PGP for Electronic Mail Security

- PGP stands for Pretty Good Privacy. It was developed originally by Phil Zimmerman. However, in its incarnation as **OpenPGP**, it has now become an open standard. PGP is open-source.

- Although PGP can be used for protecting data in long-term storage, it is used primarily for email security.

- PGP's operation consists of **five services**:

  1. **Authentication:** Sender authentication consists of the sender attaching his/her digital signature to the email and the receiver verifying the signature using public-key cryptography. Here is an **example** of authentication operations carried out by the sender and the receiver:

     1. At the sender's end, the SHA-1 hash function is used to create a 160-bit message digest of the outgoing email message.

     2. The message digest is encrypted with RSA using the sender's private key and the result **prepended** to the message. The composite message is transmitted to the recipient.

3. The receiver uses RSA with the sender's public key to decrypt the message digest.

4. The receiver compares the locally computed message digest with the received message digest.

The above description above was based on using a RSA/SHA based digital signature. PGP also support DSS/SHA based signatures. DSS stands for **Digital Signature Standard**. The above description was also based on attaching the signature to the message. PGP also supports **detached signatures** that can be sent separately to the receiver. Detached signatures are also useful when a document must be signed by multiple individuals.

2. **Confidentiality:** This service can also be used for encrypting disk files. As you'd expect, PGP uses symmetric-key encryption for confidentiality. The user has the choice of three different block-cipher algorithms for this purpose: CAST-128, IDEA, or 3DES, with CAST-128 being the default choice. [Like DES, **CAST-128** is a block cipher that uses the Feistel cipher structure (see lecture notes on "Block Ciphers and the Data Encryption Standard" for what is meant by a Feistel structure). The block size in CAST-128 is 64-bits and the key size varies between 40 and 128 bits. Depending on the key size, the number of rounds used in the Feistel structure is between 12 and 16, it being the latter when the key size exceeds 80 bits. Obviously, as you'd expect, how each round of processing works in CAST is different from how it works in DES. But, overall, as in DES, each round

carries out a series of substitutions and permutations in the incoming data. **IDEA** (International Data Encryption Algorithm) is also a block cipher. IDEA uses 64-bit blocks and 128 bit keys. The cipher uses 8 rounds of processing on the input bit blocks (and an additional half round), each round consisting of substitutions and permutations.]

- The block ciphers are used in the **Cipher Feedback Mode** (CFB) explained in the lecture notes entitled "Using Block Ciphers ...".

- The 128-bit encryption key, called the **session key**, is generated for each email message separately.

- The session key is encrypted using RSA with the receiver's public key. Alternatively, the session key can also be encrypted using the **ElGamal** algorithm. ElGamal is a variant of the Diffie-Hellman that also allows for encryption and decryption. (Diffie-Hellman only does secret key exchange.) (See lecture notes entitled "Secret Key Exchange ...." for Diffie-Hellman.)

- What is put on the wire is the email message after it is encrypted with the session key and the session key after it is encrypted with the receiver's public key.

- If confidentiality and sender-authentication are needed simultaneously, a digital signature for the message is generated using the hash code of the message plaintext and appended to the email message before it is encrypted with

the session key. (See the previously shown PGP's authentication service.)

3. **Compression:** By Default PGP compresses the email message after applying the signature but before encryption. This is to allow for long-term storage of uncompressed messages along with their signatures. This also decouples the encryption algorithm from the message verification procedures. Compression is achieved with the ZIP algorithm.

4. **E-Mail Compatibility:** Since encryption, even when it is limited to the signature, results in arbitrary binary strings, and since many email systems only permit the use of ASCII characters, we have to be able to represent binary data with ASCII strings. PGP uses **radix 64** encoding for this purpose. [Radix 64 encoding, also known as **Base-64 encoding** has emerged as probably the most common way to transmit binary data over a network. It first segments the binary stream of bytes (the same thing as bytes) into 6-bit words. The $2^6 = 64$ different possible 6-bit words are represented by printable characters as follows: The first 26 are mapped to the uppercase letters A through Z, the next 26 to the lowercase a through z, the next 10 to the digits 0 through 9, and the last two to the characters / and +. This causes each triple of adjoining bytes to be mapped into four ASCII characters. The Base-64 character set includes a 65^{th} character, '=', to indicate how many characters the binary string is short of being an exact multiple of 3 bytes. When the binary string is short one byte, that is indicated by terminating the Base-64 string with a single '='. And when it is short two bytes, the termination becomes '=='.]

