiliki tingkat kepercayaan satu tingkat di bawahnya. Tanda tangan tingkat 0 dapat dibandingkan tanda tangan *web of trust* karena hanya validitas kunci yang disertifkasi. Tanda tangan tingkat 1 sama dengan kepercayaan yang diberikan seseorang pada sertifikat otoritas (*certificate authorities*) karena kunci yang dengan tanda tangan tingkat 1 dapat membuka tanda tangan tingkat 0 tanpa adanya batasan jumlah. Tanda tangan tingkat 2 dapat disamakan dengan asumsi kepercayaan pengguna ketika menggunakan sertifikat otoritas (*certificate authorities*) pada Internet Explorer dan memungkinkan pemilik kunci untuk membuat sertifikat otoritas kunci lainya.

PGP juga memiliki fitur untuk membatalkan (*revoke*) sertifikat identitas yang sudah tidak valid. Hal ini kurang lebih sama dengan *certificate revocation list* pada skema *Public Key Infrastructure*. PGP versi terakhir juga mendukung fitur untuk memeriksa sertifikat yang sudah tidak berlaku lagi.

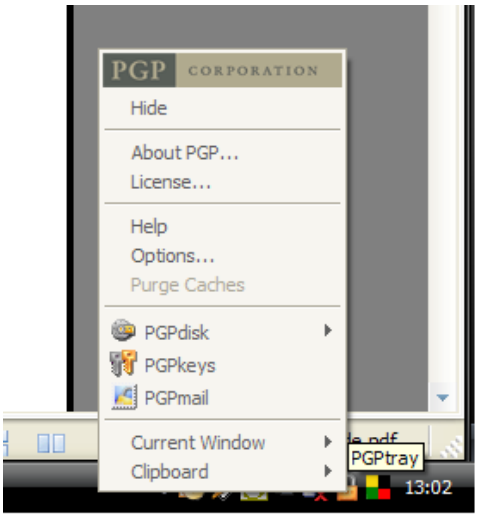
## 4. Eksperimen dengan PGP

Pada bagian ini akan dijelaskan mengenai cara penggunaan dan hasil eksperimen menggunakan PGP. Beberapa hal yang akan dibahas adalah cara membuat pasangan kunci (*keypair*), meyimpan kunci ke server, mendapatkan kunci publik milik orang lain, dan mengenkripsi dan mendekripsi pesan. Sebelum membahas mengenai beberapa hal tersebut, akan dijelaskan terlebih dahulu mengenai antarmuka PGP dan cara menggunakannya.

### 4.1 Antarmuka PGP

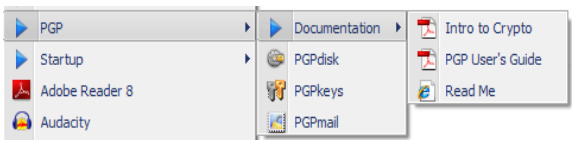
Terdapat beberapa cara untuk mengakses PGP, yaitu melalui PGPtray icon, "Start" menu, Windows Explorer dan Microsoft Outlook.

1. PGPtray icon



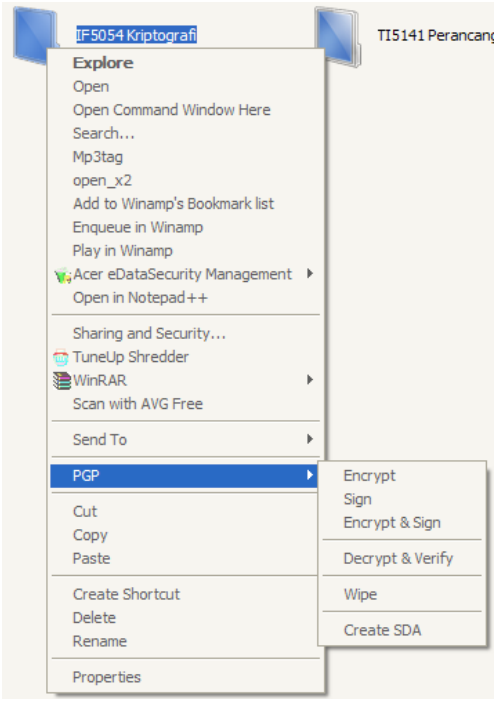

**Gambar 3 PGP-PGPtray icon**

2. "Start" menu



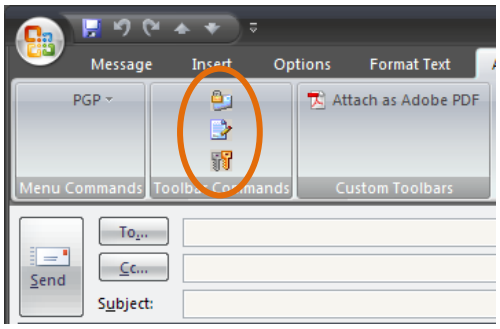

**Gambar 4 PGP-"Start" menu**

3. Windows Explorer




**Gambar 5 PGP-Windows Explorer**

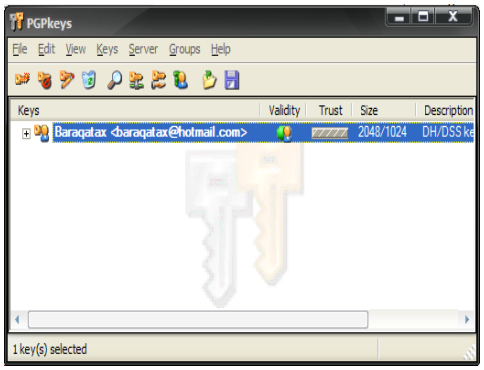
4. Microsoft Outlook




**Gambar 6 PGP-Microsoft Outlook**

Pada PGP terdapat tiga antarmuka utama, yaitu:

1. PGPkeys

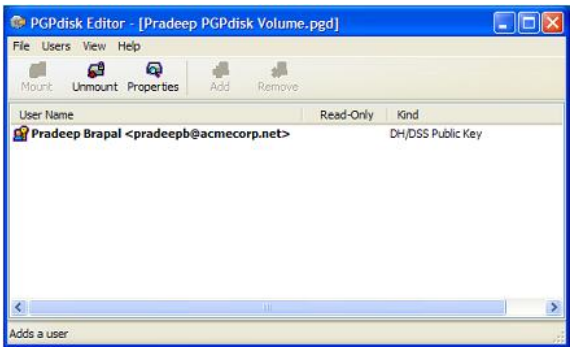



**Gambar 7 Antarmuka PGPkeys**

2. PGPmail


**Gambar 8 Antarmuka PGPmail**

3. PGPdisk
   Fitur ini memerlukan License Number PGP Personal untuk aktivasinya.

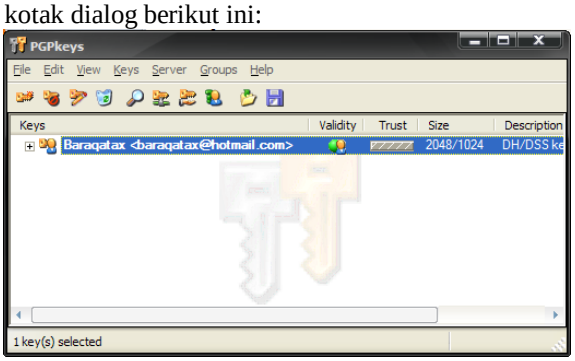



**Gambar 9 Antarmuka PGPdisk**

## 4.1 Pembuatan Pasangan Kunci (Keypair)

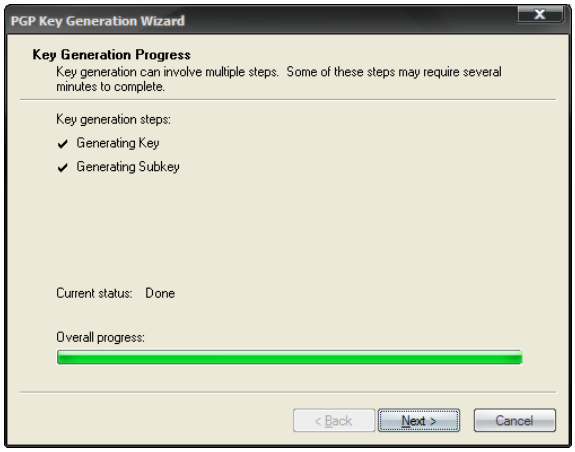
Berikut ini adalah langkah-langkah yang diperlukan untuk membuat pasangan kunci (*keypair*):

1. Pilih menu **Start → All Programs → PGP → PGPkeys** atau klik PGPtray icon kemudian pilih PGPkeys. Akan muncul kotak dialog berikut ini:




**Gambar 10 Antarmuka PGPkeys**

2. Klik icon '**Generate new keypair**' atau pilih menu **Keys → New Key**.
3. Kemudian akan muncul kotak dialog '**PGP Key Generation Wizard'**, klik **Next**.
4. Masukkan **Full Name** dan **Email Address** pada field yang tersedia, klik **Next**.
5. Masukkan **Passphrase** (minimal 8 karakter), kemudian lakukan konfirmasi **Passphrase**, klik **Next.** Pada kotak dialog ini terdapat petunjuk mengenai kualitas **Passphrase** yang dibuat.
6. Setelah itu, PGP akan melakukan pembuatan kunci dan juga subkunci, sperti terlihat pada gambar berikut ini:



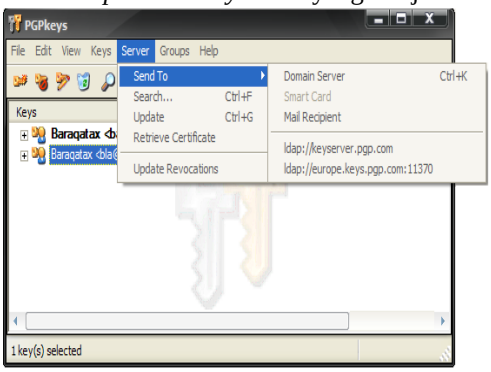

**Gambar 11 Pembuatan Kunci dan Subkunci**

7. Setelah pembuatan kunci selesai, klik **Next**.
8. Klik **Finish**.

## 4.2 Menyimpan Kunci ke Server

Berikut ini adalah langkah-langkah yang diperlukan untuk menyimpan kunci ke server:

1. Pilih menu **Start → All Programs → PGP → PGPkeys** atau klik PGPtray icon kemudian pilih PGPkeys.
2. Kemudian pilih menu **Server → Send To**, pilih *keyserver* yang menjadi tujuan. Kunci akan di-*upload* ke *keyserver* yang dituju.




**Gambar 12 Menyimpan Kunci ke Server**

## 4.3 Mendapatkan Kunci Publik

Berikut ini adalah langkah-langkah yang diperlukan untuk mendapatkan kunci publik milik orang lain:

1. Pilih menu **Start → All Programs → PGP → PGPkeys** atau klik PGPtray icon kemudian pilih PGPkeys.
2. Pada kotak dialog PGPkeys, pilih menu Server → Search. Kemudian akan muncul kotak dialog **PGPkeys Search Window**, seperti terlihat pada gambar berikut:


**Gambar 13 PGPkeys Search Window**

3. Pilih server pada *field* **Search for keys on**.
4. Tentukan kriteria pencarian, apakah berdasarkan *user ID*, *key ID*, *key type*, *creation date*, *expiration date*, *key status*, atau *key size*.
5. Kemudian klik **Search**.

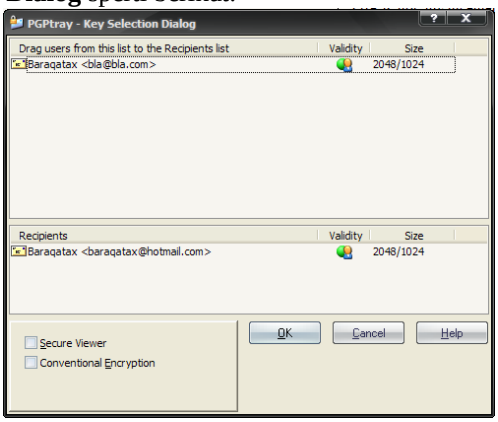
## 4.4 Enkripsi dan Dekripsi Pesan

Berikut ini adalah langkah-langkah yang diperlukan untuk melakukan enkripsi dan dekripsi pesan:

1. Siapkan pesan yang akan dienkripsi, dapat dilakukan dengan menggunakan program notepad. Contoh pesan:

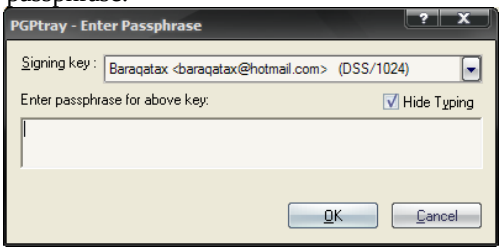
> "Life is not advancement. It is growth. It does not move upward, but expands outward, in all directions."
> ==Russell G. Alexander==

2. Kemudian klik **PGPtray icon**, pilih **Current Window → Encrypt**. Setelah itu, akan muncul kotak dialog **Key Selection Dialog** sperti berikut:




**Gambar 14 Key Selection Dialog**

3. Setelah itu pilih user yang akan menjadi *recipient* (penerima), klik **OK**.
4. Akan muncul kotak dialog **PGPtray-Enter Passphrase** berikut untuk memasukkan passphrase:




**Gambar 15 PGPtray-Enter Passphrase**

5. Hasil dari enkripsi pesan tersebut adalah sebagai berikut:

> -----BEGIN PGP MESSAGE-----
> Version: PGP 8.0 - not licensed for commercial use: www.pgp.com
>
> qANQR1DBwU4DgrMrtiWA8x4QCADB3i2w
> kO32xuto5zSg5aVMrxfcQP5YXu0ypAZT

```
DJEiOJGSrxWUH9LPB1PL57oBAaGrNqigolY
tI8GR36VVBcDoHGgsL6WwDvb0Uup8
bONWBmMun9sz513BWbwJ48js98PKprJolcz
UNonLIwNSwL4MOpU+hReKhLaacKqa
Rz9k1c3JwVfVN5EYYaqpMjoWXgFOQJiBuZ
zXN2/0qJmR2BmOr8bQUok6rnpDlZS2
AWquF/Bqp9FDDPXDaaN2Qsi8pgyplXA3iaFi
PNQMZELIFa6IIgTjSMQJaHJMwPki
qkm7iR+M2C8iEGzI3o+tgbxZoSosoiRiOhBS+
4F8TlANfB0ICACATFqr16wk1bHp
uTO8FlwxNBLN/2JRavg2RpPlxjdbHK89cZN7
/dY60ZZ/aTFfj0bLkP5OLVP2RFY+
fgzQNFkQXkhBPy+GRJ17nM+ISiVK2hOnda
VVCGuF0T37ZyfdhoWoOtwGWsHvdUB7
CyOxrICinx6I/xGXDBoga7zl8bBj4TcOGd2bef
kEignE/LgfR2GIHdnx+0fazSN0
RvbWLql/egKkgh3ZicJlxkRxAqFFYUsVb5Jbu
oiXnW3uMdU4B2Bpm6FSIyof3NJy
q4KXamxZDvfBUb8kER8cUACedG2PjpQePn
BNusvToHayHpC/HKMqz8049w2wzK4d
ZjAufsH6yY7RbpezYwFx0pQhv9HCJRHRyT0
CXEKpKmpQyiUA96iD6gg9gxy8myq0
Wy0CNJp3rvwjJwVCvFjBm8WNROePLmYGi
+ENqTW6R2JP4yHkNExTqu+VxQcJ+Cys
iVPu2OHAASFlaXdL2Oz340g7nql6w/jksZrW
XB/ldFrKFkWB1w41lh7iLWvP2IHi
sGCFiJ9a
=6iC0
-----END PGP MESSAGE-----
```

6. Untuk melakukan dekripsi pesan, klik PGPtray icon, lalu pilih **Current Window → Decrypt**. Setelah itu, masukkan *passphrase* pada kotak dialog.
7. Pesan akan terdekripsi menjadi pesan semula.

## 5. Algoritma pada PGP

Setiap algoritma kriptografi dapat memecahkan suatu permasalahan tertentu. Versi terbaru dari PGP mendukung berbagai jenis algoritma yang dapat dipilih untuk digunakan. Hal ini sesuai dengan prinsip disribusi PGP, yaitu setiap pengguna dapat menggunakan algoritma kripografi tertentu yang dianggap aman. Berikut ini adalah penjelasan mengenai tipe dan jenis algoritma kripografi tersebut:

1. Cipher Blok
   Algoritma enkripsi simetri melakukan enkripsi secara normal di mana kunci yang sama digunakan untuk melakukan enkripsi dan dekripsi. Cipher blok adalah cipher simetri yang beroperasi pada blok berukuran 8 atau 16 byte. Kunci cipher blok juga memiliki ukuran tertentu, umumnya 16 atau 128 byte. Cipher blok banyak digunakan karena algoritma ini cepat, mudah untuk diprogram, efisien, dan dapat menangani data dalam jumlah besar.
2. Algoritma Kunci Publik
   Algoritma kunci publik memiliki dua jenis kunci, yaitu kunci publik dan kunci privat. Algoritma ini biasanya lambat dan juga rentan terhadap beberapa serangan kriptanalis tertentu. Oleh karena itu, algoritma ini hanya akan digunakan untuk melakukan enkripsi pada bagian tertentu pada data, seperti kunci sesi (*session key*) dari cipher blok atau untuk mengenkripsi nilai hash tertentu. Algoritma kunci publik umumnya digunakan untuk melakukan enkripsi dan tanda tangan digital.
3. Fungsi *Hash*
   Fungsi *hash* menerima input berupa panjang seluruh dokumen atau pesan untuk kemudian menghasilkan nilai *hash* atau disebut juga *fingerprint* dari dokumen dengan ukuran tertentu, biasanya 16 atau 20 byte. *Fingerprint* memiliki dua atribut penting, yaitu seorang tidak akan dapat menemukan dokumen hanya dengan melihat *fingerprint* dokumen tersebut dan pada kenyataannya tidak ada dua dokumen yang memiliki *fingerprint* yang sama. Secara teori mungkin saja terdapat *fingerprint* yang sama untuk dua jenis dokumen, akan tetapi karena nilai hash yang sangat banyak maka akan dibutuhkan waktu yang sangat lama untuk membangkitkan *fingerprint* yang sama tersebut. Oleh karena itu, enkripsi cukup dilakukan terhadap nilai *hash* suatu dokumen bukan terhadap keseluruhan dokumen.
4. Algoritma *Secret Sharing*
   Algoritma ini tidak dibutuhkan pada fitur dasar dari PGP tetpai dapat digunakan pada versi yang baru. Algoritma ini dapat membagi kunci privat menjadi sejumlah *m* blok sehingga jika beberapa orang menggabungkan sejumlah *n* blok secara bersamaan menjadi *m* blok maka akan didapatkan kembali kunci privat tersebut. Algoritma ini biasanya digunakan pada perusahaan yang memiliki tiga orang direktur, di mana dibutuhkan dua dari tiga orang untuk menandatangani suatu kontrak.

Pada bagian ini akan dibahas mengenai beberapa algoritma yang digunakan pada PGP, antara lain algoritma Cipher Feedback (CFB), RSA, dan SHA-1. Ketiga algoritma tersebut merupakan algoritma yang biasa digunakan pada enkripsi dengan menggunakan PGP.

### 5.1 Cipher Feedback (CFB)

Enkripsi yang dilakukan dengan algoritma Cipher Feedback (CFB) tidak dapat diterapkan jika blok plainteks yang diterima belum lengkap. Pada mode CFB ini, data dienkripsikan menjadi unit yang lebih kecil daripada ukuran blok. Unit yang dienkripsikan

dapat berupa bit per bit seperti cipher aliran atau beberapa bit. Jika unit yang dienkripsikan satu karakter dalam setiap kali operasinya maka mode CFB disebut CFB 8-bit.

