## PGP, Pretty Good Privacy

- **General**
- Created 1991 by Philip Zimmerman (a former political activist irritated by restricting freedom of using encryptation)
- Uses IDEA, a symmetric very strong cryptoalgorithm, to encrypt data.
- RSA is used to exchange a session key for IDEA.
- PGP was under US export restrictions because of RSA patent in the US. This was solved by the company Via Crypt, which got a license to sell PGP in the USA. 1996 the PGP license was purchased by PGP Inc.
- There is also MIT free version of PGP for US citizens who had an RSA license.
- PGP is currently used world wide and is believed to provide security which governments cannot break.

# PGP, Pretty Good Privacy

- **Encryptation**
- Originally in PGP version 1.0 data was encrypted with Zimmerman's own Bass-O-Matic cryptoalgorithm, which was broken very easily. Bass-O-Matic was replaced by IDEA.
- IDEA originated as PES by Xuejia Lai and James Massey in 1990, later it was called IPES and finally IDEA.
- IDEA is thought to be very strong. It is a symmetric cryptoalgorithm. like DES. It operates on blocks of 64 bits using XOR, addition modulo $2^{16}$ and multiplication modulo $2^{16}-1$. Key size is 128 bits.
- IDEA divides the 64 bit block into 4 subblocks, has 8 rounds and is not a Feistel network. It is patented but no fee is required from using it. RSA is used to exchange IDEA keys.
- RSA with the key lengths used in PGP is weaker than IDEA .

# PGP, Pretty Good Privacy

- **PGP key rings**
- A key ring is a file used by the PGP binary where the public keys are stored. It is possible to have several key rings, but they do not work well. It is better to have one key ring and add and remove public keys there.
- When a user for instance sends PGP mail, the recipient’s public key is taken form the key ring.
- **Public key rings**
- Other people’s public keys and trust levels assigned to them by you. A public key is trusted only if it has sufficient trust levels.
- **Secret key rings**
- This is the secret data, it is usually only your own secret keys. The data is encrypted and the passphrase is needed to open the encryptation.

## PGP, Pretty Good Privacy

- **Web of trust**
- PGP does not assume a Public Key Infrastructure, like the X.500 or LDAP directory with X.509-type certificates.
- Instead, you can sign other user’s public keys. You can set the amount of trust that you place on the user.
- PGP asks the questions:
- Based on your first hand knowledge do you solemnly certify that this public key belongs to the user X? (yes/no)
- This is a relatively easy question, if you know the user, you often can say if he is he. The next question is more tricky:
- Do you trust this person to act as an introducer and certify other people’s public keys for you?
- I don’t know, No, Usually, Always

## PGP, Pretty Good Privacy

- Notice,  just how unscalable this web of trust is.
- You cannot send PGP mail with assumed security level to a user unless you trust his public key enough.
- If you personally know the receivers you can trust public keys that they gave to you.
- If you do not know personally all receivers so well that you can solemnly certify their public keys, you must trust other people’s judgements.
- Basically when would you trust another person to act as an introducer, it means that you trust that never in the future he will cheat by falsifying a public key and never in the future he can be cheated to certify a false key.
- If you could sent PGP mail to anybody, everybody is trusted, then who are the ones to protect against?

## PGP, Pretty Good Privacy

- **How to use PGP? Before using PGP:**
  - Get the PGP binary. You will have to trust the PGP binary, so it should be the correct one.
  - Create the PGPPath directory, this is the directory where you keep PGP specific files
  - Set the PGPPATH variable, if you set the variable to PGPPath directory, PGP will use it, else it will use the current directory in other operating systems than Unix, in Unix it will use $HOME/.pgp by default. You must create the .pgp directory.
  - Choose a passphrase. It should be long and complex. recommended 8-10 characters, created from a sentence with some separators and words transformed in some way. 8 characters of only letters and numbers is roughly
  - $62^8 \approx 10^{14}$ about 7 years by brute force (1 million/second)

# PGP, Pretty Good Privacy

- **Generating a PGP key**
- You write **pgp -kg**
- PGP starts to generate RSA keys. It lest you to choose between key length.
- PGP asks for a user ID, it is recommended to be your name and email address. You can give any user ID, so it is trivial to create false public keys for any users.
- You need to give the passphrase.
- Then PGP creates random numbers by measuring times between keyboard hits. You have to hit the keyboard for 784 random bits. Key generation takes a long while.
- RSA needs two prime numbers which are obtained by starting from a random number, checking it for primality and decreasing by one until you hit a prime number.