5. **Segmentation:** For long email messages (these are generally messages with attachments), many email systems place restrictions on how much of the message will be transmitted as a unit. For example, Some email systems segment long email messages into $50,000$ byte segments and transmit each segment separately. PGP has built-in facilities for such segmentation and re-assembly.

- The figure shown on the next slide shows the three different modes in which PGP can be used for secure email exchange. The top diagram is for when only authentication is desired, the middle when only confidentiality is needed, and the bottom when both are wanted. The symbols EP and DP in the figure stand for public-key encryption and decryption; the symbols EC and DC for symmetric-key encryption and decryption; the symbol K_s for the session key for symmetric-key encryption/decryption; H for hashing; $||$ for concatenation; Z for compression using the ZIP algorithm, and $R64$ for conversion to radix 64 ASCII format.

- The figure shown on Slide 10 presents another look at the working of the PGP algorithm at the sender and the receiver ends of an email exchange. Various PGP modules come into play depending on what services a user requests from PGP.

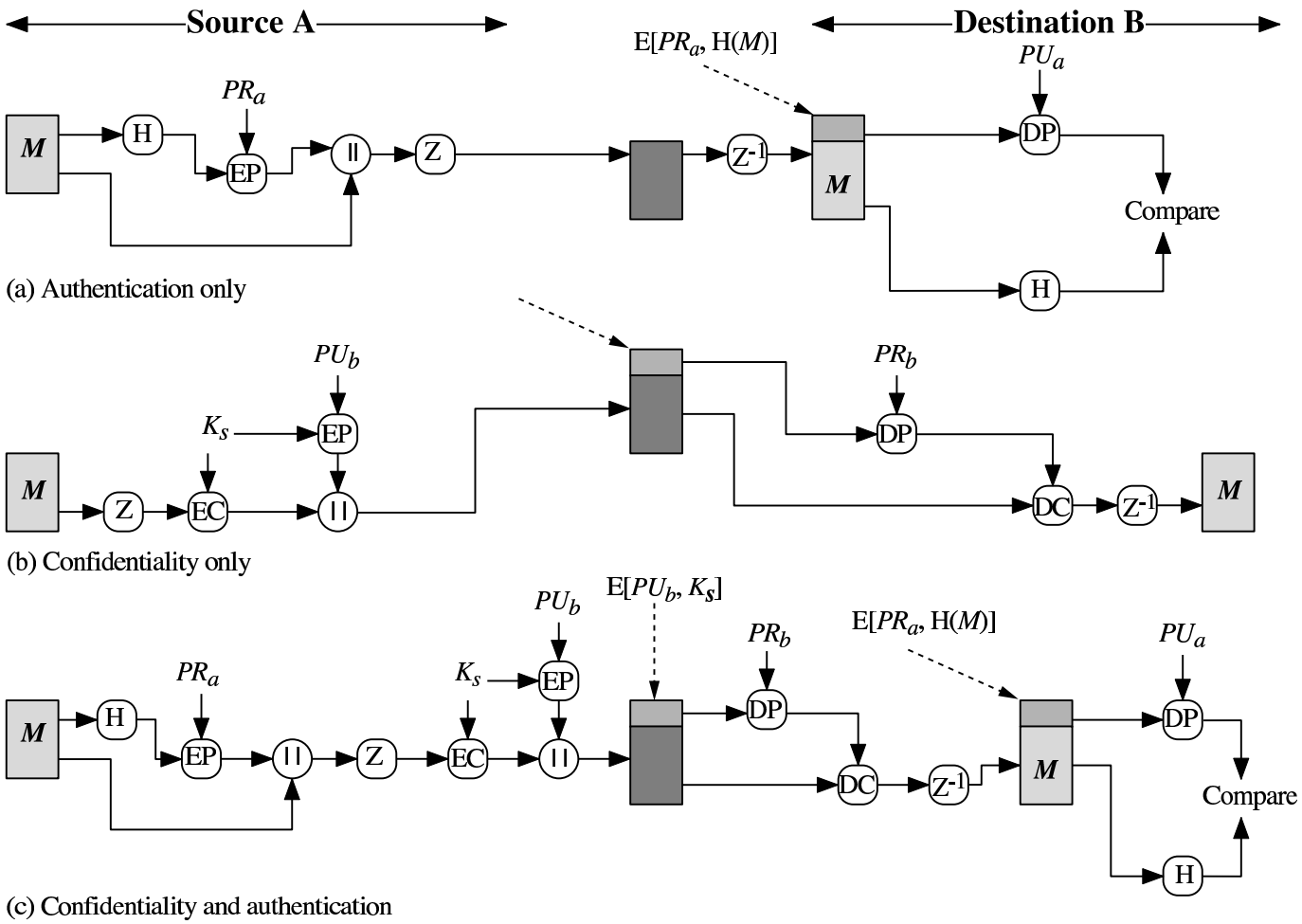
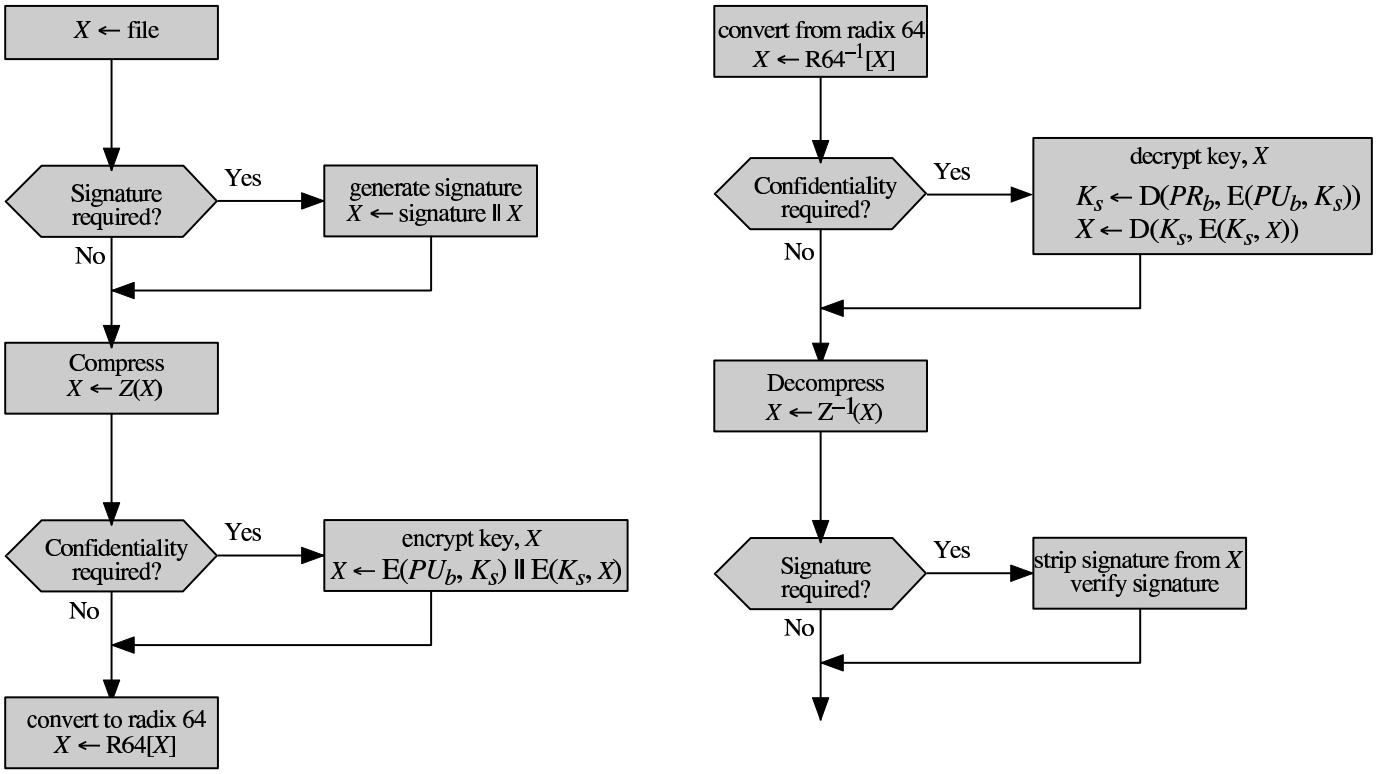


