ELECTRONIC MAIL SECURITY PRETTY GOOD PRIVACY (PGP)

PGP provides the confidentiality and authentication service that can be used for electronic mail and file storage applications. The steps involved in PGP are

Select the best available cryptographic algorithms as building blocks.

Integrate these algorithms into a general purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.

Make the package and its documentation, including the source code, freely available via the internet, bulletin boards and commercial networks.

Enter into an agreement with a company to provide a fully compatible, low cost commercial version of PGP.

PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth.

- It is available free worldwide in versions that run on a variety of platform.
- It is based on algorithms that have survived extensive public review and are considered extremely secure.
- e.g., RSA, DSS and Diffie Hellman for public key encryption CAST-128, IDEA and 3DES for conventional encryption SHA-1 for hash coding.
- it has a wide range of applicability.
- It was not developed by, nor it is controlled by, any governmental or standards organization.

Operational description

The actual operation of PGP consists of five services: authentication, confidentiality, compression, e-mail compatibility and segmentation.

1. Authentication

The sequence for authentication is as follows:

The sender creates the message

SHA-1 is used to generate a 160-bit hash code of the message

The hash code is encrypted with RSA using the sender's private key and the result is prepended to the message

The receiver uses RSA with the sender's public key to decrypt and recover the hash code.

The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.

2. Confidentiality

Confidentiality is provided by encrypting messages to be transmitted or to be stored locally as files. In both cases, the conventional encryption algorithm CAST-128 may be used. The 64-bit cipher feedback (CFB) mode is used.

In PGP, each conventional key is used only once. That is, a new key is generated as a random 128-bit number for each message. Thus although this is referred to as **a session key**, it is in reality a **one time key**. To protect the key, it is encrypted with the receiver's public key.

The sequence for confidentiality is as follows:

- The sender generates a message and a random 128-bit number to be used as a session key for this message only.
- The message is encrypted using CAST-128 with the session key.
- The session key is encrypted with RSA, using the receiver's public key and is prepended to the message.
- The receiver uses RSA with its private key to decrypt and recover the session key.
- The session key is used to decrypt the message.

Confidentiality and authentication

Here both services may be used for the same message. First, a signature is generated for the plaintext message and prepended to the message. Then the plaintext plus the signature is encrypted using CAST-128 and the session key is encrypted using RSA.