Secara umum, CFB n-bit mengenkripsi plainteks sebanyak n bit setiap kali operasinya, di mana n $\leq$ m (m=ukuran blok). Dengan kata lain, CFB mengenkripsikan cipher blok seperti pada cipher aliran. Mode CFB membutuhkan sebuah antrian (*queue*) yang berukuran sama dengan blok masukan.

Berikut ini adalah langkah detil algoritma CFB 8-bit:

1. Antrian diisi dengan *IV* (*initialization vector*).
2. Enkripsikan antrian dengan kunci K. Delapan bit paling kiri dari hasil enkripsi berlaku sebagai keystream yang kemudian di-XOR-kan dengan karakter 8-bit dari plainteks menjadi karakter 8-bit pertama dari cipherteks. Karakter cipherteks ini dikirim pada komunikasi data atau disimpan pada aplikasi penyimpanan data. Salinan dari cipherteks ini juga dimasukkan ke dalam antrian, menempati 8 posisi bit paling kanan, dan semua byte yang lainnya di dalam antrian digeser ke kiri menggantikan 8-bit pertama yang sudah digunakan.
3. Karakter plainteks berikutnya dienkripsikan dengan cara yang sama seperti pada langkah 2.
4. Dekripsi dilakukan dengan sebagai kebalikan dari proses enkripsi. Baik enkripsi maupun dekripsi, algoritma E dan D yang digunakan adalah sama.

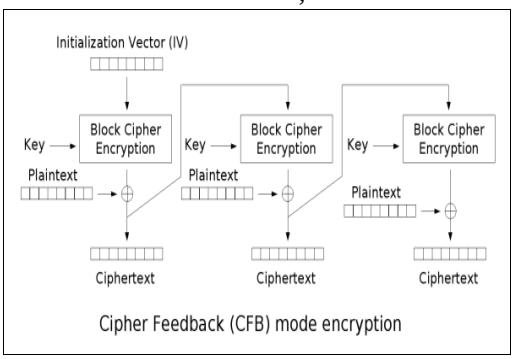
*IV* (*initialization vector*) pada CFB tidak perlu dirahasiakan tetapi *IV* harus unik untuk setiap pesan karena *IV* yang sama untuk setiap pesan yang berbeda akan menghasilkan *keystream* yang sama.

Selain itu, hal lain yang perlu diperhatikan pada CFB adalah mengenai perambatan kesalahan. Kesalahan 1-bit pada blok plainteks akan merambat pada blok-blok cipherteks yang berkoresponden dan blok-blok cipherteks selanjutnya pada proses enkripsi. Hal ini juga berlaku pada proses dekripsi. Perubahan ini seringkali dideteksi oleh PGP jika data terenkripsi telah ditandatangani.

Serangan yang dilakukan pada blok plainteks yang dipilih, seseorang dapat melakukan dekripsi pada setiap blok, kecuali pada blok plainteks tersebut. Serangan ini bukan merupakan ancaman yang serius pada PGP tetapi dapat menjadi menarik jika dilakukan pada penggunaan CFB yang lain.

Enkripsi pada mode CFB:

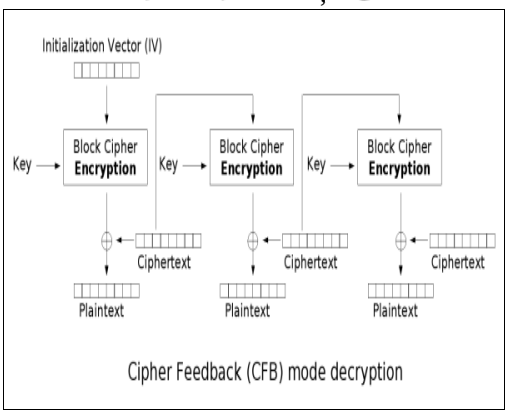
$$C_i = E_K(C_{i-1}) \oplus P_i, C_0 = \text{IV}$$




**Gambar 16 Skema Enkripsi CFB**

Dekripsi pada mode CFB:

$$P_i = E_K(C_{i-1}) \oplus C_i, C_0 = \text{IV}$$




**Gambar 17 Skema Dekripsi CFB**

## 5.2 RSA

Algoritma RSA dibuat oleh 3 orang peneliti dari MIT (*Massachussets Institute of Technology*) pada tahun 1976, yaitu Ron Rivest, Adi Shamir, dan Leonard Adleman. Keamanan algoritma RSA terletak pada sulitnya memaktorkan bilangan yang besar menjadi faktor-faktor prima. Pemfaktoran dilakukan untuk memperoleh kunci privat. Selama pemfaktoran bilangan besar menjadi faktor-faktor prima belum ditemukan algoritma yang mangkus, maka selama itu pula keamanan RSA tetap terjamin.

Algoritma RSA memiliki besaran-besaran sebagai berikut:

1. p dan q bilangan prima (rahasia)
2. n=p.q (tidak rahasia)

3. $\Phi(n)=(p-1)(q-1)$ (rahasia)
4. e (kunci enkripsi) (tidak rahasia)
5. d (kunci dekripsi) (rahasia)
6. m (plainteks) (rahasia)
7. c (cipherteks) (tidak rahasia)

Algoritma untuk membangkitkan pasangan kunci:

1. Pilih dua bilangan prima sembarang, p dan q.
2. Hitung n=p.q (sebaiknya $p \neq q$, sebab jika p=q maka $n=p^2$ sehingga p dapat diperoleh dengan menarik akar pangkat dua dari n).
3. Hitung $\Phi(n)=(p-1)(q-1)$.
4. Pilih kunci publik e, yang relatif prima terhadap $\Phi(n)$.
5. Bangkitkan kunci privat dengan menggunakan persamaan $e.d \equiv 1 (\text{mod} \Phi(n))$. Perhatikan bahwa $e.d \equiv 1 (\text{mod} \Phi(n))$ ekivalen dengan $e.d=1+k\ \Phi(n)$, sehingga secara sederhana d dapat dihitung dengan rumus $d = \frac{1 + k\Phi(n)}{e}$.

Hasil dari algoritma di atas adalah:

- Kunci publik adalah pasangan (e,n).
- Kunci privat adalah pasangan (d,n).
  Catatan: n tidak bersifat rahasia karena diperlukan pada saat melakukan enkripsi dan dekripsi.

Algoritma enkripsi RSA:

1. Ambil kunci publik penerima pesan, e, dan modulus n.
2. Nyatakan plainteks m menjadi blok-blok m1, m2, ..., sedemikian sehingga setiap blok merepresentasikan nilai dalam selang [0, n-1].
3. Setiap blok m, dienkripsi menjadi blok $c_i$ dengan rumus $c_i=m_i^e \bmod n$.

Algoritma dekripsi RSA:

1. Setiap blok cipherteks $c_i$ didekripsi kembali menjadi $m_i$ dengan rumus $m_i=c_i^d \bmod n$.
2. Kemudian setiap blok mi digabungkan kembali untuk mendapatkan plainteks m semula.