Figure 15.1 PGP Cryptographic Functions

This figure is from Chapter 15 of Stallings: "Cryptography and Network Security", Fourth Edition



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

Figure 15.2 Transmission and Reception of PGP Messages

This figure is from Chapter 15 of Stallings: "Cryptography and Network Security", Fourth Edition

Key Management Issues in PGP

- As you have already seen, public key encryption is central to PGP. It is used for two purposes: sender uses his/her private key for placing his/her digital signature on the outgoing message, and the sender uses the receiver's public key for encrypting the secret session key.

- We can expect people to have multiple public keys (and, of course, the corresponding private keys). This could happen because an individual in in the process of retiring an old public key, but, to allow for a period of transition, decides to make available both the old and the new for a while. Some people may also choose to public multiple public keys for various reasons.

- So PGP must allow for the possibility that the receiver of a message may have multiple public keys. This raises the following procedural questions:

  - Let's say PGP uses one of the public keys made available by the recipient, how does the recipient know which public key it is?

  - Let's say that the sender uses one of the multiple private keys that the sender has at his/her disposal for signing the message,

how does the recipient know which of the corresponding public keys to use?

- Both of these problems can be gotten around by the sender also sending along the public key used. The only problem here is that it is wasteful in space **because the RSA public keys can be hundreds of decimal digits long.**

- The PGP protocol solves this problem by using the notion of a relatively short **key identifiers** (key ID) and requiring that every PGP agent maintain its own list of private/public keys, along with their key identifiers, and a list of public keys, along with their associated key identifiers, for all the email correspondents. **The former list is known as the private key ring** and the latter **as the public key right**. Examples of a private key ring and a public key ring are shown in the figure on Slide 16.

- The keys for a particular user are uniquely identifiable through a combination of the user ID and the key ID.

- The key ID associated with a public key consists of its least significant 64 bits.

- Going back to private key ring shown in the figure on Slide 16, for security reasons, PGP stores the private keys in the table in

an encrypted form so that the keys are only accessible to the user who owns them. PGP can use any of the three block ciphers at its disposal, CAST-128, IDEA, and 3DES, with CAST-128 serving as the default choice, for this encryption. The encryption algorithm asks the user to enter a pass-phrase. **The pass-phrase is hashed with SHA-1 to yield a 160-bit hash code.** The first 128 bits of the hash code are used as the encryption key by the CAST-128 algorithm. **Both the pass-phrase and the hash code are immediately discarded.**

- With regard to the public key ring shown in the figure on Slide 16, the fields **Owner Trust**, **Key Legitimacy**, **Signature**, and **Signature Trust** are to assess how much trust to place in the public keys belonging to other people. [If A has B's public key in the ring, but the key really belongs to C, then C can send messages to A and forge B's signature (A would think a message was from B whereas it is really from C) and any encrypted messages from A to B would be readable by C.]

- The values in the **key legitimacy field** column are computed by PGP. This value tells PGP as to how much trust to place in the public key in the corresponding row to be a valid key for the user ID.