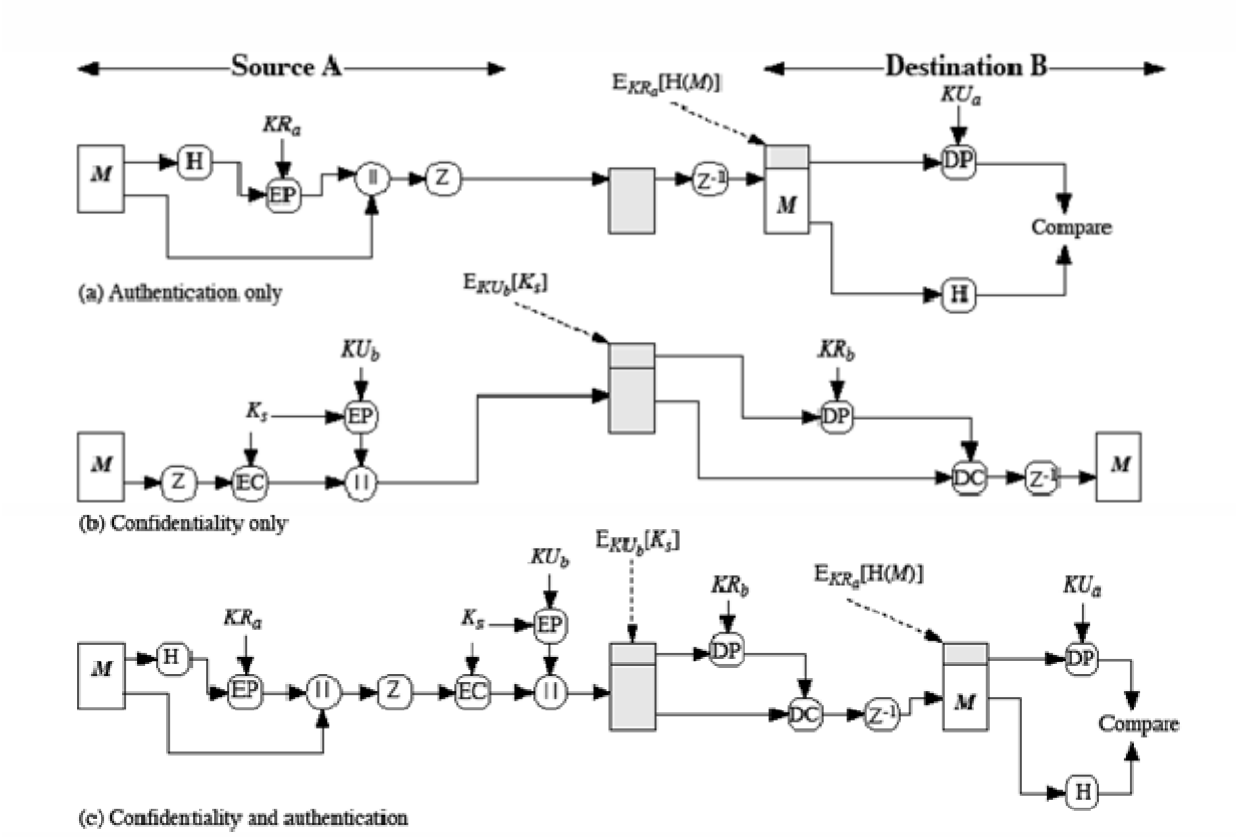


Figure 15.1 PGP Cryptographic Functions

3. Compression

As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space for both e-mail transmission and for file storage.

The signature is generated before compression for two reasons:

It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.

Even if one were willing to generate dynamically a recompressed message fro verification, PGP's compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and as a result, produce different compression forms.

Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

The compression algorithm used is ZIP.

4. e-mail compatibility

Many electronic mail systems only permit the use of blocks consisting of ASCII texts. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is **radix-64 conversion**.

Each group of three octets of binary data is mapped into four ASCII characters.

e.g., consider the 24-bit (3 octets) raw text sequence 00100011 01011100 10010001, we can express this input in block of 6-bits to produce 4 ASCII characters.