## 5.3 SHA-1

SHA-1 menerima masukan berupa pesan dengan ukuran maksimum $2^{64}$ bit dan menghasilkan *message digest* yang panjangnya 160 bit, lebih panjang dari *message digest* yang dihasilkan oleh MD5 yang hanya 128 bit.

Langkah-langkah pembuatan *message digest* dengan SHA-1 secara garis besar adalah sebagai berikut:

1. Penambahan bit-bit pengganjal (padding bits).
2. Penambahan nilai panjang pesan semula.
3. Inisialisasi penyangga (*buffer*) *message digest.*
4. Pengolahan pesan dalam blok berukuran 512 bit.

**Penambahan Bit-Bit Pengganjal**

Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Ini berarti panjang pesan setelah ditambahi bit-bit pengganjal adalah 64 bit kurang dari kelipatan 512. Angka 512 ini muncul karena SHA-1 memproses pesan dalam blok-blok yang berukuran 512. Pesan dengan panjang 448 bit pun akan tetap ditambah dengan bit-bit pengganjal. Jika panjang pesan 448 bit maka pesan tersebut ditambah dengan 512 bit menjadi 960 bit. Jadi, panjang bit-bit pengganjal adalah antara 1 sampai 512. Bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

**Penambahan Panjang Pesan Semula**

Pesan yang telah diberi bit-bit pengganjal selanjutnya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Setelah ditambah dengan 64 bit, panjang pesan menjadi kelipatan 512 bit.

**Inisialisasi Penyangga *Message Digest***

SHA mempunyai 5 buah penyangga (*buffer*) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 5x32=160 bit. Kelima penyangga ini menampung hasil antara dan hasilakhir. Kelima penyangga itu diberi nama A, B, C, D, E. Setiap penyangga diinisialisasi dengan nilai heksadesimal sebagai berikut:

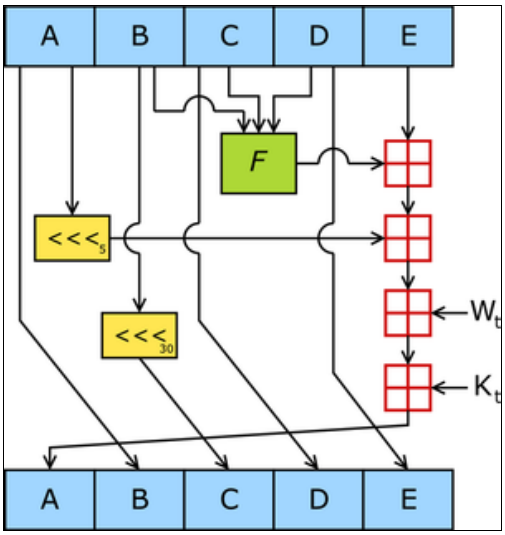
A=0x67452301
B=0xEFCDAB89
C=0x98BADCFE
D=0x10325476
E=0xC3D2E1F0

**Pengolahan Pesan dalam Blok Berukuran 512 bit**

Pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit ($Y_0$ sampai $Y_{L-1}$). Setiap blok 512 bit diproses bersama dengan penyangga *message digest* menjadi keluaran 160 bit, proses ini disebut $H_{SHA}$. Proses $H_{SHA}$ terdiri dari 80 putaran dan masing-masing putaran menggunakan bilangan penambah $K_t$, yaitu:

Putaran $0 \le t \le 19$ $K_t$=5A827999
Putaran $20 \le t \le 39$ $K_t$=6ED9EBA1
Putaran $40 \le t \le 59$ $K_t$=8F1BBCDC
Putaran $60 \le t \le 79$ $K_t$=CA62C1D6

Setiap putaran menggunakan operasi dasar yang sama, dapat dilihat pada gambar berikut ini:




**Gambar 18 Skema Operasi RSA**

Keterangan gambar:

- A, B, C, D, E: lima buah variabel penyangga 32 bit, masing-masing memiliki 80 putaran
- F: fungsi logika
- <<<: *circular left shift* (CLS)
- $W_t$: *word* 32 bit yang diturunkan dari blok 512 bit yang sedang diproses
- Kt: konstanta penambah
- ⊞: operasi penjumlahan dalam modulo 232

## 6. Analisis Keamanan pada PGP

Setiap aplikasi atau program komputer dapat memiliki sejumlah *bug* (kesalahan) pada kodenya, tetapi tidak semua *bug* tersebut berbahaya dan ada pula yang tidak terdeteksi hinga akhirnya digantikan oleh kode lain dengan *bug* yang berbeda pula. Beberapa *bug* ditemukan pada saat pengujian oleh pembuat program. Oleh karena itu, suatu program dapat memiliki versi terbaru dengan perbaikan pada kesalahan yang ditemukan tersebut. Umumnya pembuat aplikasi akan memberikan *update* versi terbaru itu melalui internet.

Terkadang terdapat pula *bug* yang ditemukan oleh orang lain yang bukan pembuat program tersebut. Jika suatu *bug* mempengaruhi sistem keamanan komputer, prosedur berikut ini akan dilakukan, yaitu menghubungi perusahaan atau individu yang membuat program tersebut untuk memperbaiki program sehingga tidak lagi mengandung *bug*. Kemudian perusahaan atau individu tesebut akan mempublikasikan *bug* melalui internet dengan tujuan *bug* tersebut dapat diperbaiki bersama-sama oleh komunitas pengguna.

Seringkali perusahaan tidak bereaksi dengan cepat dan tepat dalam menangani masalah seperti ini sehingga dapat menimbulkan celah keamanan pada aplikasi tersebut. Celah keamanan ini dapat dimanfaatkan oleh pihak yang tidak bertanggung jawab untuk melakukan eksploitasi terhadap keamanan suatu sistem komputer.

Pada bagian ini akan dibahas mengenai beberapa celah dan analisis keamanan pada PGP, yaitu Linux *random bug*, jadwal kunci PGPdisk, ADK *bug*, serangan kunci privat Czech, ASCII armor *bug*, dan serangan *multiple user ID*.

### 6.1 Linux *Random Bug*

Pada PGP untuk Linux terdapat beberapa *bug* yang dapt membangkitkan kunci yang mudah ditebak. Versi PGP tersebut adalah PGP 5.01 untuk Linux. Kesalahan terjadi jika pengguna membangkitkan sebuah kunci dengan perintah pgpk dari konsol, kemudian PGP mencoba untuk membaca dari *random device* dengan kode berikut ini:

```
RandBuf = read(fd, &RandBuf, count);
```

Masalah yang timbul adalah perintah *read* tidak mengembalikan nilai *byte* yang *random*, tetapi mengembalikan jumlah *byte* yang dibaca, dalam hal ini adalah 1 karena count=1. Seharusnya kode tersebut adalah sebagai beikut:

```
read(fd, &RandBuf, count);
```

Kesalahan ini ditemukan oleh Germano Caronni pada tanggal 23 Mei 2000 dan telah diselesaikan tanpa perlu menunggu versi baru PGP.

