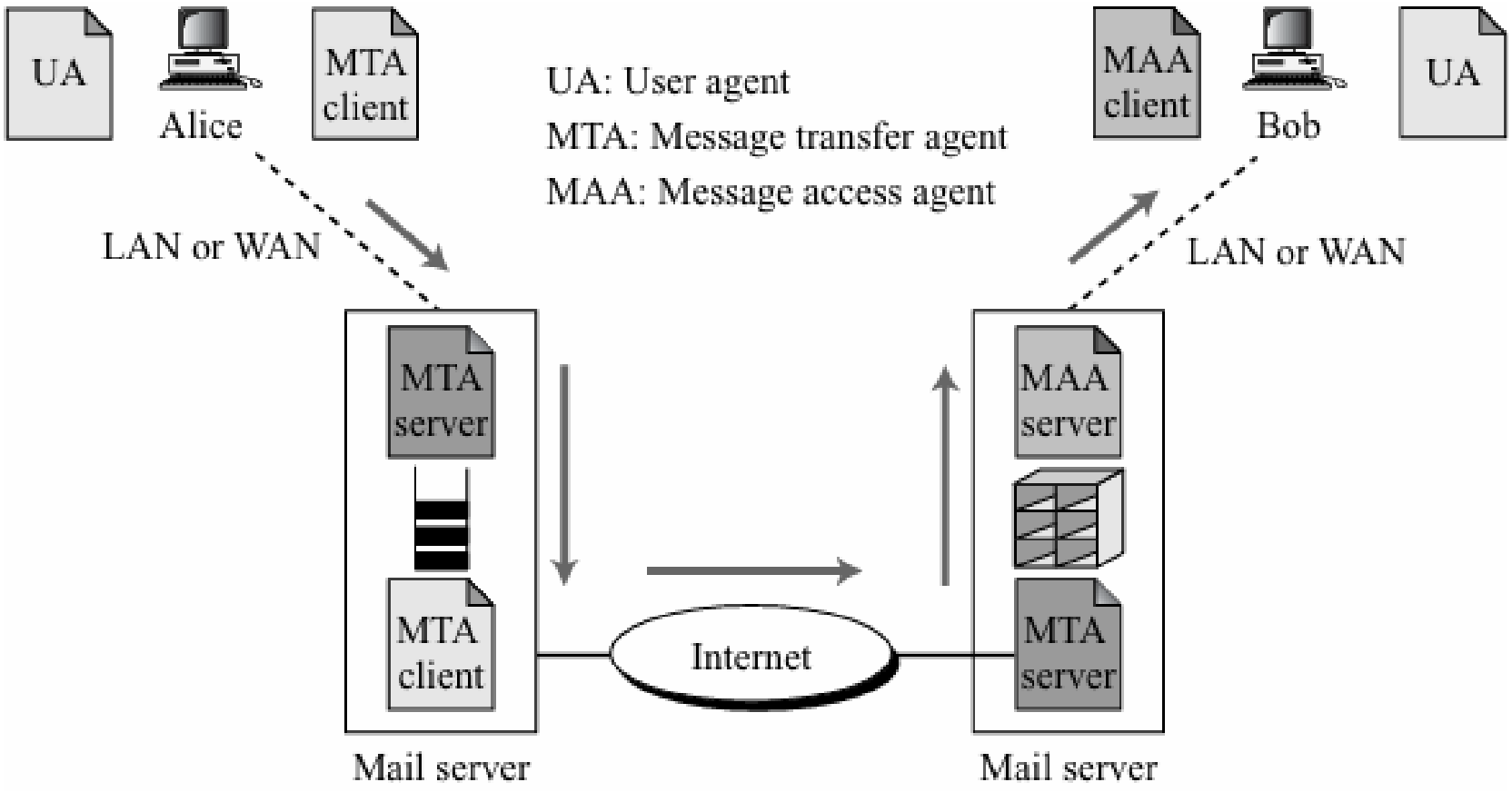
Email Security

Sachin Tripathi

IIT(ISM), Dhanbad

E-mail Architecture



Example

- Assume that Alice is working in an organization that runs an e-mail server; every employee is connected to the e-mail server through a LAN. Or alternatively, Alice could be connected to the e-mail server of an ISP through a WAN (telephone line or cable line).
- Bob is also in one of the above two situations.
- The administrator of the e-mail server at Alice's site has created a queuing system that sends e-mail to the Internet one by one. The administrator of the e-mail server at Bob's site has created a mailbox for every user connected to the server; the mailbox holds the received messages until they are retrieved by the recipient

- When Alice needs to send a message to Bob, she invokes a **user agent (UA)** program to prepare the message. She then uses another program, a **message transfer agent (MTA)**, to send the message to the mail server at her site.

Note that the MTA is a client/server program with the client installed at Alice's computer and the server installed at the mail server.

- The message received at the mail server at Alice's site is queued with all other messages; each goes to its corresponding destination. In Alice's case, her message goes to the mail server at Bob's site.

- A client/server MTA is responsible for the e-mail transfer between the two servers.
- When the message arrives at the destination mail server, it is stored in Bob's mailbox, a special file that holds the message until it is retrieved by Bob.
- When Bob needs to retrieve his messages, including the one sent by Alice, he invokes another program, which we call a **message access agent (MAA).**
- The MAA is also designed as a client/server program with the client installed at Bob's computer and the server installed at the mail server.

Important Points of Email Architecture

- The sending of an e-mail from Alice to Bob is a store-retrieve activity. Alice can send an e-mail today; Bob, being busy, may check his e-mail three days later. During this time, the e-mail is stored in Bob's mailbox until it is retrieved.
- The main communication between Alice and Bob is through two application programs: the MTA client at Alice's computer and the MAA client at Bob's computer.
- The MTA client program is a push program; the client pushes the message when Alice needs to send it. The MAA client program is a pull program; the client pulls the messages when Bob is ready to retrieve his e-mail.

- Alice and Bob cannot directly communicate using an MTA client at the sender site and an MTA server at the receiver site. This requires that the MTA server be running all the time, because Bob does not know when a message will arrive. This is not practical, because Bob probably turns off his computer when he does not need it.

Email Security

- Sending an e-mail is a one-time activity.
- In IPSec or SSL, we assume that the two parties create a session between themselves and exchange data in both directions.
- In e-mail, there is no session. Alice and Bob cannot create a session. Alice sends a message to Bob; sometime later, Bob reads the message and may or may not send a reply.
- Security of a unidirectional message is needed because what Alice sends to Bob is totally independent from what Bob sends to Alice.

Cryptographic Algorithms

- If e-mail is a one-time activity, how can the sender and receiver agree on a crypto graphic algorithm to use for e-mail security?
- If there is no session and no handshaking to negotiate the algorithms for encryption/decryption and hashing, how can the receiver know which algorithm the sender has chosen for each purpose?

- One solution is for the underlying protocol to select one algorithm for each crypto graphic operation and to force Alice to use only those algorithms. This solution is very restrictive and limits the capabilities of the two parties.

- A better solution is for the underlying protocol to define a set of algorithms for each operation that the user used in his/her system. Alice includes the name (or identifiers) of the algorithms she has used in the e-mail.

- For example, Alice can choose triple DES for encryption/decryption and MD5 for hashing. When Alice sends a message to Bob, she includes the corresponding identifiers for triple DES and MD5 in her message. Bob receives the message and extracts the identifiers first. He then knows which algorithm to use for decryption and which one for hashing.

Note :In e-mail security, the sender of the message needs to include the name or identifiers of the algorithms used in the message**.**

Cryptographic Secrets

- The same problem for the cryptographic algorithms applies to the cryptographic secrets (keys). If there is no negotiation, how can the two parties establish secrets between themselves? Alice and Bob could use asymmetric-key algorithms for authentication and encryption, which do not require the establishment of a symmetric key.

- Most e-mail security protocols today require that encryption/decryption be done using a symmetric-key algorithm and a one-time secret key sent with the message. Alice can create a secret key and send it with the message she sends to Bob. To protect the secret key from interception by Eve, the secret key is encrypted with Bob's public key. In other words, the secret key itself is encrypted.

In e-mail security, the encryption/decryption is done using a symmetric-key algorithm, but the secret key to decrypt the message is encrypted with the public key of the receiver and is sent with the message.

Certificates

- One more issue needs to be considered before we discuss any e-mail security protocol in particular. It is obvious that some public-key algorithms must be used for e-mail security. For example, we need to encrypt the secret key or sign the message.
- To encrypt the secret key, Alice needs Bob's public key; to verify a signed message, Bob needs Alice's public key. So, for sending a small authenticated and confidential message, two public keys are needed. How can Alice be assured of Bob's public key, and how can Bob be assured of Alice's public key?

Each e-mail security protocol has a different method of certifying keys.

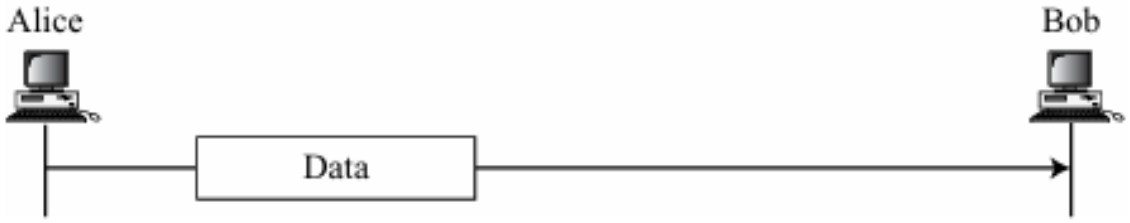
Pretty Good Privacy (PGP)

- PGP was invented by Phil Zimmermann to provide e-mail with privacy, integrity, and authentication.
- PGP can be used to create a secure e-mail message or to store a file securely for future retrieval.

