CODING TECHNIQUES

Departamento de Ingeniería Telemática

OpenCourseWare

PRACTICE 3: PGP

INTRODUCTION

PGP (*Pretty Good Privacy*) is a cryptosystem that is capable of providing authentication and confidentiality. These security functions are provided for the transmission of messages using electronic mail, as well as for the protection of data stored on disks (encrypted storage and safe erasure). It was invented by Phil Zimmermann in 1991 with the objective of separating the security of the user information from any governmental involvement (governmental certification authorities, cryptosystems suspected of having back doors, programs like Clipper, in which the American government had reserved the possibility of deciphering any message with a 'master password', etc.) Quoting Zimmermann: "*It's personal. It's private. And it's no one's business but yours*".

PGP was considered to be so safe, that the U.S. government prohibited exporting the software (among others things incorporating RSA, whose export was already prohibited), because the government thought that would permit the illegal exchange of information without possibility of any legal intrusion by the state.

Subsequently, a version of PGP appeared that could be obtained without problem at international level (PGPi 5.0, in 1997). It was created by a compilation of the original code which had been obtained by scanning a publication that described PGP and that was exported legally, and to apply an OCR (*Optical Character Recognizer*) afterwards (the export of source code in paper was not prohibited). The last version of "scanned PGP", was version 6.5.1i (23-9-1999). At the end of 1999, the company NAi (Network Associates, Inc) announced that they had received an export license for PGP from the American government, which marked the end of the PGPi project.

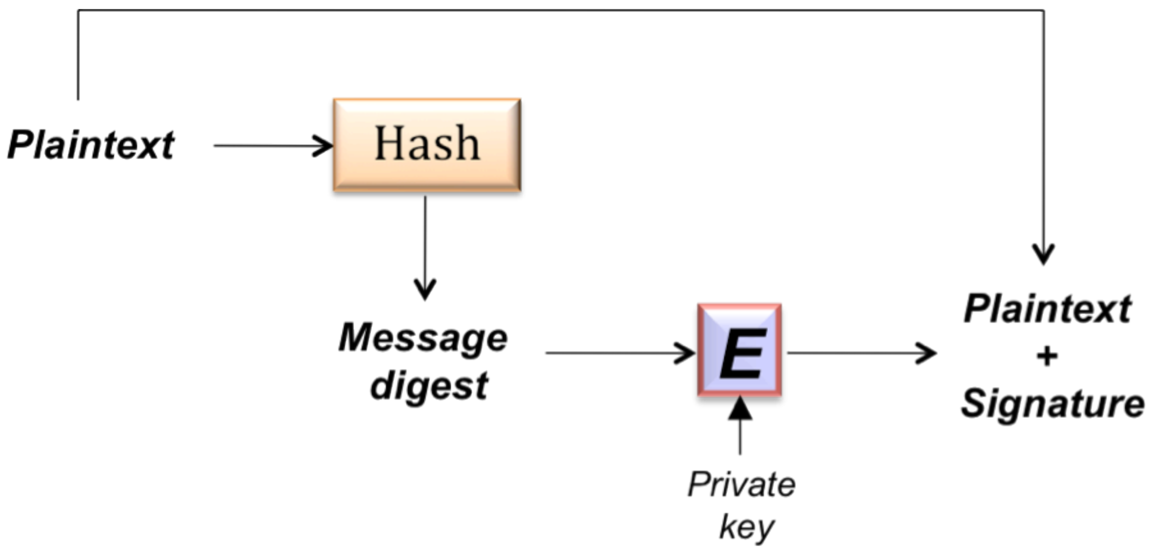
Currently PGP Corporation is the company that exploits the PGP rights. In this practice we will use the version 10.0.0 of PGP Desktop, which can be downloaded here:

http://www.pgp.com/downloads/desktoptrial/desktoptrial2.html

PRINCIPLES

The ideas of symmetric encryption (with session keys) or asymmetric encryption (with public and private keys) are concepts invented quite some time ago. However, PGP introduced a novel hybrid scheme which made it possible to combine the advantages of both systems conserving an adequate degree of security. This scheme takes advantage of the efficiency of the symmetric encryption and the key distribution facilities provided by public key encryption.

On the authentication level, PGP proposes the conventional mechanism that we see illustrated in the following figure: a scheme for digital signatures, based on a hash function (MD5 or SHA-1 as of the version 5) and a signature algorithm with a private key (RSA or, as for PGP version 5, DSS, according to the NIST standard).