Kesalahan ini cukup fatal karena dapat membuat fitur pembangkitan kunci secara non-interaktif menjadi tidak berguna. Sebagian besar fungsi pada PGP adalah untuk membangkitkan bilangan *random* secara aman. PGP memiliki administrasi pembangkitan bilangan *random* yang sangat canggih dan melakukan fungsi *hash* pada data *random* sebanyak dua kali. Akan tetapi pada saat itu, kesalahan atau *bug* ini bukan merupakan celah yang menarik jika dilihat dari sudut pandang kriptografi.

### 6.2 Jadwal Kunci PGPdisk

Pada beberapa versi PGPdisk terdapat *bug* atau kesalahan yang muncul pada jadwal kunci cipher simetri.Versi yang memiliki *bug* tersebut adalah PGP disk versi 1.0 untuk Windows dan PGPdisk yang terdapat pada paket PGP versi

6.0.1 untuk Windows. Namun, pada PGP versi 6.0.2 kesalahan ini telah diperbaiki.

PGPdisk adalah program yang memungkinkan pengguna untuk melakukan enkripsi pada *hard disk* sehingga orang lain tidak dapat menggunakan *hard disk* tanpa mempunyai *passphrase*. PGPdisk merupakan program yang terpisah dari PGP. *Bug* yang terdapat pada PGPdisk terletak pada fungsi jadwal kunci pada enkripsi CAST. Seperti halnya pada IDEA, CAST dapat membangkitkan himpunan subkunci dari kunci dengan ukuran 1024 bit. Kunci tersebut hanya disalin ke *buffer* sehingga 128 bit pertama dari subkunci yang akan diinisialisasi, sedangkan yang bit-bit lainnya dibiarkan nol. Hal ini berarti enkripsi hanya baik untuk dua putaran CAST sehingga kunci relatif mudah ditebak jika seseorang mengetahui sebagian dari plainteks.

*Bug* ini ditemukan oleh NAI, perusahaan perangkat lunak yang memproduksi PGP. Kesalahan yang ditimbulkan tidak akan berpengaruh pada PGP karena terletak pada cipher yang menyebabkan pesan tidak dapat terbaca.

## 6.3 ADK *Bug*

Pada bulan Agustus 2000, Ralph Senderek, seorang peneliti dari Jerman menemukan bug yang berhubungan dengan *Additional Decryption Keys* (ADK). Senderek membuktikan *bug* ini melalui eksperimen yang dilakukannya. Eksperimen dilakukan dengan menambahkan subpaket ADK pada kunci publik di luar daerah yang ditandatangani. Hasilnya adalah sebagian subpaket harus ditandatangani oleh pemilik dan sebagian lagi tidak. Pada spesifikasi OpenPGP, kunci ADK harus ditempatkan pada daerah yang telah ditandatangani, tetapi eksperimen ini membuktikan bahwa PGP dapat menerima kunci ADK yang diletakkan di luar daerah tersebut.

Penemuan ini merupakan *bug* yang terdapat pada implementasi PGP bukan pada spesifikasi OpenPGP. Dengan *bug* ini, seseorang akan mendapatkan kunci publik orang lain dan menambahkan kunci ADK miliknya kemudian mendistribusikan kunci sebanyak yang diinginkannya. *Bug* ini terdapat pada PGP versi 5.5.x sampai 6.5.3 dan telah diperbaiki pada versi 6.5.8. Kesalahan ini didefinisikan dalam dua cara, yaitu sebagai kesalahan implementasi pada perancangan spesifikasi yang dapat ditemukan pada fungsi `ringKeyAdditionalRecipientRequestKey` pada *file* pgpRngPub.c. Pada *file* tersebut terdapat kode sebagai berikut:

```
krpdata = ringKeyFindSubpacket (obj,
set,SIGSUB_KEY_ADDITIONAL_RECIPIENT_
REQUEST, nth, &krdatalen,
&critical, &hashed, NULL, &matches,
error);

if (!krpdata || !hashed) {
if (IsntPGPError(*error))
*error = kPGPError_ItemNotFound;
return NULL;
}
```

Pada kode tersebut terdapat variabel `hashed` yang seharusnya diabaikan. Pada PGP versi 6.5.1 kode tersebut telah diperbaiki menjadi:

```
krpdata = ringKeyFindSubpacket (obj,
set,SIGSUB_KEY_ADDITIONAL_RECIPIENT_
REQUEST, nth, &krdatalen,
&critical, NULL, NULL, &matches,
error);

if (!krpdata) {
if (IsntPGPError(*error))
*error = kPGPError_ItemNotFound;
return NULL;
}
```

Ralph Senderek menyatakan bahwa PGP versi 2.6.x merupakan versi terbaik dari seluruh versi PGP karena versi yang baru terlalu kompleks untuk dinyatakan aman. Dia juga menyatakan bahwa tidak mungkin untuk melakukan analisis kode sumber (*source code*) untuk program yang berukuran sangat besar tersebut.

## 6.4 Serangan Kunci Privat Czech

Pada tanggal 20 Maret 2001, Vlastimil klima dan Tomas Rosa, dua orang ilmuwan dari Ceko, mempublikasikan jenis serangan baru pada OpenPGP dan PGP. Mereka melakukan eksploitasi pada kelemahan OpenPGP untuk fitur DSA. Tujuan dari serangan ini adalah untuk mendapatkan kunci privat milik orang lain. Kunci ini terdapat pada *secret keyring* dan didapatkan dengan cara memasuki komputer seseorang. PGP menyimpan kunci privat ini dalam bentuk yang terenkripsi sehingga dibutuhkan *passphrase* untuk mendekripsi kunci tersebut.

Kedua ilmuwan tersebut mengetahui fakta bahwa kesalahan yang dibuat dalam mengkalkulasi tanda tangan dapat membocorkan informasi rahasia pada kunci. Mereka memperlihatkan kesalahan tersebut dengan melakukan modifikasi pada *file* kunci

rahasia. Skenario serangan tersebut adalah sebagai berikut:

1. Memasuki kantor seseorang, menjalankan komputer, menyalin *private keyring* ke *floppy disk*, dan melakukan perubahan pada kunci privat.
2. Menunggu hingga seseorang kembali dan membuat tanda tangan dengan kunci privat miliknya. Pada kondisi normal, pembuatan dan publikasi tanda tangan merupakan sesuatu yang aman sehingga orang tersebut tidak akan curiga pada saat melakukannya.
3. Melakukan komputasi kunci rahasia dengan menggunakan tanda tangan. Hal ini membutuhkan sedikit pengetahuan matematika tergantung pada format kunci, apakah format RSA atau Diffie-Hellman.
4. Memasuki kembali kantor tersebut kemudian mengembalikan *private keyring* asli. Langkah ini perlu dilakukan agar korban tidak mengetahui bahwa kunci privat miliknya telah diganti.