Let us first discuss the general idea of PGP in Different Scenarios

Plaintext

The simplest scenario is to send the e-mail message (or store the file) in plaintext . There is no message integrity or confidentiality in this scenario. Alice, the sender, composes a message and sends it to Bob, the receiver. The message is stored in Bob's mailbox until it is retrieved by him.

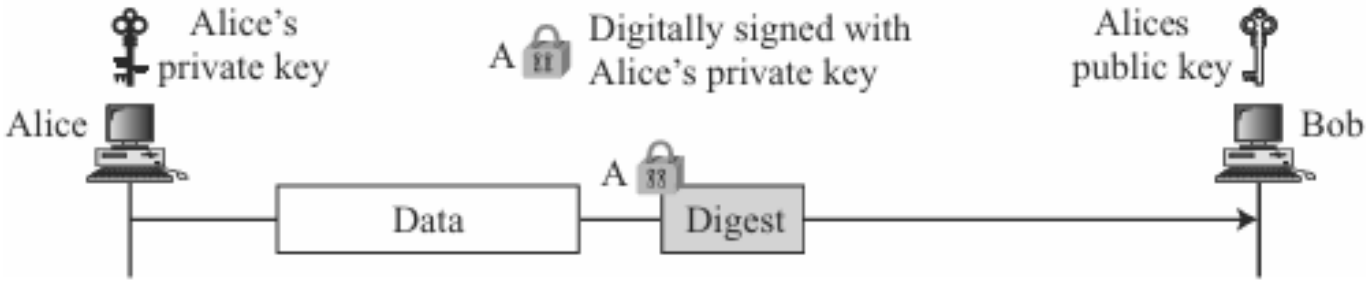


Pretty Good Privacy (PGP)

Message Integrity

- Probably the next improvement is to let Alice sign the message. Alice creates a digest of the message and signs it with her private key. When Bob receives the message, he verifies the message by using Alice's public key. Two keys are needed for this scenario. Alice needs to know her private key; Bob needs to know Alice's public key.

An authenticated message

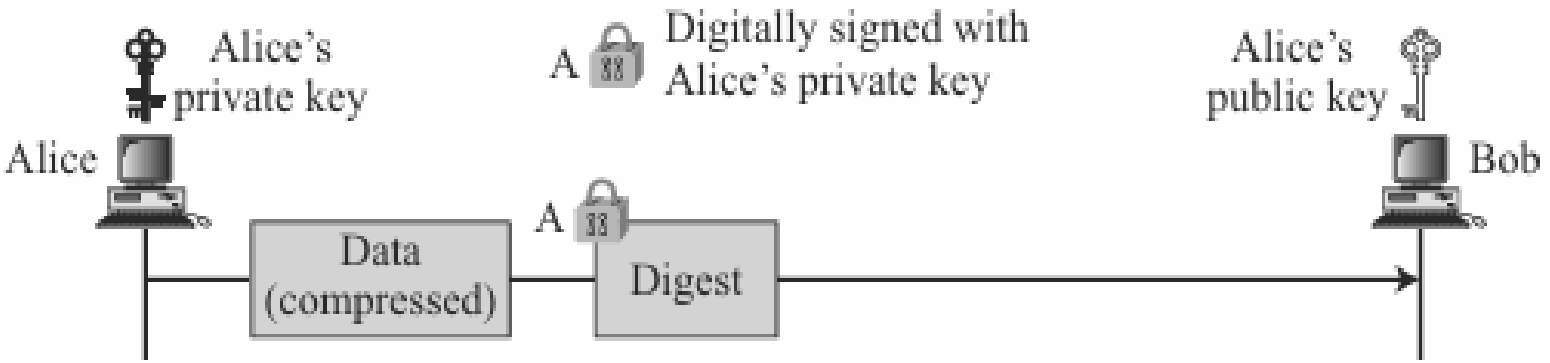


Pretty Good Privacy (PGP)

Compression

- A further improvement is to compress the message to make the packet more compact. This improvement has no security benefit, but it eases the traffic. Figure below shows the new scenario. .

An compressed message

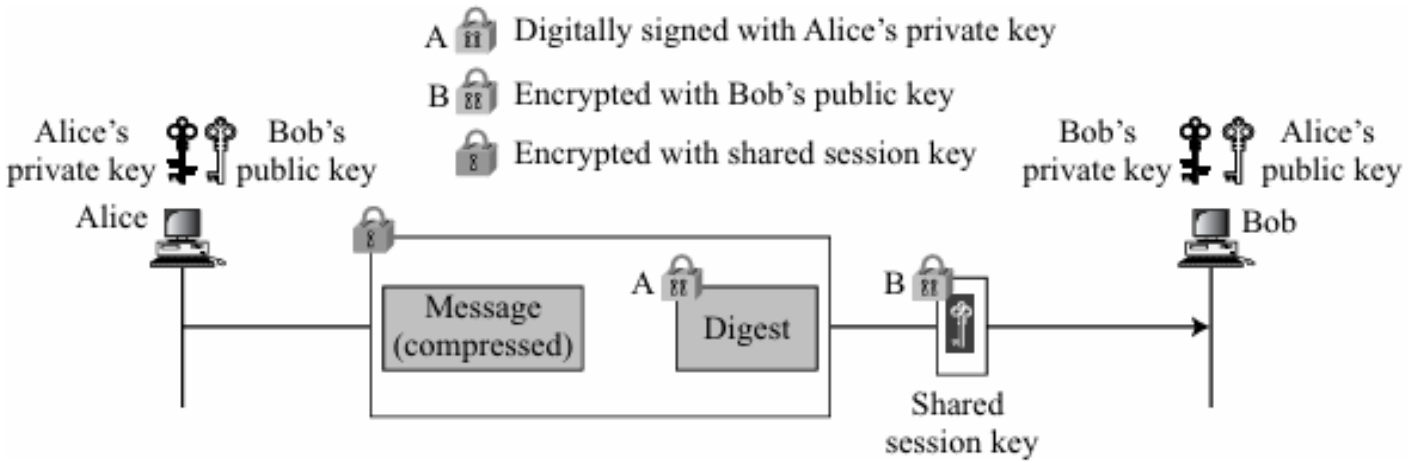


Pretty Good Privacy (PGP)

Confidentiality with One-Time Session Key

- confidentiality in an e-mail system can be achieved using conventional encryption with a one-time session key. Alice can create a session key, use the session key to encrypt the message and the digest, and send the key itself with the message. However, to protect the session key, Alice encrypts it with Bob's public key. Figure below shows the situation.

An confidential message



Pretty Good Privacy (PGP)

Code Conversion

- Another service provided by PGP is code conversion. Most e-mail systems allow the message to consist of only ASCII characters. To translate other characters not in the ASCII set, PGP uses Radix-64 conversion. Each character to be sent (after encryption) is converted to Radix-64 code.

Segmentation

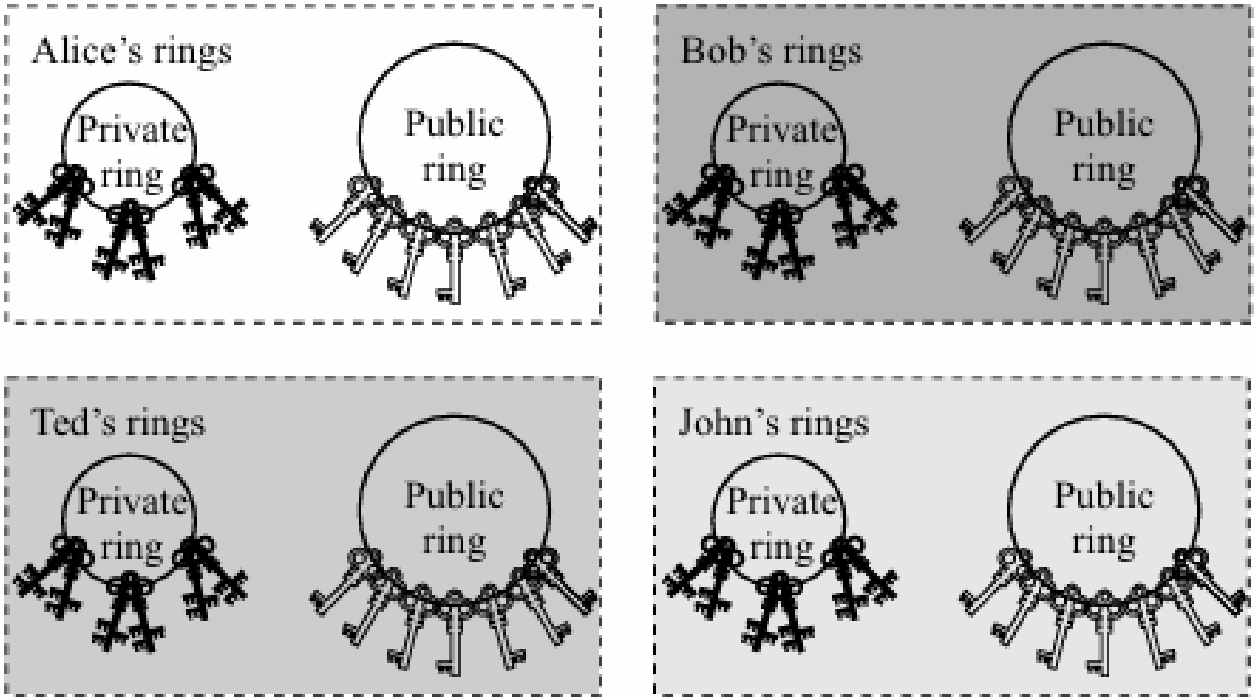
- PGP allows segmentation of the message after it has been converted to Radix-64 to make each transmitted unit the uniform size as allowed by the underlying e-mail protocol.