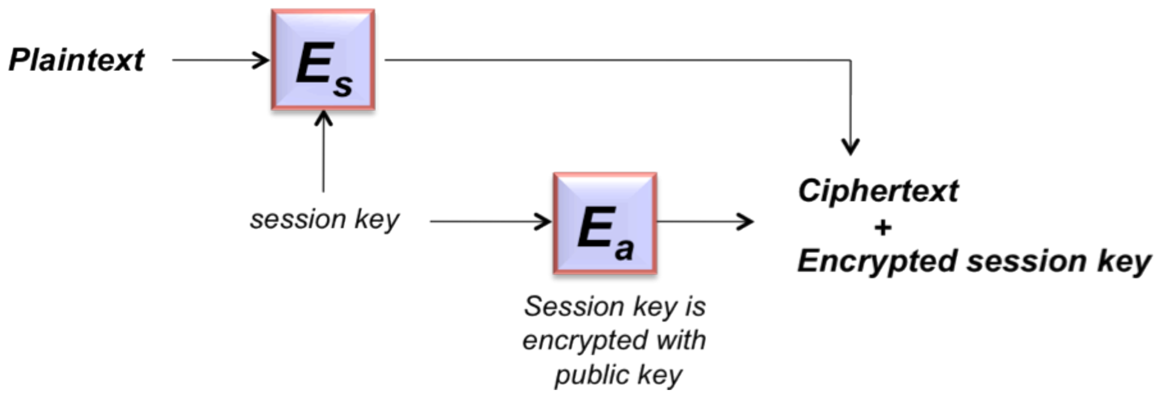


As already mentioned, the true innovation with regard to the previous cryptosystems was combining the conventional encryption with a system using private and public keys. The session key permits one to encrypt the files and to send messages (utilizing symmetrical algorithms such as IDEA, CAST, Triple DES, AES or Twofish in mode CFB), in a quicker and more efficient way compared to using an asymmetric scheme directly.

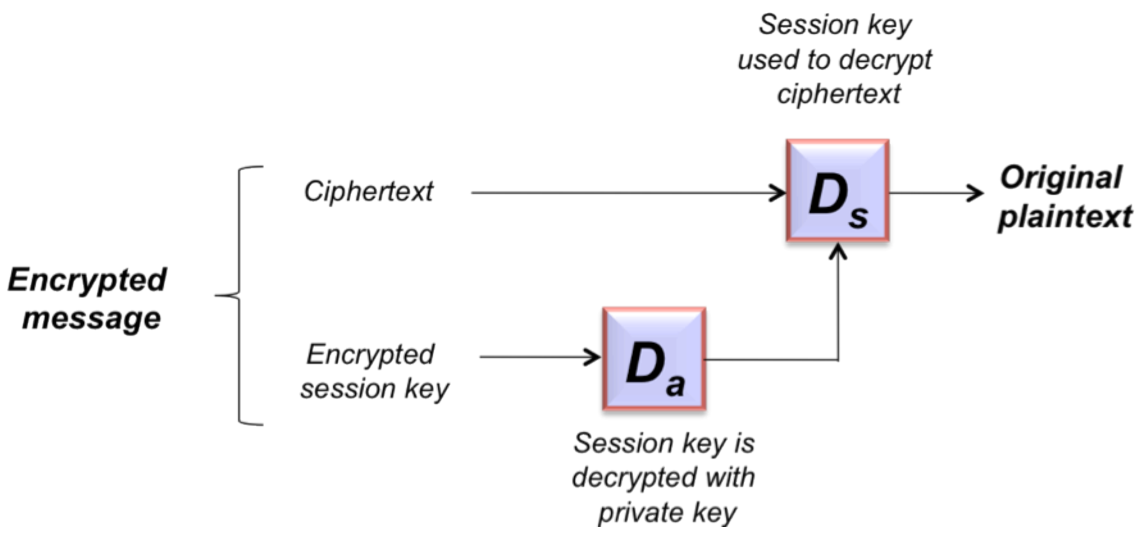
Therefore, if we want to transmit a session key securely, it will be enough to encrypt that key by means of RSA or ElGamal, using the public key of the recipient. Moreover, with a view towards providing authenticity with digital signatures, we can use the private key of the transmitter for both schemes, RSA or DSS.