001000 110101 110010 010001

I L Y R => corresponding ASCII characters

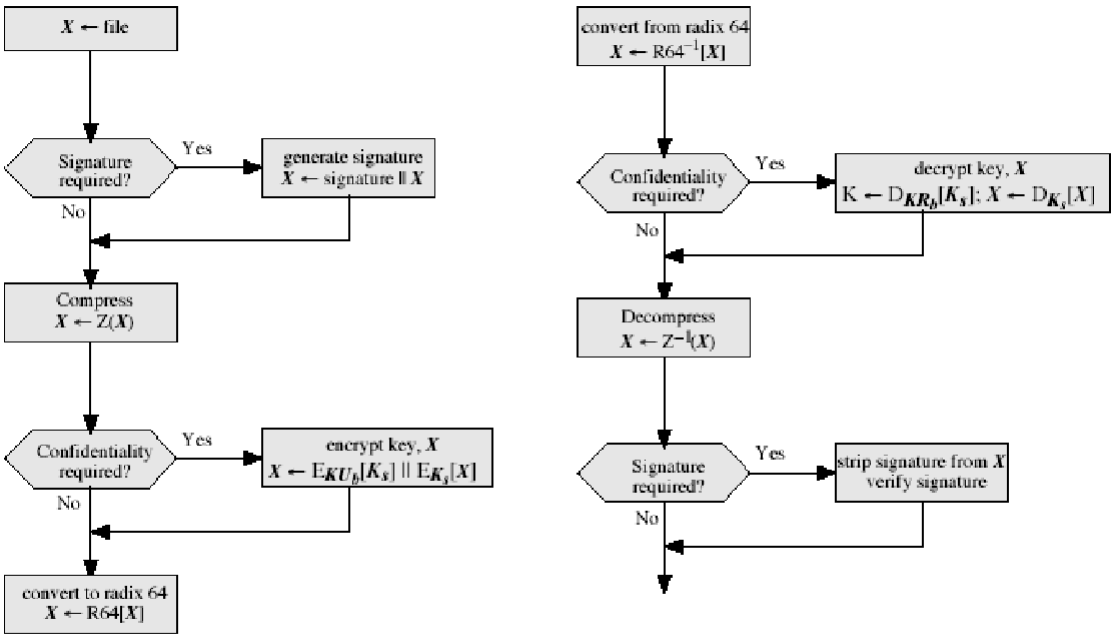
5. Segmentation and reassembly

E-mail facilities often are restricted to a maximum length. E.g., many of the facilities accessible through the internet impose a maximum length of 50,000 octets. Any message longer than that must be broken up into smaller segments, each of which is mailed separately.

To accommodate this restriction, PGP automatically subdivides a message that is too large into

segments that are small enough to send via e-mail. The segmentation is done after all the other processing, including the radix-64 conversion. At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block before performing the other steps.

PGP Operation Summary:



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

Cryptographic keys and key rings

Three separate requirements can be identified with respect to these keys:

A means of generating unpredictable session keys is needed.

It must allow a user to have multiple public key/private key pairs.

Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.

We now examine each of the requirements in turn.

1. Session key generation

Each session key is associated with a single message and is used only for the purpose of encryption and decryption of that message. Random 128-bit numbers are generated using CAST-128 itself. The input to the random number

generator consists of a 128-bit key and two 64-bit blocks that are treated as plaintext to be encrypted. Using cipher feedback mode, the CAST-128 produces two 64-bit cipher text blocks, which are concatenated to form the 128-bit session key. The plaintext input to CAST-128 is itself derived from a stream of 128-bit randomized numbers. These numbers are based on the keystroke input from the user.

2. Key identifiers

If multiple public/private key pair are used, then how does the recipient know which of the public keys was used to encrypt the session key? One simple solution would be to transmit the public key with the message but, it is unnecessary wasteful of space. Another solution would be to associate an identifier with each public key that is unique at least within each user.

The solution adopted by PGP is to assign a key ID to each public key that is, with very high probability, unique within a user ID. The key ID associated with each public key consists of its least significant 64 bits. i.e., the key ID of public key KU_a is $(KU_a \bmod 2^{64})$.

A message consists of three components.

Message component – includes actual data to be transmitted, as well as the filename and a timestamp that specifies the time of creation.

Signature component – includes the following

- Timestamp – time at which the signature was made.
- Message digest – hash code.

O Two octets of message digest – to enable the recipient to determine if the correct public key was used to decrypt the message.

- Key ID of sender's public key – identifies the public key

Session key component – includes session key and the identifier of the recipient public key.