- The entry stored in the **public key** field is actually a certificate. The **signature(s)** field contains the signature(s) of one or more certifying authorities on the certificate. Each signature has as-

sociated with it a **signature trust field** value that indicated how much trust PGP has in the signer of the certificate. The value for the **key legitimacy field** is derived from the value(s) stored for the **signature trust field(s)**.

- The entry in the **owner trust field** of the public-key-ring table indicates the extent to which the owner of a particular public key can be trusted to sign other certificates. This value is assigned by the user to whom the public-key-ring belongs.

- The figure on Slide 17 shows the general format of a PGP message. As the figure shows, a PGP message consists of three components: a session key component, a signature component, and the actual email message itself. Perhaps the only unexpected entry is the "leading two bytes of message digest". This is to enable the recipient to determine that the correct public key (of the sender) was used to decrypt the message digest for authentication. These two octets also server as a 16-bit **frame check sequence** for the actual email message. The message digest itself is calculated using SHA-1.

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
• • •	• • •	• • •	• • •	• • •

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
T_i	$PU_i \bmod 2^{64}$	PU_i	$trust\_flag_i$	User i	$trust\_flag_i$		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

* = field used to index table

Figure 15.4 General Structure of Private and Public Key Rings

This figure is from Chapter 15 of Stallings: "Cryptography and Network Security", Fourth Edition

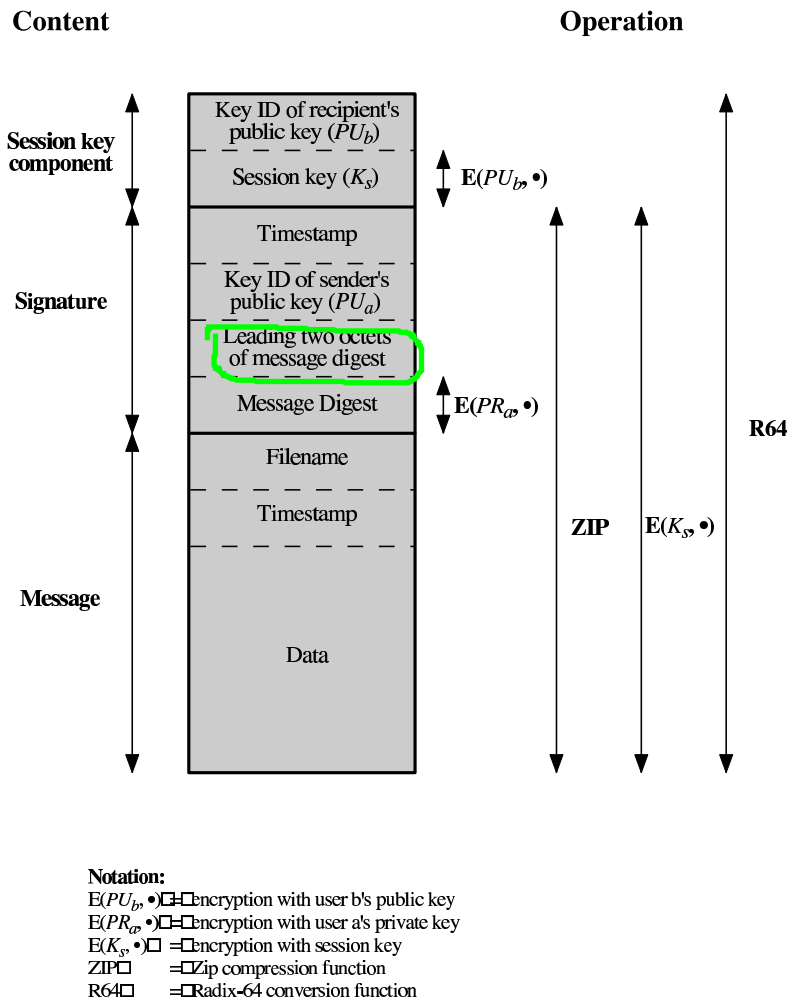


Figure 15.3 General Format of PGP Message (from A to B)

This figure is from Chapter 15 of Stallings: "Cryptography and Network Security", Fourth Edition