Key Rings

- In all previous scenarios, we assumed that Alice needs to send a message only to Bob. That is not always the case. Alice may need to send messages to many people; she needs key rings. In this case, Alice needs a ring of public keys, with a key belonging to each person with whom Alice needs to correspond (send or receive messages). In addition, the PGP designers specified a ring of private/public keys. One reason is that Alice may wish to change her pair of keys from time to time. Another reason is that Alice may need to correspond with different groups of people (friends, colleagues, and so on). Alice may wish to use a different key pair for each group. Therefore, each user needs to have two sets of rings: a ring of private/public keys and a ring of public keys of other people.

Key Rings

Figure below a community of four people, each having a ring of pairs of private/public keys and, at the same time, a ring of public keys belonging to other people in the community.



Key Rings

- Alice, for example, has several pairs of private/public keys belonging to her and public keys belonging to other people. Note that everyone can have more than one public key. Two cases may arise.

1. Alice needs to send a message to another person in the community.

a. She uses her private key to sign the digest.

b. She uses the receiver's public key to encrypt a newly created session key.

c. She encrypts the message and signed digest with the session key created.

2. Alice receives a message from another person in the community

   a. She uses her private key to decrypt the session key.

   b. She uses the session key to decrypt the message and digest.

   c. She uses her public key to verify the digest

PGP Algorithms

1.Public-Key Algorithms:-The public-key algorithms that are used for signing the digests or encrypting the messages are listed below

ID	*Description*
1	RSA (encryption or signing)
2	RSA (for encryption only)
3	RSA (for signing only)
16	ElGamal (encryption only)
17	DSS
18	Reserved for elliptic curve
19	Reserved for ECDSA
20	ElGamal (for encryption or signing)
21	Reserved for Diffie-Hellman
100–110	Private algorithms

PGP Algorithms

2.Symmetric-Key Algorithms:-The symmetric-key algorithms that are used for conventional encrypting are shown in Table below

ID	*Description*
0	No Encryption
1	IDEA
2	Triple DES
3	CAST-128
4	Blowfish
5	SAFER-SK128
6	Reserved for DES/SK
7	Reserved for AES-128
8	Reserved for AES-192
9	Reserved for AES-256
100–110	Private algorithms

PGP Algorithms

3. Hash Algorithms:- The hash algorithms that are used for creating hashes in PGP are shown in Table below

ID	*Description*
1	MD5
2	SHA-1
3	RIPE-MD/160
4	Reserved for double-width SHA
5	MD2
6	TIGER/192
7	Reserved for HAVAL
100–110	Private algorithms

PGP Algorithms

4.Compression Algorithms:- The compression algorithms that are used for compressing text are shown in Table below

ID	*Description*
0	Uncompressed
1	ZIP
2	ZLIP
100–110	Private methods

PGP Certificates

- PGP, uses certificates to authenticate public keys. However, the process is totally different

X.509 Certificates

Protocols that use X.509 certificates depend on the hierarchical structure of the trust. There is a predefined chain of trust from the root to any certificate. Every user fully trusts the authority of the CA at the root level (prerequisite). The root issues certificates for the CAs at the second level, a second level CA issues a certificate for the third level, and so on. Every party that needs to be trusted presents a certificate from some CA in the tree. If Alice does not trust the certificate issuer for Bob, she can appeal to a higher level authority up to the root (which must be trusted for the system to work). In other words, there is one single path from a fully trusted CA to a certificate. .

Note:-In X.509, there is a single path from the fully trusted authority to any certificate.

PGP Certificates

❑ PGP Certificates

- In PGP, there is no need for CAs; anyone in the ring can sign a certificate for anyone else in the ring.
- Bob can sign a certificate for Ted, John, Anne, and so on.
- There is no hierarchy of trust in PGP; there is no tree.
- The lack of hierarchical structure may result in the fact that Ted may have one certificate from Bob and another certificate from Liz. If Alice wants to follow the line of certificates for Ted, there are two paths: one starts from Bob and one starts from Liz.
- An interesting point is that Alice may fully trust Bob, but only partially trust Liz. There can be multiple paths in the line of trust from a fully or partially trusted authority to a certificate. In PGP, the issuer of a certificate is usually called an introducer.

Note:-In PGP, there can be multiple paths from fully or partially trusted authorities to any subject.

PGP Certificates

❑ Trusts and Legitimacy

The entire operation of PGP is based on introducer trust, the certificate trust, and the legitimacy of the public keys.

❑ Introducer Trust Levels

With the lack of a central authority, it is obvious that the ring cannot be very large if every user in the PGP ring of users has to fully trust everyone else. (Even in real life we cannot fully trust everyone that we know.) To solve this problem, PGP allows different levels of trust. The number of levels is mostly implementation dependent, but for simplicity, let us assign three levels of trust to any introducer: none, partial, and full. The introducer trust level specifies the trust levels issued by the introducer for other people in the ring. For example, Alice may fully trust Bob, partially trust Anne, and not trust John at all. There is no mechanism in PGP to deter mine how to make a decision about the trustworthiness of the introducer; it is up to the user to make this decision.

PGP Certificates

Certificate Trust Levels

- When Alice receives a certificate from an introducer, she stores the certificate under the name of the subject (certified entity).
- She assigns a level of trust to this certificate. The certificate trust level is normally the same as the introducer trust level that issued the certificate. Assume that Alice fully trusts Bob, partially trusts Anne and Janette, and has no trust in John. The following scenarios can happen.

1. Bob issues two certificates, one for Linda (with public key K1) and one for Lesley (with public key K2). Alice stores the public key and certificate for Linda under Linda's name and assigns a full level of trust to this certificate. Alice also stores the certificate and public key for Lesley under Lesley's name and assigns a full level of trust to this certificate.
2. Anne issues a certificate for John (with public key K3). Alice stores this certificate and public key under John's name, but assigns a partial level for this certificate.

PGP Certificates

- 3. Janette issues two certificates, one for John (with public key K3) and one for Lee (with public key K4). Alice stores John's certificate under his name and Lee's certificate under his name, each with a partial level of trust. Note that John now has two certificates, one from Anne and one from Janette, each with a partial level of trust.

- 4.John issues a certificate for Liz. Alice can discard or keep this certificate with a sig nature trust of none.

PGP Certificates

❑ Key Legitimacy

- The purpose of using introducer and certificate trusts is to deter mine the legitimacy of a public key. Alice needs to know how legitimate the public keys of Bob, John, Liz, Anne, and so on are.
- PGP defines a very clear procedure for deter mining key legitimacy. The level of the key legitimacy for a user is the weighted trust levels of that user. For example, suppose we assign the following weights to certificate trust levels:

1. A weight of 0 to a nontrusted certificate
2. A weight of 1/2 to a certificate with partial trust
3. A weight of 1 to a certificate with full trust

Then to fully trust an entity, Alice needs one fully trusted certificate or two partially trusted certificates for that entity..

- For example, Alice can use John’s public key in the previous scenario because both Anne and Janette have issued a certificate for John, each with a certificate trust level of 1/2.
- Note that the legitimacy of a public key belonging to an entity does not have anything to do with the trust level of that person. Although Bob can use John’s public key to send a message to him, Alice cannot accept any certificate issued by John because, for Alice, John has a trust level of none

PGP Certificates

❑ Starting the Ring

You might have realized a problem with the above discussion. What if nobody sends a certificate for a fully or partially trusted entity? For example, how can the legitimacy of Bob's public key be determined if no one has sent a certificate for Bob? In PGP, the key legitimacy of a trusted or partially trusted entity can be also determined by other methods.

1. Alice can physically obtain Bob's public key. For example, Alice and Bob can meet personally and exchange a public key written on a piece of paper or to a disk.
2. If Bob's voice is recognizable to Alice, Alice can call him and obtain his public key on the phone.

PGP Certificates

- 3. A better solution proposed by PGP is for Bob to send his public key to Alice by e-mail. Both Alice and Bob make a 16-byte MD5 (or 20-byte SHA-1) digest from the key. The digest is normally displayed as eight groups of 4 digits (or ten groups of 4 digits) in hexadecimal and is called a fingerprint. Alice can then call Bob and verify the fingerprint on the phone. If the key is altered or changed during the e-mail transmission, the two fingerprints do not match. To make it even more convenient, PGP has created a list of words, each representing a 4-digit combination. When Alice calls Bob, Bob can pronounce the eight words (or ten words) for Alice. The words are carefully chosen by PGP to avoid those similar in pronunciation; for example, if sword is in the list, word is not.

- 4. In PGP, nothing prevents Alice from getting Bob's public key from a CA in a separate procedure. She can then insert the public key in the public key ring.

Key Ring Tables

- Each user, such as Alice, keeps track of two key rings: one private-key ring and one public key ring. PGP defines a structure for each of these key rings in the form of a table.

- Format of Private Key Ring Table

User ID	Key ID	Public key	Encrypted private key	Timestamp
⋮	⋮	⋮	⋮	⋮

- **User ID.** The user ID is usually the e-mail address of the user. However, the user may designate a unique e-mail address or alias for each key pair. The table lists the user ID associated with each pair.