Contoh serangan pertama dilakukan pada kunci privat algoritma RSA, yang memiliki variabel dengan struktur sebagai berikut:

- data publik: n dan e
- data rahasia: d, p, q, u, dan *checksum*

Semua bilangan bulat besar tersebut (n, e, d, p, q, u) disimpan dengan format sebagai berikut, panjang sebagai bilangan 16-bit dan *content byte* untuk menyimpan data pada orde pertama *most significant byte* (MSB). Pada PGP versi 3, hanya *content byte* yang dienkripsi sedangkan panjangnya tidak. Pada PGP versi 4, seluruh bagian data rahasia dienkripsi menjadi satu partisi (*chunk*) yang besar. Dua *byte checksum* terdiri dari jumlah seluruh *byte* data rahasia modulo 65536. Dalam hal ini, penyerang berusaha mengubah nilai u. Untuk melakukan serangan pada PGP versi 3, penyerang akan mengubah nilai panjang u menjadi 1 dan juga mengubah nilai *checksum*.

Ketika korban mencoba untuk melakukan kalkulasi pada kunci maka ia akan mengkonstruksi hasil dari operasi RSA sebagai berikut:

$$S` = s1+p*u`*(s2-s1) \bmod n = S+k*p$$

Dengan nilai k tertentu yang bukan nol sehingga perbedaan antara S dengan S` merupakan kelipatan dari p. Penyerang tidak mengetahui S, akan tetapi jika ia mengetahui dokumen yang sudah ditandatangani maka nilai $S^e$ akan dapat diketahui melalui perhitungan berikut:

$$A = S`^e - S^e = S^e + p*[\text{bilangan lain}] - S^e$$

A merupakan kelipatan p sehingga penyerang dapat melakukan kalkulasi p=ged(A,N) dan memfaktorkan kunci milik korban.
Serangan yang dilakukan pada versi 4 sedikit lebih sulit karena panjang dari bilangan bulat besar yang dienkripsi. Hal yang dilakukan oleh penyerang adalah dengan mengubah satu bit terakhir pada u dan mengubah posisi bit yang sama pada *checksum*. Perubahan satu bit pada data terenkripsi ini akan mengakibatkan perubahan satu bit pula pada data yang didekripsi karena penggunaan mode CFB pada PGP. *Checksum* akan melakukan pemeriksaan apakah byte u dan *checksum* keduanya turun atau naik. Serangan ini memiliki lima puluh persen kemungkinan berhasil.

Serangan pada RSA ini tidak dapat dilakukan pada PGP versi 7 karena versi ini melakukan pemeriksaan pada semua kunci publik yang dibacanya.

## 6.5 ASCII Armor *Bug*

*Bug* ini ditemukan pada bulan April 2001 oleh Chris Anley. Dia menyatakan bahwa membuka file ASCII sebagai kunci publik atau hanya mengambil tanda tangan saja dapat menyebabkan terciptanya *arbitrary file* pada komputer. Pada sistem operasi Windows, hal ini dapat meneyebabkan eksekusi dari kode secara otomatis pada mesin target.

Permasalahannya adalah parser ASCII pada PGP dapat membuat *temporary file* dan tidak melakukan penghapusan *file* tersebut jika terjadi sesuatu pada input. Hal ini memungkinkan untuk membuat suatu *file* yang menyebabkan parser akan menuliskan *file* apapun ke *hard disk*. *File* ini akan tetap ada pada sistem dan tidak terhapus.

Chris Anley melakukan percobaan dengan menggunakan *file* DLL. DLL adalah *Dynamic Link Library* yang mengandung subrutin yang dapat dipanggil oleh suatu program lain. Setiap program pada Windows dapat terdiri dari file .exe dan juga DLL. Jika *file* DLL yang dibuat tersebut memiliki nama yang sama dengan nama DLL yang terdapat pada program lain maka DLL tersebut akan dieksekusi setiap kali pengguna menjalankan program yang berhubungan dengan DLL tersebut. Versi PGP yang rentan terhadap serangan ini adalah versi 5.0 sampai 7.0.4. Sebagai tindak lanjut terhadap masalah ini, NAI memperbaiki *bug* ini dengan memberi peringatan jika terdapat file yang tertinggal pada sistem.

Contoh *file* yang dibuat Anley bukan merupakan *file* ASCII yang terlindungi. *File* ini diberi nama *notes.doc.sig* yang mengandung hanya satu literal paket, yaitu:

```
af 30 c4 70 67 70 73 63 2e 64 6c 6c 00
20 61 72 62 69 74 72 61 72 79 20 66 69
6c 65 6e
61 6d 65 00 a0 ee 01 3d de f1 05 4c a0
16 92 fa de cb 69 cf 8a 85 3f 84 0b 62
c9 c5 ed
0d 16 35 d7 e2 21 b3 bd 52 a7 dc 56 89
36 d0 d1 4f 87 39 c4 2c 0d 2f 2e 2e 2f
2e 2e 2f
2e 2e 2f 2e 2e 2f 2e 2e 2f 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20 20 20 20 20 20 20 20 20 20 20
20 20 20
4d 5a 90 00 03 ...
```

Pada baris pertama, *af*, atau *10 1011 11*, menyatakan tipe dari paket tersebut. Nilai *30 c4* menyatakan panjang dan nilai 70-6c menyatakan nama *file* adalah pgpsc.dll. *File* ini tidak persis sama dengan standar OpenPGP untuk paket data literal, hal inilah yang menyebabkan kesalahan pada program PGP.

Kenyataan bahwa PGP menggunakan DLL yang terletak pada direktori saat ini (*current directory*) juga merupakan *bug* tetapi bukan kesalahan yang berhubungan dengan kriptografi. Hal ini merupakan kelemahan sistem operasi Windows yang juga diketahui oleh Microsoft. Serangan ini menunjukkan bahwa keamanan PGP sangat bergantung juga pada keamanan sistem operasi.

### 6.6 Serangan *Multiple User ID*

Serangan ini ditemukan oleh Sieuwert van Otterloo pada bulan Juli 2001. *Bug* ini berdasarkan pada metode pembuatan kunci dengan banyak user ID. Serangan ini mengeksploitasi kesalahan kecil yang terjadi pada antarmuka grafis PGP versi 5. Perbaikan untuk serangan ini dikeluarkan pada tanggal 4 September 2001.