## PGP, Pretty Good Privacy

- **Distributing the public key**
- When the key has been generated. it is in the public key ring. Give the command
- **pgp -kvc userID**
- This creates a fingerprint. Then give the command
- **pgp -kxa userID key-file**
- this extracts the public key from the public key ring.
- Then you can send the public key to other people via any way, like email, finger, public keyservers or any other way. A safe way is to give the key on a floppy to the  person and hope nobody switches his floppy to some falsified public key floppy.
- There is no PKI (Public Key Infrastructure), so the way to distribute your public key is one of the weak points of PGP.

## PGP, Pretty Good Privacy

- **Signing a message**
- By signing a message you can later show that you sent the message and it is not tampered with, i.e., integrity is not violated (since if the message is modified, the signature is no longer valid)
- **pgp -sat message** (Your passphrase is needed here.)
- Signing is also used to make messages, which cannot be repudiated, that is, you cannot deny that you created the message. There are mixed feelings about non-repudiation, what if there is some unknown gap in the cryptosystem.
- RSA signatures satisfy both properties, as do DSA signatures. We can create cryptosystems where signatures can be shown to be correct by the author and which guarantee integrity, but do not provide non-repudiation.

## PGP, Pretty Good Privacy

- **Adding someone else’s key**
- Before you can use PGP you must add the public keys of the other parties to your key ring.
- **pgp -ka userPubKeyFile**
- Here key userPubKeyFile is a file containing the public key in ASCII. The key will be added to the key ring.
- The keys in the key ring can be signed, or unsigned. You can sign them yourself if you trust in the public key.
- The public key ring can be easily modified. PGP will not alarm if the content of the key ring is changed, like by falsifying a public key or by modifying the trust settings.
- The user of PGP will see a public key for
- <user, email address> and will not easily notice modifications.
- The key ring is one of the weak point of PGP.

## PGP, Pretty Good Privacy

- **Encrypting a message**
- Encrypting a message to a user means sending him a random session key for IDEA by which the data is encrypted. The session key is sent encrypted by the user's public key.
- **pgp -eat message userID**
- If the public key is not certified by a signature, pGP asks you if you want still to use the public key. If you want to send it, you get **message.asc**, which is sent to the recipient.
- **Decrypting and verifying a message**
- **pgp -m message.asc**
- This command prints the message on the screen. Assuming, that the user wants to read the message later, he could use
- **pgp message.asc** (decrypts and prints on a file)

# PGP, Pretty Good Privacy

- **Clearsigning, Detached signing**
- Clearsigning means a signature which is connected with a text, which is clear text. **pgp -sat message**
- Detached signing means a signature, which is stored separately.
  **pgp -sba text file**
- **Basic message operations in PGP**

  **pgp -c text file** Encrypts with IDEA only

  **pgp -s text file [your userID]** Signs with your secret key

  **pgp -e text file her_userID [other userIDs]** Encrypts with receiver's RSA key and IDEA.

  **pgp -es text file her_userID [other userIDs]** Signs text with your secret key, encrypts with receiver's public key.

  **pgp cipherfile [plaintext file]** decrypts and checks signature

## PGP, Pretty Good Privacy

- **Key generation and management**
- Operations on public keys and key rings.
- **pgp -kg [length] [ebits] [-u userid]** Generates your own RSA key pair.
- **pgp -ka keyfile [key ring]** Adds a key to your public or secret key ring
- **pgp -kx userid keyfile [key ring]** Extracts (=copies) a key from the key ring.
- **pgp -ks her_userid [-u your_userid] [key ring]** Signs somebody's public key on your public key ring.
- **pgp -kv[v] [userid] [key ring]** View the content of the key ring.
- There are more operations, the list is not complete.