Key Ring Tables

- **Key ID**.
  - This column uniquely defines a public key among the user's public keys. In PGP, the key ID for each pair is the first (least significant) 64 bits of the public key. In other words, the key ID is calculated as (key mod 2 64). The key ID is needed for the operation of PGP because Bob may have several public keys belonging to Alice in his public key ring. When he receives a message from Alice, Bob must know which key ID to use to verify the message.
  - The key ID, which is sent with the message, enables Bob to use a specific public key for Alice from his public ring.
  - Note the size of the public key may be very long. Sending just 8 bytes reduces the size of the message.
- **Public Key**. This column just lists the public key belonging to a particular private key/public key pair

Key Ring Tables

- **Encrypted Private Key**. This column shows the encrypted value of the private key in the private key/public key pair. Although Alice is the only person accessing her private ring, PGP saves only the encrypted version of the private key. We will see later how the private key is encrypted and decrypted.
- **Timestamp.** This column holds the date and time of the key pair creation. It helps the user decide when to purge old pairs and when to create new ones.

Key Ring Tables

Format of a Public Key Ring Table

User ID	Key ID	Public key	Producer trust	Certificate(s)	Certificate trust(s)	Key Legitimacy	Timestamp
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- **User ID**. As in the private key ring table, the user ID is usually the e-mail address of the entity.
- **Key ID**. As in the private key ring table, the key ID is the first (least significant) 64 bits of the public key.
- **Public Key.** This is the public key of the entity.
- **Producer Trust**. This column defines the producer level of trust. In most implementations, it can only be of one of three values: none, partial, or full.

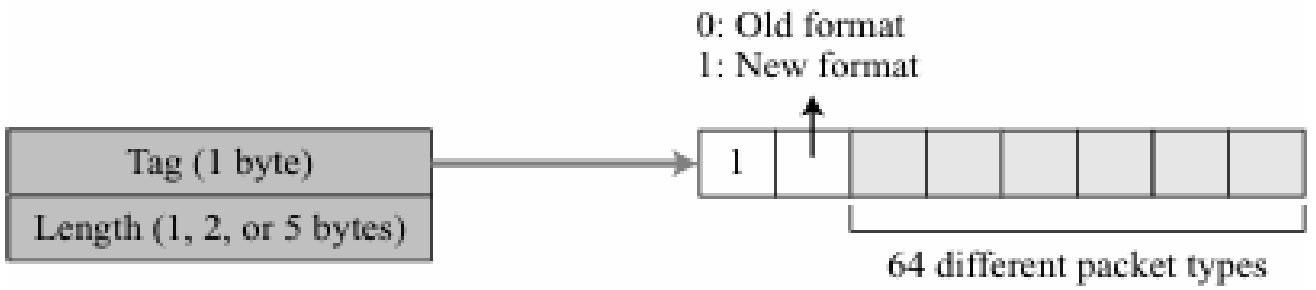
Key Ring Tables

- **Certificate(s).** This column holds the certificate or certificates signed by other entities for this entity. A user ID may have more than one certificate.
- **Certificate Trust(s).** This column represents the certificate trust or trusts.
- **Key Legitimacy.** This value is calculated by PGP based on the value of the certificate trust and the predefined weight for each certificate trust.
- **Timestamp.** This column holds the date and time of the column creation.

PGP Packets

A message in PGP consists of one or more packets. During the evolution of PGP, the format and the number of packet types have changed. Like other protocols we have seen so far, PGP has a generic header that applies to every packet. The generic header, in the most recent version, has only two fields

Format of packet header



PGP Packets

- **Tag.** The recent format for this field defines a tag as an 8-bit flag; the first bit (most significant) is always 1. The second bit is 1 if we are using the latest version. The remaining six bits can define up to 64 different packet types

Value	*Packet type*
1	Session key packet encrypted using a public key
2	Signature packet
5	Private-key packet
6	Public-key packet
8	Compressed data packet
9	Data packet encrypted with a secret key
11	Literal data packet
13	User ID packet

PGP Packets

❑ **Length**. The length field defines the length of the entire packet in bytes. The size of this field is variable; it can be 1, 2, or 5 bytes. The receiver can determine the number of bytes of the length field by looking at the value of the byte immediately following the tag field.

a. If the value of the byte after the tag field is less than 192, the length field is only one byte. The length of the body (packet minus header) is calculated as: **body length = first byte**

b. If the value of the byte after the tag field is between 192 and 223 (inclusive), the length field is two bytes. The length of the body can be calculated as: body length = **(first byte - 192) << 8 + second byte + 192**

PGP Packets

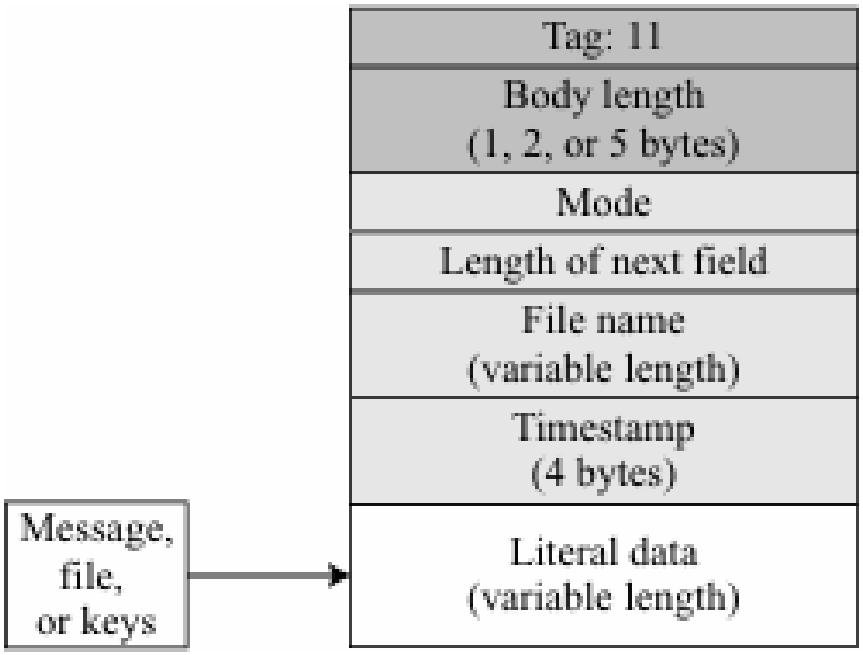
c. If the value of the byte after the tag field is between 224 and 254 (inclusive), the length field is one byte. This type of length field defines only the length of part of the body (partial body length). The partial body length can be calculated as:

partial body length = 1 << (first byte & 0x1F)

d. If the value of the byte after the tag field is 255, the length field con sists of five bytes. The length of the body is calculated as:
Body length=second byte<< 24 | third byte << 16 | fourth byte << 8

PGP Packets –Literal Data Packet

- **Literal Data Packet:-** The literal data packet is the packet that carries or holds the actual data that is being transmitted or stored. This packet is the most elementary type of message; that is, it cannot carry any other packet.



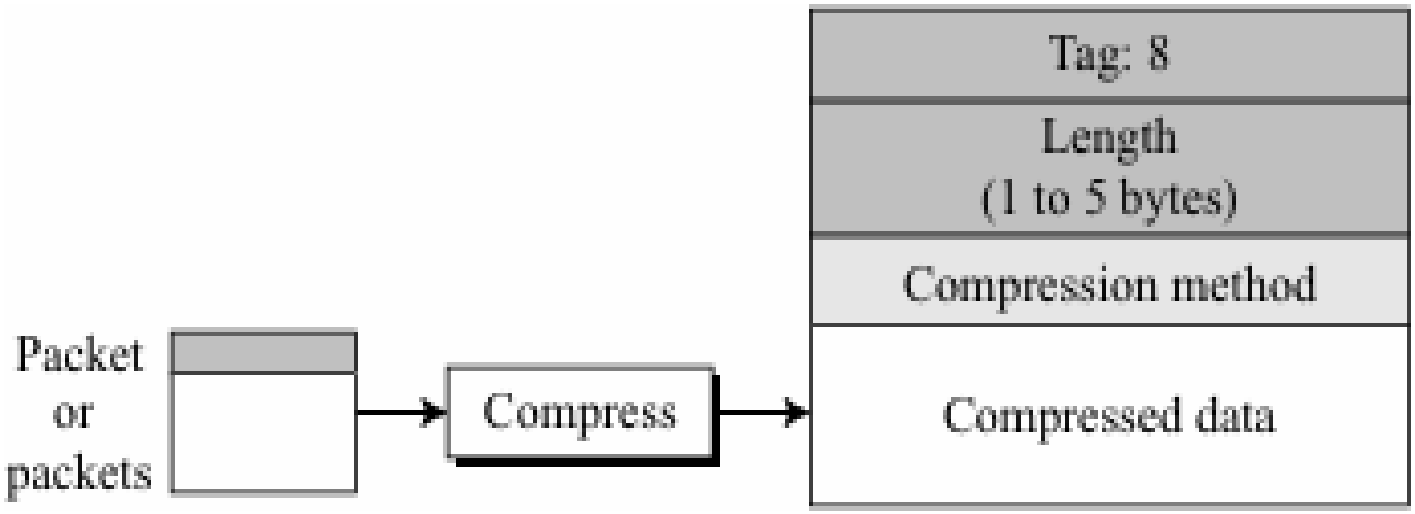
PGP Packets –Literal Data Packet

- **Mode.** This one-byte field defines how data is written to the packet. The value of this field can be “b” for binary, “t” for text, or any other locally defined value.
- **Length of next field.** This one-byte field defines the length of the next field (file name field).
- **File name**. This variable-length field defines the name of the file or message as an ASCII string.
- **Timestamp.** This four-byte field defines the time of creation or last modification of the message. The value can be 0, which means that the user chooses not to specify a time.
- **Literal data.** This variable-length field carries the actual data (file or message) in text or binary (depending on the value of the mode field).