Thus, the unique thing a user needs (besides the PGP software for key generation and encryption), is to know the public key of the users to which data should be sent securely. The problem of public key distribution remains to be solved (in addition to the possibility of learning them from a classical key directory), which is done using a hierarchical network of trust. Studying them constitutes one part of this practice.

PGP ENCODER


Plaintext
E_s
session key
E_a
Ciphertext + Encrypted session key
Session key is encrypted with public key

PGP DECODER


Session key used to decrypt ciphertext
Encrypted message
Ciphertext
D_s
Original plaintext
Encrypted session key
D_a
Session key is decrypted with private key

PRACTICE AND QUESTIONS

Start *PGP Desktop* from the *PGP Tray*. Using this application, generate a pair of private and public key (DH and DSS), selecting the corresponding option in the menu **Keys** (DO NOT send them to any key server).

1. **Answer with reasons:**
   a) **How many keys has the application generated? What is the purpose of each of these key?**
   b) **How does PGP guarantee the uniqueness of the keys?**
   c) **What implications do a greater or smaller length of the key have?**
   d) **Besides the previous key, PGP utilizes two further ones. When are they generated and for what purpose are they used?**

- If our mail tool would permit it, we were able to write a message and encrypt/sign it directly with that tool before sending it (*Eudora, Pegasus,* or *Outlook* have a plug-in for PGP for that purpose). In our case, however, we are going to do the operation manually, in order to understand the complete process.

2. **Sign a text file, explaining the different modalities of the signature. Furthermore, validate the signature for all cases, explaining in detail each of the processes (comment them also from the cryptographic point of view).**

- The following task is to incorporate the public key of those people into our public key file (in PGP is called *ring*), to whom we want to send something confidentially. Therefore, we will use PGP's facility of importing and exporting keys.

3. **Export the public key to a file, in order to provide it to the partner, and detail the steps you have made. What is the reason for having the possibility to export also the private key?**

4. **Obtain a file with the public key of a partner, make use of the import function of PGPKeys, and incorporate it into your ring of public keys. Show in detail the steps you have followed. You can observe that the column "*Verified*" has a different meaning compared to the one that has the key that was generated before. What's the reason?**

- From now on, we are able to encrypt a document (i.e., either encrypting or signing it).

5. **Try to encrypt any document (the executable *PGP.exe*, for example) with the key that you have just imported, describing the necessary steps. What is the purpose of each encryption the application offers?**

6. **Try now to recover the encrypted file. Comment everything that is necessary for that purpose.**

7. **Suppose that a user wants to encrypt a document with PGP in order to send it to a receiver, which does not utilize the PGP software. Indicate with reasons whether this is possible or not, and if yes, in what situations it is possible.**

8. **If we receive a signature of the same message two times back-to-back, are the signatures the same? And if we encrypt the same message twice, are the encrypted messages the same? Try to give the reason for each case.**

We will now try to understand the scheme of hierarchical networks of trust that PGP offers. This scheme is managed from PGPKeys and, as said before, has the objective of making key distribution independent from any central key server or a trusted third party, although an interaction with key distribution centers can be foreseen. The scheme is based mainly on three points:

- The `Verified` field: indicates the level of confidence that the signature actually belongs to the person and to the mail which appears in the *Keys* field
- The `Trust` field: indicates the degree of trust (great or little) for someone who can act as representative for others.
- The *signers* of a key (indicated with a *pencil*): indicates the persons that consider that key valid (initially the only signature appearing here is, obviously, your own signature).

• In order to analyze these options, import first into the keyring of trust the public keys of some companions (at least two companions which will be named companion1 and companion2 in the following). The key of the companion2 should be signed by companion1.

9. If we encrypt a document with one of these keys, from what point of view is such a mechanism secure and from what one it is not secure? Explain whether it is useful or not to have a key signed by a third person for the encryption process.

• At the time of encryption, is possible to select various recipients.

10. Encrypt a file for companion1 and companion2. Observe the obtained output and explain it from a cryptographic point of view.

• Suppose now that we have called the companion1 by phone and he has dictated his *Fingerprint*, which is the same compared to the one that you can see in his public key.

11. Sign the public key of that companion.
- **a) What implication does that signature have from the point of view of trust in the companion?**
- **b) What implications does it have for the key of companion2?**
- **c) What is the Fingerprint? For what purpose is it used?**
- **d) If now we enhance the field of *trust* for companion1 to the marginal level, what will happen? What happens if we enhance it to the maximum level? Why?**