## PGP, Pretty Good Privacy

- **For her eyes only (pager option)**
- This option tries to stop you from storing very sensitive messages on files. The message is encrypted only on the screen. By making a screen dump, it could be stored. PGP cannot stop you from doing this, but helps to avoid storing messages by accident.
- **Wiping files (pgp -w)**
- If you delete a file from a disc, usually only the addressing information is deleted but the actual data stays. In MS DOS FAT (File Allocation Table) is changed, but the data can be recovered with enough work by assembling the segments. PGP writes the data over with random bits before deleting it.
- Now, is this enough, is it not so that to really remove data from a disc you must write it over at least eight times?

## PGP, Pretty Good Privacy

- **Security of PGP**
- There are many known attacks against PGP.
- Attacks against cryptoalgorithms are not the main threat, but let us discuss it first.
- IDEA is considered strong, and while cryptoanalysis advances, it should be strong still for some time.
- RSA may or may not be strong. There are recent rumors of possible fast factorization algorithms. I have not been able to verify if there are any basis in the rumors.
- The main threats are much more simple.
- An attacker may socially engineer himself into a web of trust, or some trustable person may change. Then he could falsify public keys. This breaks most of the security.
- PGP binaries can be corrupted when they are obtained.

## PGP, Pretty Good Privacy

- **Security of PGP**
- The **PGP binaries** can be modified in the computer.
- The **passphrase** can be obtained by a Trojan. Weak passphrases can be cracked.
- On multiuser system, access to the **secret key** can be obtained.
- If PGP is used over the network, passphrase or secret key can be sniffed, in general the connection from **keyboard to PGP binary** is one vulnerability.
- The **key ring** is unprotected and can be tampered with. The trust bits can be changed, public keys can be added.
- **Revoking keys** is not any more secure in PGP than it usually is, i.e., there is no way to be assured that revoked keys are removed as the other users may not see the revoked key data if an attacker arranges a suitable scenario. Obtaining a public key from a **key server** can lead to using a false public key.

## **AAA** (Authentication, Authorization , Accounting)

- AAA is one current work item in IETF.
- If users are charged for a service,they must be authorized (subscribing to the service) and they must be authenticated.
- Users of services are authenticated using various cryptographic methods. Passwords, one-time password lists, public key cryptography based authentication and so on are common mechanisms. An AAA server can usually use different authentication methods.
- Authorization means checking access/usage rights for services/resources. With computer networks the question is access rights, like with Kerberos. In the future it is services that are charged and whose usage should be authorized.
- Accounting is gathering data for creation of a bill. Traditionally in telecommunication accounting is made with pulse or ticket (CDR) metering of usage. (CDR used to mean Call Detail Record.)

## AAA

- There are existing or planned AAA solutions:
- IETF AAA protocols:
  - TACACS, Enhanced TACACS, TACACS+ (Cisco)
  - RADIUS (the most common AAA protocol by IETF)
  - DIAMETER (an upgrade of RADIUS)
  - COPS (a part of AAA, which is not interesting to us, it is QoS management part)
- Implementations:
- Merit RADIUS AAA Server
- BillNeat (Nokia, charging mobile users of IP)
- Ipay of HUT Dynamics (TIK’s project for Mobile IP+AAA)
- Most of the work in AAA concentrates on dial-up users. RADIUS is basically for dial-up users. Wireless networks and IP is currently one of the main targets.

## AAA

- The Generic AAA Architecture Internet Draft (AAAARCH) divides the generic AAA server into the following five components:
  - Authorization Rule Evaluation
  - Application Specific Module
  - Policy Repository
  - Event Log
  - Request Forwarding

- AAA protocol stack has the OSI-reference model layers
  - Application Specific Service Layer
  - Presentation Service Layer
  - Transaction and Session Management Service Layer
  - Reliable and Secure Transport Service Layer

## AAA

- AAA security mechanisms are quite ordinary. The main importance of AAA for this course is to stress that services need authorization of users, control of access to the services, and that accounting records may also benefit from security mechanisms, in DIAMETER the accounting records are made non-reputable by public key cryptography.
- Similar needs appear in all service architectures,like in the VHE (Virtual Home Environment), which often use AAA Brokers running AAA protocols.
- QoS architectures also commonly have AAA as a part of the design,for instance EURESCOM QUASIMODO proposed charging users by two AAA-based systems, one from British telecom and the other from DeteBerCom (Deutch Telecom).
- In addition to a secure lower layer, there is needed security mechanisms to the application layer for using services.