PGP Packets –Compressed Data Packet

Compressed Data Packet This packet carries compressed data packets.

format of a compressed data packet.



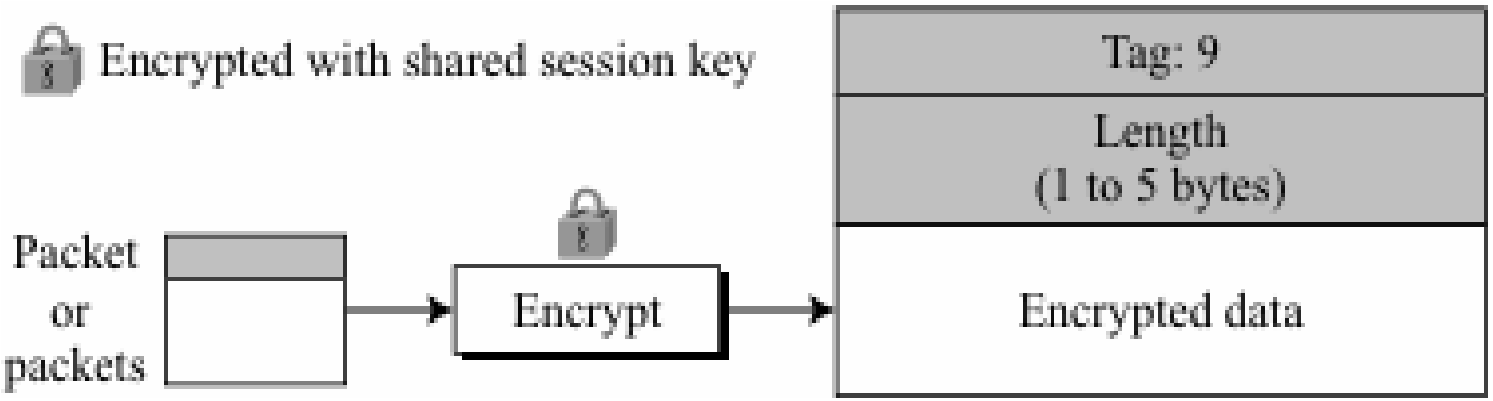
PGP Packets –Compressed Data Packet

❑ **Compression method**. This one-byte field defines the compression method used to compress the data (next field). The values defined for this field so far are 1 (ZIP) and 2 (ZLIP). Also, an implementation can use other experimental compression methods. .

❑ **Compressed data.** This variable-length field carries the data after compression. Note that the data in this field can be one packet or the concatenation of two or more packets. The common situation is a single literal data packet or a combination of a signature packet followed by a literal data packet.

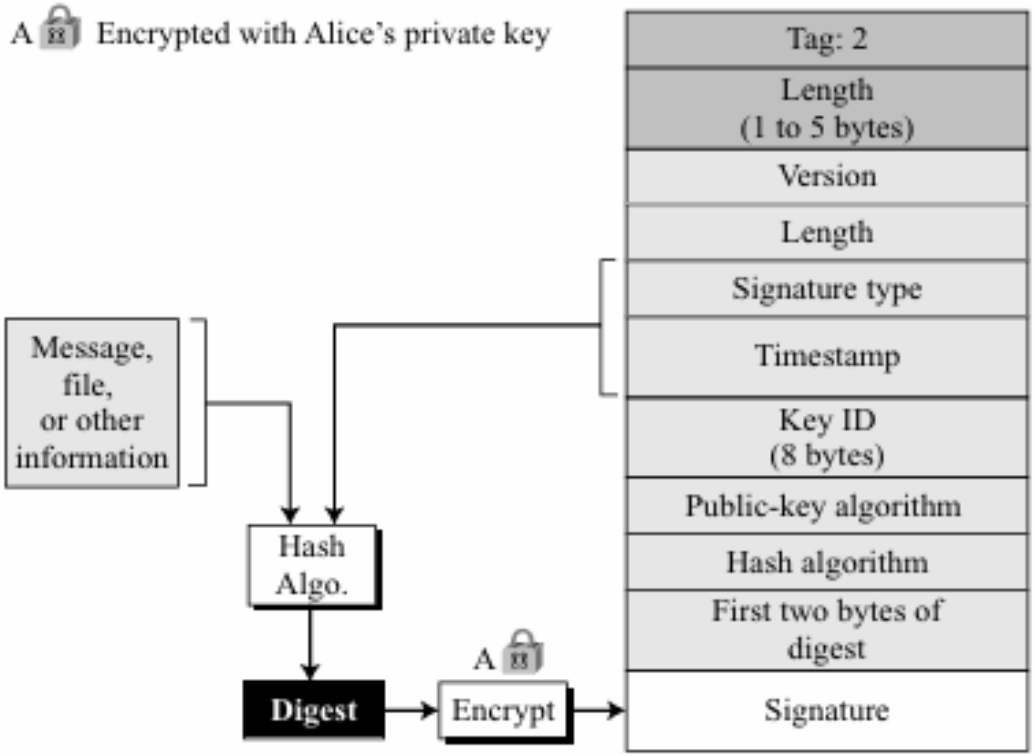
PGP Packets –Data Packet Encrypted with Secret Key

Data Packet Encrypted with Secret Key This packet carries data from one packet or a combination of packets that have been encrypted using a conventional symmetric key algorithm. Note that a packet carrying the one-time session key must be sent before this packet.



PGP Packets –Signature Packet

Signature Packet A signature packet, protects the integrity of the data**.**



PGP Packets –Signature Packet

- **Version**. This one-byte field defines the PGP version that is being used.
- **Length**. This field was originally designed to show the length of the next two fields, but because the size of these fields is now fixed, the value of this field is 5.
- **Signature type.** This one-byte field defines the purpose of the signature, the document it signs.(some signature values given below)

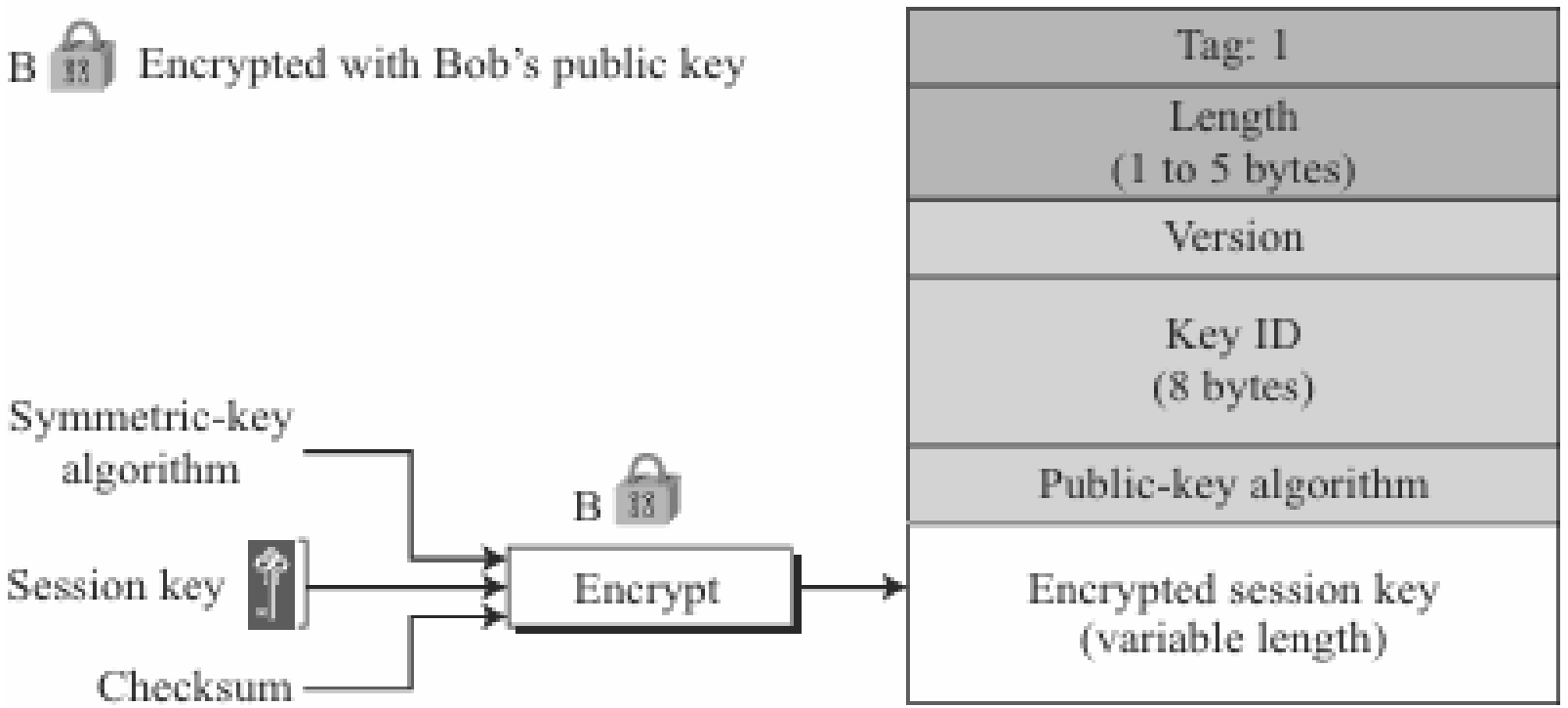
Value	*Signature*
0x00	Signature of a binary document (message or file).
0x01	Signature of a text document (message or file).
0x10	Generic certificate of a user ID and public-key packet. The signer does not make any particular assertion about the owner of the key.
0x11	Personal certificate of a user ID and public-key packet. No verification is done on the owner of the key.
0x12	Casual certificate of a User ID and public-key packet. Some casual verification done on the owner of the key.
0x13	Positive certificate of a user ID and public-key packet. Substantial verification done.
0x30	Certificate revocation signature. This removes an earlier certificate (0x10 through 0x13).