Asumsi setiap kunci hanya memiliki satu *user* ID. Akan tetapi pada PGP hal ini ternyata tidaklah benar, satu kunci dapat memiliki beberapa *user* ID. Seseorang tidak dapat lagi menyatakan validitas suatu kunci, tetapi banyak orang dapat dapat melakukannya dengan tidak bergantung pada orang lainnya. Misalkan suatu kunci memiliki dua user ID, yaitu “Alice <alice@home.com>” dan “Alice <alice@hotmail.com>”. Seseorang dapat menyatakan bahwa kunci tersebut valid jika Alice sebagai pemilik kunci memang memiliki kedua alamat email tersebut. Akan tetapi jika pembuat kunci tersebut tidak memiliki kedua alamat email tersebut maka kunci tersebut tidaklah valid.

Hal ini akan menjadi menarik jika pemilik kunci hanya memiliki salah satu alamat email tersebut dan meneyebut dirinya sebagai “Alice”. Satu pasangan kunci dan *user* ID valid dan yang lainnya tidak valid. Oleh karena itu dalam permasalahan ini tidaklah berarti jika berbicara mengenai validitas suatu kunci.

PGP tidak menggunakan konsep kuci secara internal, begitu juga pada standar OpenPGP. Setiap tanda tangan hanya valid untuk satu user ID. Akan tetapi antarmuka grafis (GUI) telah mencoba untuk memudahkan penggunaanya. Pada antarmuka tersebut terdapat tombol hijau atau abu-abu yang menandakan validitas suatu kunci. PGP menggunakan strategi heuristik yang berbeda untuk berkomunikasi dengan user, antara lain:

1. Strategi pertama adalah dengan menganggap user ID pertama sebagai user ID yang paling penting dan digunakan sebagai properti kunci yang utama. PGP akan menggunakan user ID pertama ini sebagai nama kunci.
2. Strategi kedua adalah dengan menggunakan user ID yang paling valid. PGP akan mengabaikan user ID yang tidak diketahuinya.

PGP menggabungkan kedua strategi ini sehingga dapat membingungkan pengguna. Untuk melakukan eksploitasi terhadap celah ini maka sebuah kunci dengan dua user ID dibutuhkan. Celah ini terdapat pada PGP versi 5, 6, dan 7. Serangan ini merupakan serangan yang memiliki tingkat keberhasilan tinggi tanpa usaha yang cukup berarti karena seorang penyerang (*attacker*) tidak membutuhkan kekuatan komputasi, yang diperlukan hanyalah sedikit keberuntungan dan kecerobohan.

## 5. Kesimpulan

Hingga saat ini, belum ditemukan metode untuk memecahkan enkripsi PGP secara kriptografis. Pada tahun 1996, Bruce Schneier menyatakan bahwa PGP merupakan metode enkripsi kriptografi terdekat dengan metode enkripsi yang digunakan oleh militer. Berbeda halnya dengan sistem keamanan atau protokol SSL (*Secure Socket Layer*) yang hanya mengamankan data pada saat dikirimkan melalui jaringan, PGP dapat melindungi data dalam jangka waktu yang panjang seperti pada *hard disk*.

Tingkat keamanan pada enkripsi dengan PGP bergantung pada asumsi bahwa algoritma yang digunakan hingga saat ini belum dapat dipecahkan dengan cara kriptanalisis secara langsung dengan teknik dan peralatan yang ada saat ini. Sebagai contoh, pada algoritma RSA yang digunakan utnuk mengenkripsi kunci sesi, tingkat keamanannya bergantung pada teknik faktorisasi bilangan bulat secara matematis. Demikian juga dengan algoritma IDEA yang digunakan untuk mengenkripsi kunci rahasia, yang hingga saat ini belum ditemukan teknik untuk melakukan kriptanalisisnya.

Untuk mencari semua bug dan celah keamanan pada *source code* suatu program yang besar merupakan sesuatu yang sangat sulit. Dengan hanya membaca *source code* dari awal sampai akhir bukanlah merupakan metode yang efektif. Metode yang dapat digunakan adalah dengan membaca *source code* dengan cepat, membaca seluruh komentar, dan mempelajari beberapa bagian yang dianggap menarik. Dengan cara demikian maka dihasilkan hal-hal berikut:

1. Implementasi hingga tingkat yang lebih detil pada beberapa algoritma kriptografi. Hal ini menunjukkan bahwa PGP melakukan beberapa hal secara tepat, misalnya pada bagian algoritma kunci publik. Deskripsi pada petunjuk detil (*manual*) mengenai PGP memberikan banyak petunjuk kepada *programmer*. Hal ini sangat penting karena dapat mempengaruhi tingkat keamanan aplikasi.
2. Beberapa *bug*, perbaikan, dan sesuatu yang tidak biasa pada kode. Hal ini juga memungkinkan ditemukannya lebih banyak lagi celah keamanan pada PGP.

Hal yang menarik dan berkesan adalah tim pengembang aplikasi PGP sangat memahami apa yang mereka lakukan karena setiap kode sudah diperiksa lebih dari satu kali dan terlihat pula usaha untuk membuat PGP menjadi program yang aman. Akan tetapi PGP juga memiliki beberapa celah keamanan seperti pada program lainnya. Pengguna PGP sebaiknya tetap berhati-hati saat menggunakan program ini meski tingkat keamanan pada PGP sudah sangat teruji dengan baik. Walaupun demikian, tingkat keamanan pada PGP bukanlah sesuatu yang absolut.

## DAFTAR PUSTAKA


[1] *An Introduction to Cryptography*. (2002). PGP Corporation.

[2] Callas, J, L Donnerhacke, H Finney, andRThayer. (1998). *OpenPGP Message Format*. RFC 2440 Edition.

[3] Carlisle Adams. (1997). *The CAST-128 Encryption Algorithm*. RFC 2144 Edition.

[4] Derek Atkins, W Stallings, and Philip Zimmermann. (1996). *PGP Messages Exchange Format*. RFC 1991 Edition.

[5] Dyson, Jay D. (1999). *Public Key Cryptography & PGP*. Computer Systems Specialist.

[6] Garfinkel, Simson. (1995). *PGP:Pretty Good Privacy.* O'Reilly and Associates.

[7] Otterloo, van Sieuwert. (2001). *A Security Analysis of Pretty Good Privacy*. Bluering, The Netherlands.

[8] Menezes, Alfred , Paul C. van Oorschot, and Scott A. Vanstone. (1996). *Handbook of Applied Cryptography*. CRC Press.

[9] Munir, Rinaldi. (2006). *Diktat Kuliah IF5054 Kriptografi*. Program Studi Teknik Informatika, Sekolah Teknik Elektro dan Informatika, Institut Teknologi Bandung.

[10] *PGP 8.0 User's Guide for Windows*. (2002). PGP Corporation.

[11] Schneier, Bruce. (1996). *Applied Cryptography*. John Wiley and Sons, 2 Edition.