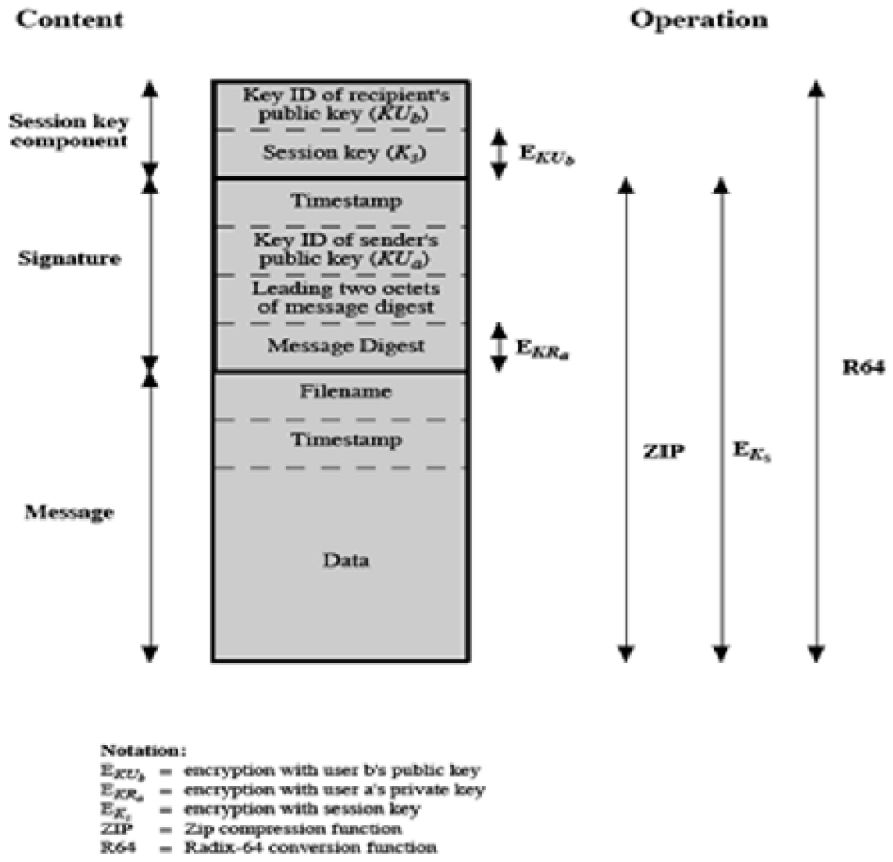


Figure 15.3 General Format of PGP Message (from A to B)

3. Key rings

PGP provides a pair of data structures at each node, one to store the public/private key pair owned by that node and one to store the public keys of the other users known at that node. These data structures are referred to as private key ring and public key ring.

The general structures of the private and public key rings are shown below: Timestamp – the date/time when this entry was made.

Key ID – the least significant bits of the public key.

Public key – public key portion of the pair. **Private key** – private key portion of the pair. **User ID** – the owner of the key.

Key legitimacy field – indicates the extent to which PGP will trust that this is a valid public key for this user.

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

Signature trust field – indicates the degree to which this PGP user trusts the signer to certify public key.

Owner trust field – indicates the degree to which this public key is trusted to sign other public key certificates.

PGP message generation

First consider message transmission and assume that the message is to be both signed and encrypted. The sending PGP entity performs the following steps:

1. **signing the message**

PGP retrieves the sender's private key from the private key ring using user ID as an index. If user ID was not provided, the first private key from the ring is retrieved.

PGP prompts the user for the passpharse (password) to recover the unencrypted private key.

The signature component of the message is constructed.

2. **encrypting the message**

PGP generates a session key and encrypts the message.

PGP retrieves the recipient's public key from the public key ring using user

ID as index.

The session key component of the message is constructed. The receiving PGP entity performs the following steps:

Decrypting the message

PGP retrieves the receiver's private key from the private key ring, using the key ID field in the session key component of the message as an index.

PGP prompts the user for the passpharse (password) to recover the unencrypted private key.

PGP then recovers the session key and decrypts the message.

2. Authenticating the message

PGP retrieves the sender's public key from the public key ring, using the key ID field in the signature key component of the message as an index.

PGP recovers the transmitted message digest.

PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate.

Public-Key Management

This whole business of protecting public keys from tampering is the single most difficult problem in practical public key applications. PGP provides a structure for solving this problem, with several suggested options that may be used.

Approaches to Public-Key Management

The essence of the problem is this: User A must build up a public-key ring containing the public keys of other users to interoperate with them using PGP. Suppose that A's key ring contains a public key attributed to B but that the key is, in fact, owned by C. This could happen if, for

example, A got the key from a bulletin board system (BBS) that was used by B to post the public key but that has been compromised by C. The result is that two threats now exist. First, C can send messages to A and forge B's signature, so that A will accept the message as coming from B. Second, any encrypted message from A to B can be read by C.