PGP Packets –Signature Packet

- **Timestamp**. This four-byte field defines the time the signature was calculated.
- **Key ID.** This eight-byte field defines the public-key ID of the signer. It indicates to the verifier which signer public key should be used to decrypt the digest.
- **Public-key algorithm**. This one-byte field gives the code for the public-key algorithm used to encrypt the digest. The verifier uses the same algorithm to decrypt the digest.
- **Hash algorithm**. This one-byte field gives the code for the hash algorithm used to create the digest.
- **First two bytes of message digest**. These two bytes are used as a kind of check sum. They ensure that the receiver is using the right key ID to decrypt the digest.
- **Signature.** This variable-length field is the signature. It is the encrypted digest signed by the sender.

PGP Packets-Session-Key Packet Encrypted with Public Key

Session-Key Packet Encrypted with Public Key This packet is used to send the session key encrypted with the receiver public key.



PGP Packets-Session-Key Packet Encrypted with Public Key

- **Version.** This one-byte field defines the PGP version being used.
- **Key ID**. This eight-byte field defines the public-key ID of the sender. It indicates to the receiver which sender public key should be used to decrypt the session key.
- **Public-key algorithm.** This one-byte field gives the code for the public-key algorithm used to encrypt the session key. The receiver uses the same algorithm to decrypt the session key.

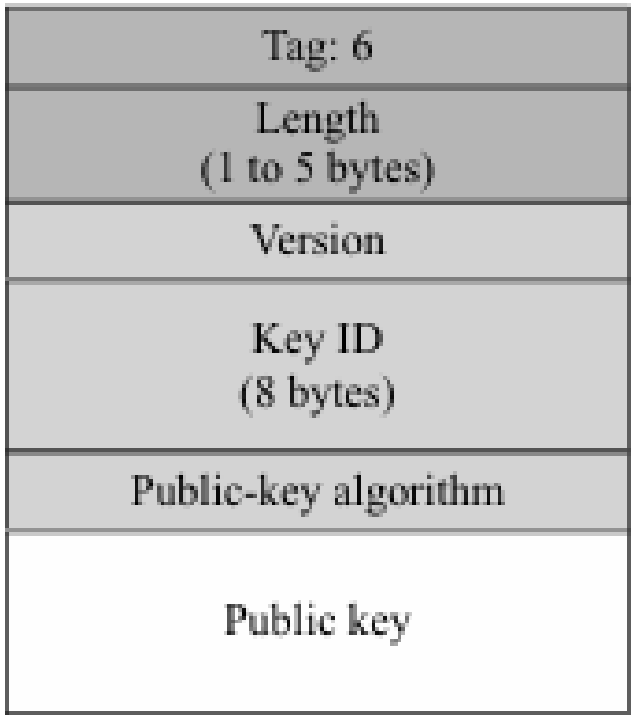
PGP Packets-Session-Key Packet Encrypted with Public Key

❑ **Encrypted session.** This variable-length field is the encrypted value of the session key created by the sender and sent to the receiver. The encryption is done on the following:

a. One-octet symmetric encryption algorithm
b. The session key
c. A two-octet checksum equal to the sum of the preceding session-key octets

PGP Packets-Public-Key Packet

Public-Key Packet:- This packet contains the public key of the sender.

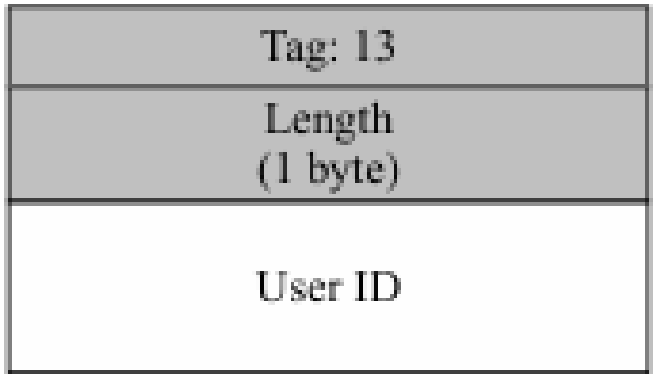


PGP Packets-Public Key Packet

- **Version**. This one-byte field defines the PGP version of the PGP being used.
- **Timestamp.** This four-byte field defines the time the key was created.
- **Validity.** This two-byte field shows the number of days the key is valid. If the value is 0, it means the key does not expire.
- **Public-key algorithm**. This one-byte field gives the code for the public-key algorithm.
- **Public key.** This variable-length field holds the public key itself. Its contents depend on the public-key algorithm used.

PGP Packets-User ID Packet

User ID Packet This packet identifies a user and can normally associate the user ID contents with a public key of the sender. Note that the length field of the general header is only one byte



❑ **User ID.** This variable-length string defines the user ID of the sender. It is normally the name of the user followed by an e-mail address.

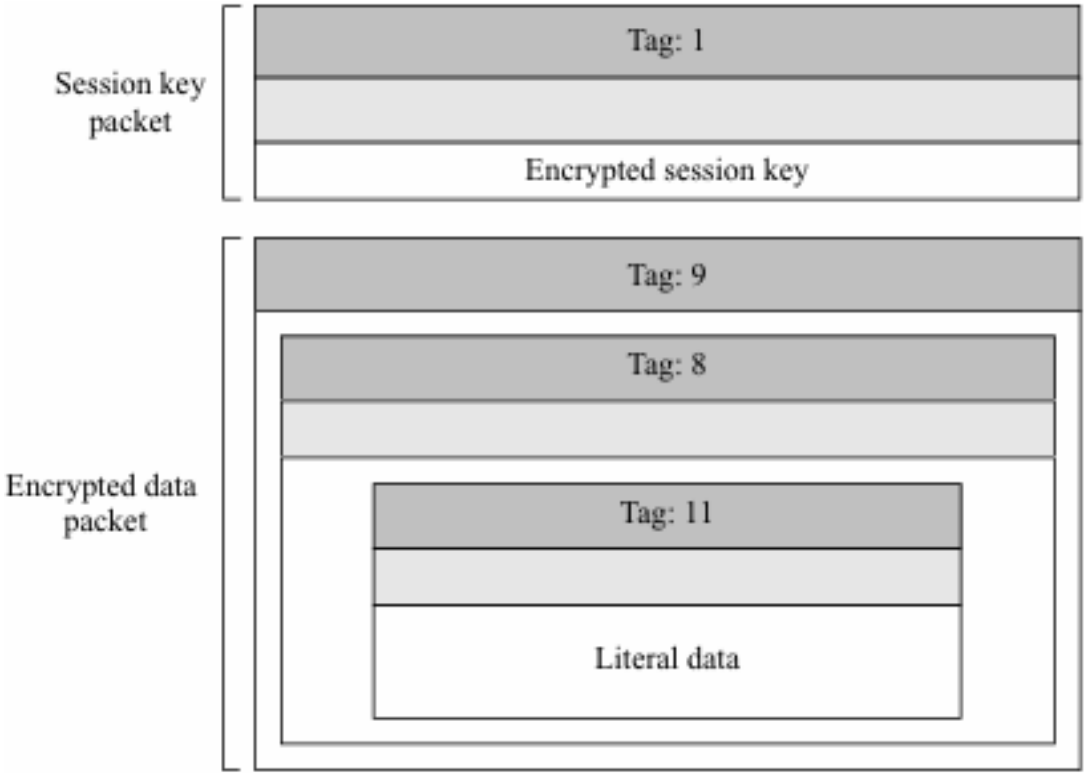
PGP Messages

- A message in PGP is a combination of sequenced and/or nested packets. Even though not all combinations of packets can make a message.

Encrypted Message

- An encrypted message can be a sequence of two packets, a session-key packet and a symmetrically encrypted packet. The latter is normally a nested packet.

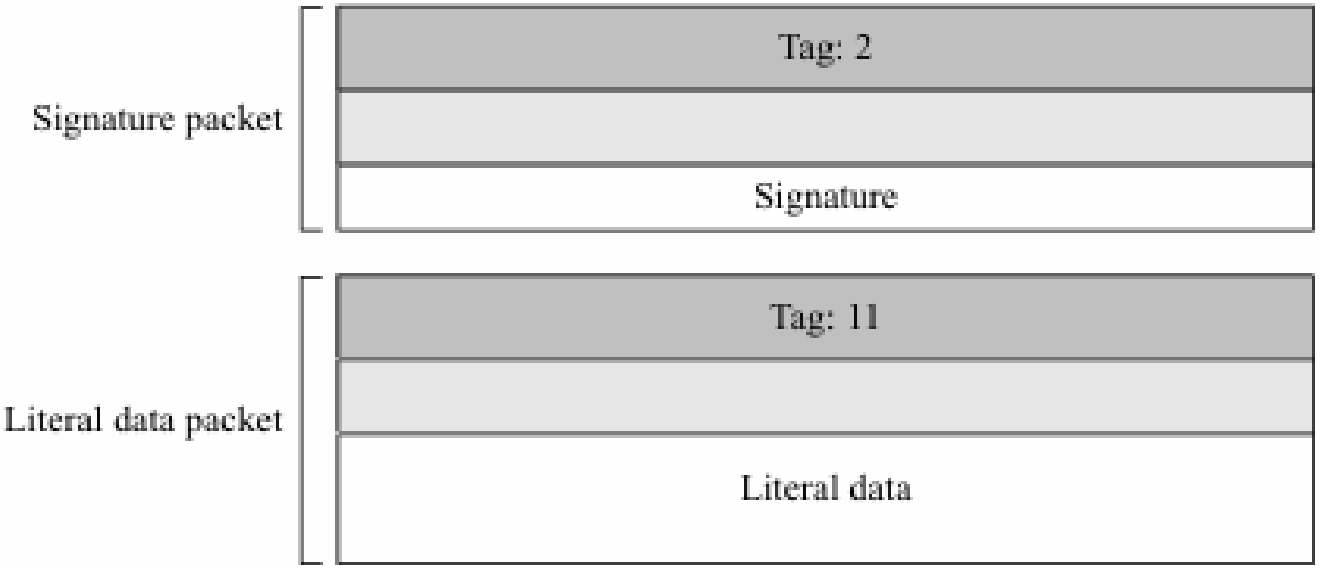
Encrypted Message



Note:-The session-key packet is just a single packet. The encrypted data packet, however, is made of a compressed packet. The compressed packet is made of a literal data packet. The last one holds the literal data.

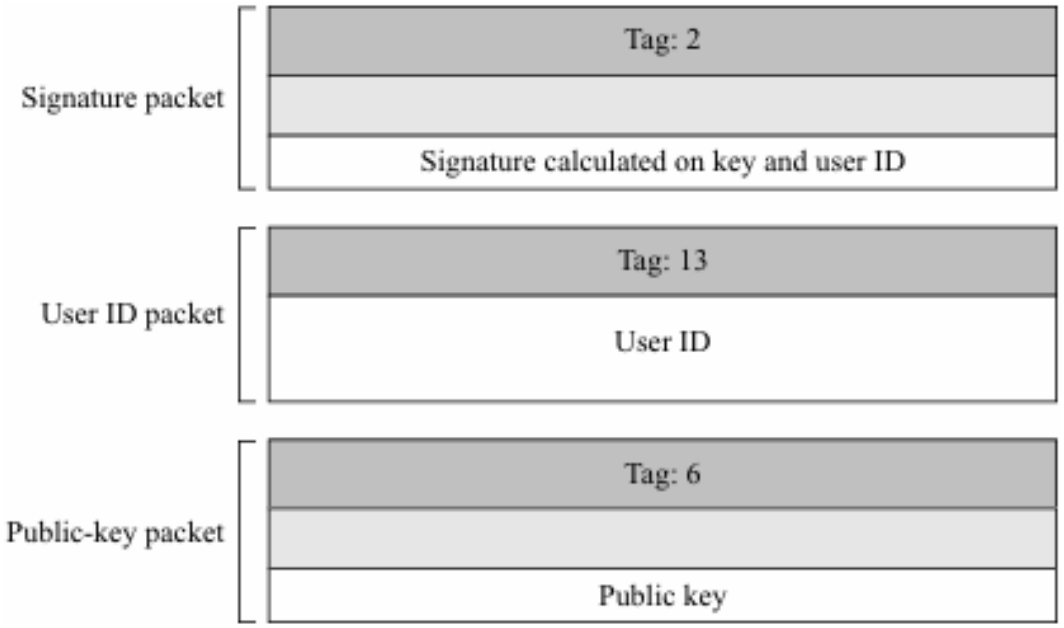
PGP Messages -Signed Message

- A signed message can be the combination of a signature packet and a literal packet, as shown in Figure



PGP Messages -Certificate Message

- Although a certificate can take many forms, one simple example is the combination of a user ID packet and a public-key packet The signature is then calculated on the concatenation of the key and user ID.



Applications of PGP

- PGP has been extensively used for personal e-mails. It will probably continue to be.

S/MIME

- S/MIME adds some new content types to include security services to the MIME.

All of these new types include the parameter "**application/pkcs7-mime**," in which "**pkcs**" defines "Public Key Cryptography Specification**.**"

S/MIME

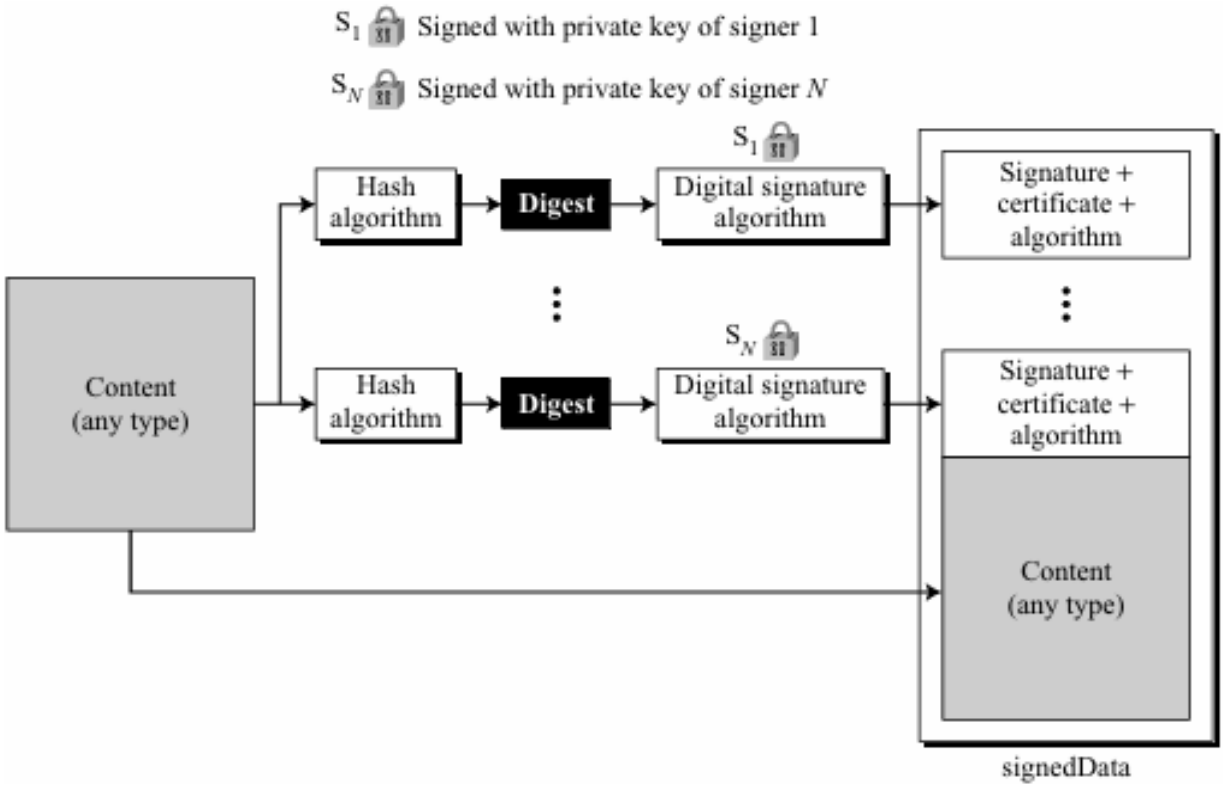
❑ Cryptographic Message Syntax (CMS)

To define how security services, such as confidentiality or integrity, can be added to MIME content types, S/MIME has defined Cryptographic Message Syntax (CMS). The syntax in each case defines the exact encoding scheme for each content type. The following describe the type of message and different subtypes that are created from these messages.

Data Content Type This is an arbitrary string. The object created is called Data.

S/MIME

- **Signed-Data Content Type** This type provides only integrity of data. It contains any type and zero or more signature values. The encoded result is an object called signedData.



S/MIME

The following are the steps in the process:

1. For each signer, a message digest is created from the content using the specific hash algorithm chosen by that signer.
2. Each message digest is signed with the private key of the signer.
3. The content, signature values, certificates, and algorithms are then collected to create the signedData object