A number of approaches are possible for minimizing the risk that a user's public-key ring contains false public keys. Suppose that A wishes to obtain a reliable public key for B. The following are some approaches that could be used:

1. Physically get the key from B. B could store her public key (PU_b) on a floppy disk and hand it to A.
2. Verify a key by telephone. If A can recognize B on the phone, A could call B and ask her to dictate the key, in radix-64 format, over the phone.
3. Obtain B's public key from a mutual trusted individual D. For this purpose, the introducer, D, creates a signed certificate. The certificate includes B's public key, the time of creation of the key, and a validity period for the key.
4. Obtain B's public key from a trusted certifying authority. Again, a public key certificate is created and signed by the authority. A could then access the authority, providing a user name and receiving a signed certificate.

For cases 3 and 4, A would already have to have a copy of the introducer's public key and trust that this key is valid. Ultimately, it is up to A to assign a level of trust to anyone who is to act as an introducer.

The Use of Trust

Although PGP does not include any specification for establishing certifying authorities or for establishing trust, it does provide a convenient means of using trust, associating trust with public keys, and exploiting trust information.

The basic structure is as follows. Each entry in the public-key ring is a public-key certificate. Associated with each such entry is a key legitimacy field that indicates the extent to which PGP will trust that this is a valid public key for this user; the higher the level of trust, the stronger is the binding of this user ID to this key.

This field is computed by PGP.Also associated with the entry are zero or more signatures that the key ring owner has collected that sign this certificate. In turn, each signature has associated

with it a signature trust field that indicates the degree to which this PGP user trusts the signer to certify public keys.The key legitimacy field is derived from the collection of signature trust fields in the entry.Finally, each entry defines a public key associated with a particular owner, and an owner trust field is included that indicates the degree to which this public key is trusted to sign other public-key certificates; this level of trust is assigned by the user.

The three fields mentioned in the previous paragraph are each contained in a structure referred to as a trust flag byte.

Suppose that we are dealing with the public-key ring of user A. We can describe the operation of the trust processing as follows:

1. When A inserts a new public key on the public-key ring, PGP must assign a value to the trust flag that is associated with the owner of this public key. If the owner is A, and therefore this public key also appears in the private-key ring, then a value of ultimate trust is automatically assigned to the trust field. Otherwise, PGP asks A for his assessment of the trust to be assigned to the owner of this key, and A must enter the desired level. The user can specify that this owner is unknown, untrusted, marginally trusted, or completely trusted.
2. When the new public key is entered, one or more signatures may be attached to it.

More signatures may be added later. When a signature is inserted into the entry, PGP searches the public-key ring to see if the author of this signature is among the known public-key owners. Ifso, the OWNERTRUST value for this owner is assigned to the SIGTRUST field for this signature.If not, an unknown user value is assigned.

3. The value of the key legitimacy field is calculated on the basis of the signature trust fields present in this entry. If at least one signature has a signature trust value of ultimate, then the key legitimacy value is set to complete.