S/MIME

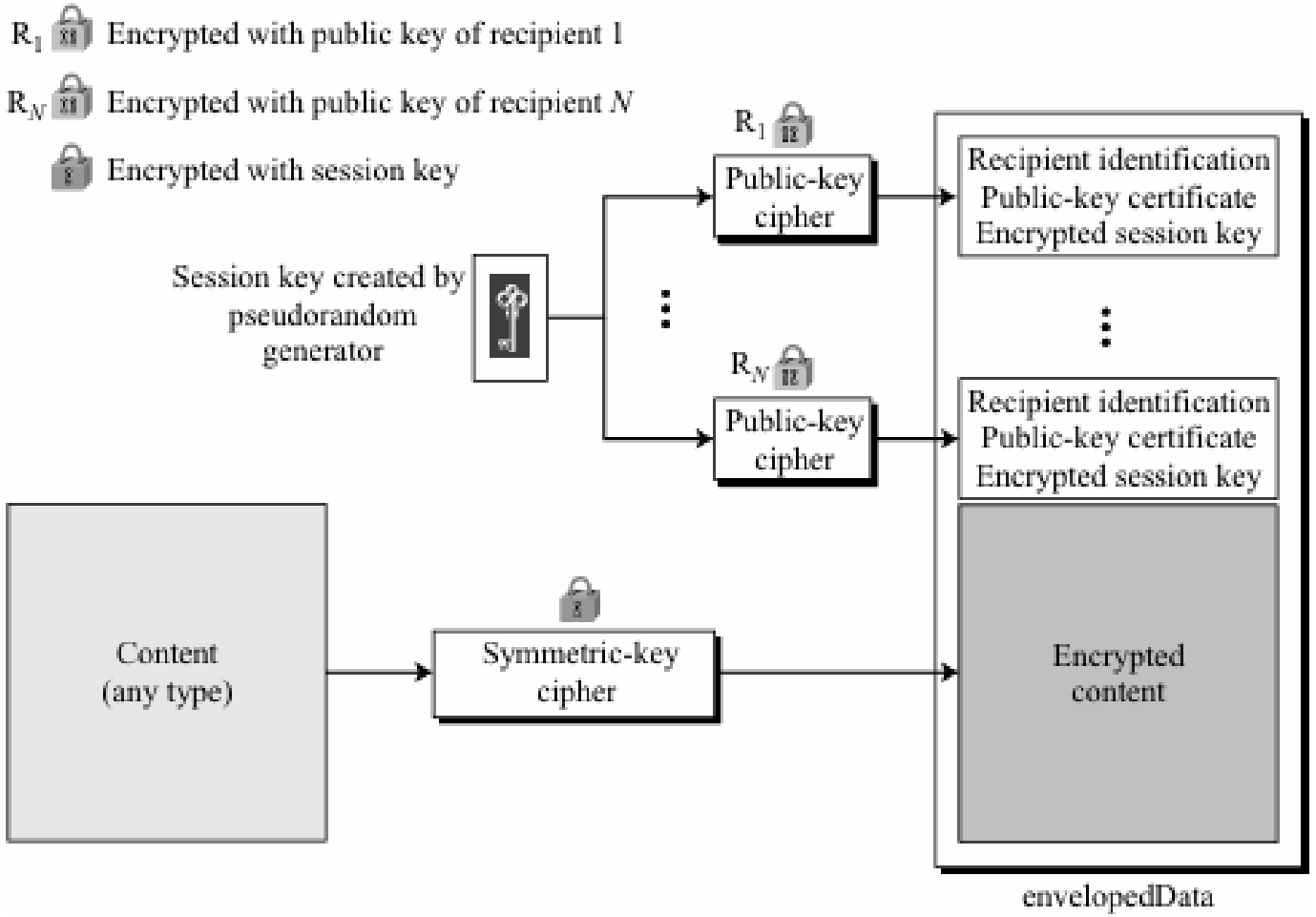
❑ Enveloped-Data Content Type

This type is used to provide privacy for the message. It contains any type and zero or more encrypted keys and certificates. The encoded result is an object called envelopedData. Process of creating an object of this type

1. A pseudorandom session key is created for the symmetric-key algorithms to be used.
2. For each recipient, a copy of the session key is encrypted with the public key of each recipient.
3. The content is encrypted using the defined algorithm and created session key.
4. The encrypted contents, encrypted session keys, algorithm used, and certificates are encoded using Radix-64.

S/MIME

- **Enveloped-data content type**

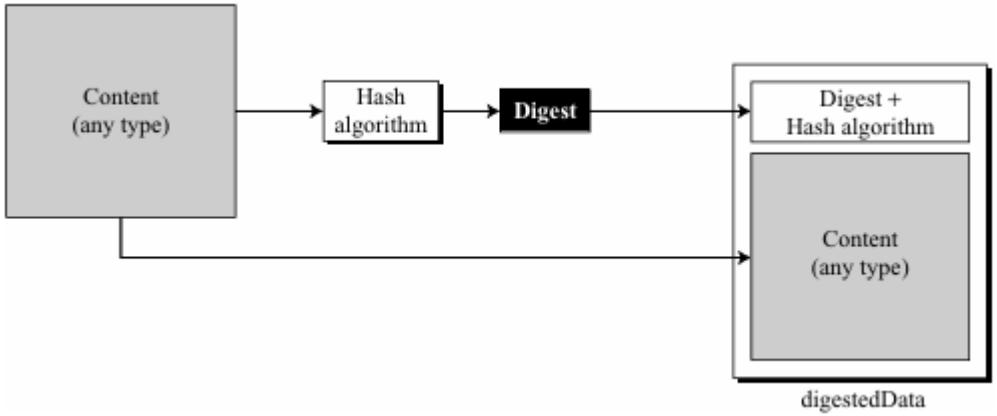


S/MIME

❑ Digested-Data Content Type

This type is used to provide integrity for the message. The result is normally used as the content for the enveloped-data content type. The encoded result is an object called digestedData.

process of creating an object of this type.



1. A message digest is calculated from the content.
2. The message digest, the algorithm, and the content are added together to create the digestedData object.

S/MIME

- **Encrypted-Data Content Type**

This type is used to create an encrypted version of any content type. Although this looks like the enveloped-data content type, the encrypted-data content type has no recipient. It can be used to store the encrypted data instead of transmitting it. The process is very simple, the user employs any key (normally driven from the password) and any algorithm to encrypt the content. The encrypted content is stored without including the key or the algorithm. The object created is called encrypted Data.

S/MIME

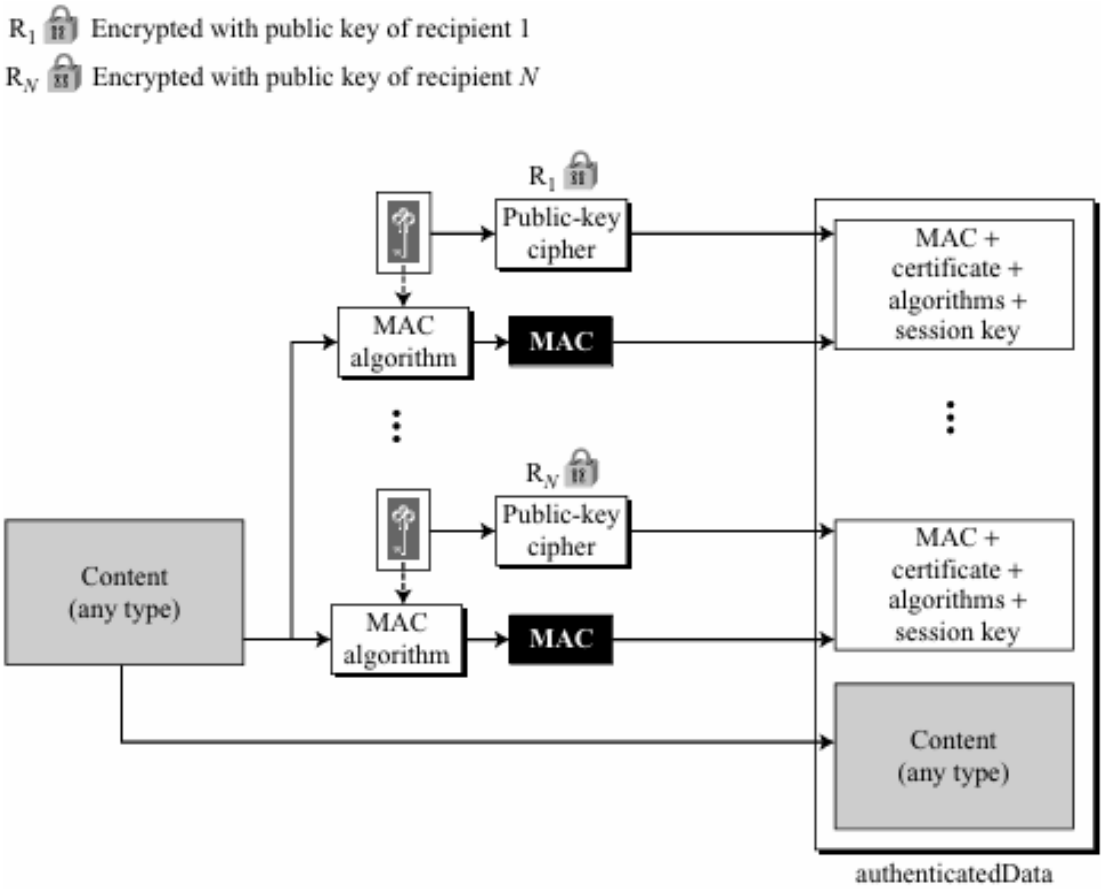
Authenticated-Data Content Type This type is used to provide authentication of the data. The object is called authenticatedData.

process

1. Using a pseudorandom generator, a MAC key is generated for each recipient.
2. The MAC key is encrypted with the public key of the recipient.
3. A MAC is created for the content.
4. The content, MAC, algorithms, and other informations are collected together to form the authenticatedData object.

S/MIME

Authenticated-Data Content Type



S/MIME

Key Management

- The key management in S/MIME is a combination of key management used by X.509 and PGP. S/MIME uses public-key certificates signed by the certificate authorities defined by X.509. However, the user is responsible to maintain the web of trust to verify signatures as defined by PGP.

S/MIME

- **Cryptographic Algorithms**

- S/MIME defines several cryptographic algorithms as shown in table below .
- The term “must” means an absolute requirement; the term “should” means recommendation.

Algorithm	*Sender must support*	*Receiver must support*	*Sender should support*	*Receiver should support*
Content-encryption algorithm	Triple DES	Triple DES		1. AES 2. RC2/40
Session-key encryption algorithm	RSA	RSA	Diffie-Hellman	Diffie-Hellman
Hash algorithm	SHA-1	SHA-1		MD5
Digest-encryption algorithm	DSS	DSS	RSA	RSA
Message-authentication algorithm		HMAC with SHA-1		

Applications of S/MIME

- It is predicted that S/MIME will become the industry choice to provide security for commercial e-mail.

Thank You