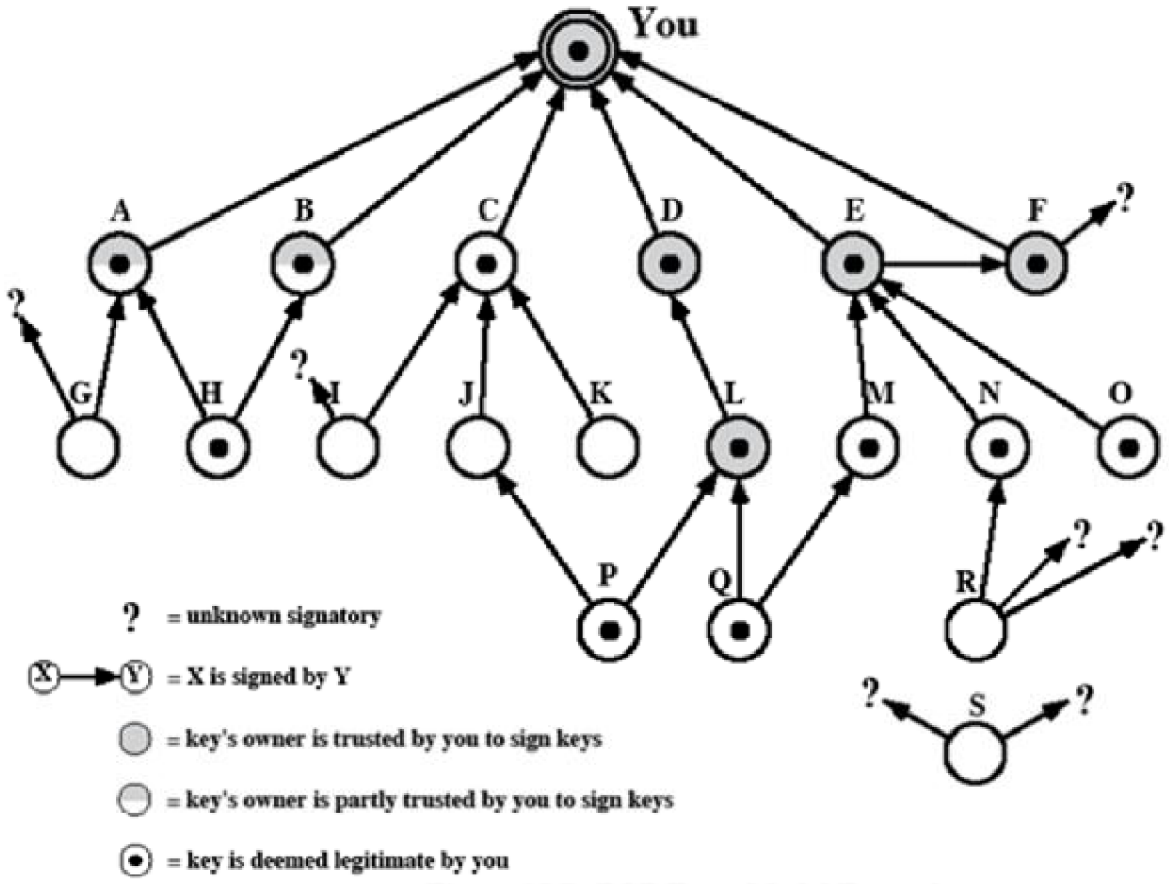


Figure 15.7 PGP Trust Model Example

The node labeled "You" refers to the entry in the public-key ring corresponding to this user. This key is legitimate and the OWNERTRUST value is ultimate trust. Each other node in the key ring has an OWNERTRUST value of undefined unless some other value is assigned by the user. In this example, this user has specified that it always trusts the following users to sign other keys: D, E, F, L. This user partially trusts users A and B to sign other keys.

So the shading, or lack thereof, of the nodes in Figure 15.7 indicates the level of trust assigned by this user. The tree structure indicates which keys have been signed by which

other users. If a key is signed by a user whose key is also in this key ring, the arrow joins the signedkey to the signatory. If the key is signed by a user whose key is not present in this key ring, the arrow joins the signed key to a question mark, indicating that the signatory is unknown to this user.

Note that all keys whose owners are fully or partially trusted by this user have been signed by thisuser, with the exception of node L.

1. We assume that two partially trusted signatures are sufficient to certify a key. Hence, the keyfor user H is deemed legitimate by PGP because it is signed by A and B, both of whom are partially trusted.
2. A key may be determined to be legitimate because it is signed by one fully trusted or two partially trusted signatories, but its user may not be trusted to sign other keys. For example, N's key is legitimate because it is signed by E, whom this user trusts, but N is not trusted to sign other keys because this user has not assigned N that trust value. Therefore, although R's key is signed by N, PGP does not consider R's key legitimate. This situation makes perfect sense. If you wish to senda private message to some individual, it is not necessary that you trust that individual in any respect. It is only necessary that you are sure that you have the correct public key for that individual.
3. Figure 15.7 also shows an example of a detached "orphan" node S, with two unknown signatures. Such a key may have been acquired from a key server. PGP cannot assume that this key is legitimate simply because it came from a reputable server. The user must declare the key legitimate by signing it or by telling PGP that it is willing to trust fully one of the key's signatories.