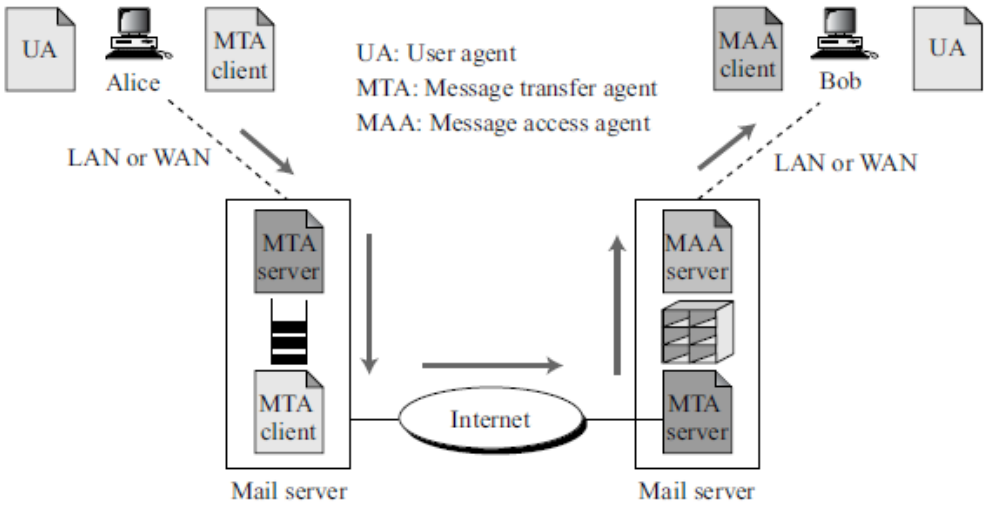
Network Security

Email Security

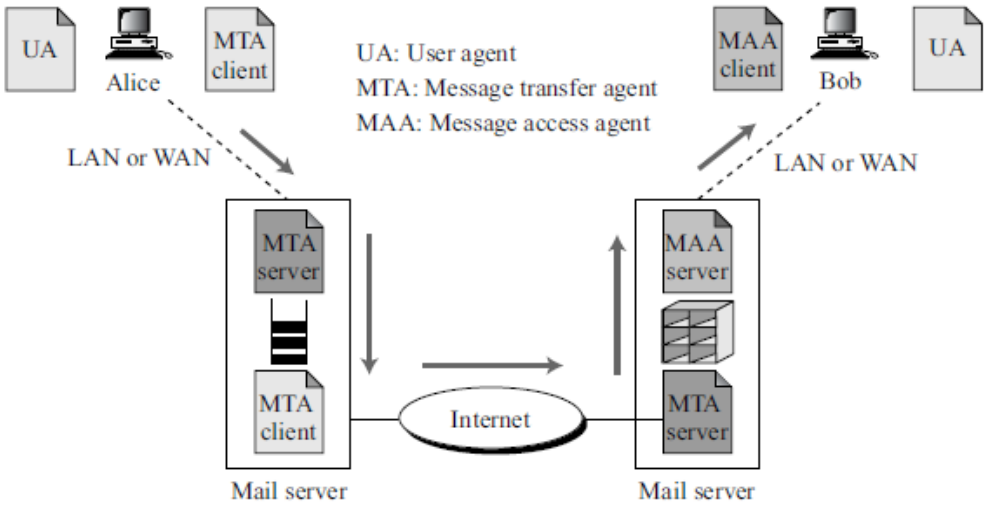
Kamalika Bhattacharjee

E-mail Architecture



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

E-mail Architecture



- 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 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.

E-mail 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 may turn off his computer when he does not need it.

E-mail Security

- Sending an e-mail is a one-time activity.
- The nature of this activity is different than IPSec or SSL, where 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.
- This is **security of a unidirectional message** because what Alice sends to Bob is totally independent from what Bob sends to Alice.
- **Cryptographic Algorithms**
- **Cryptographic Secrets**
- **Certificates**

E-mail Security

- **Cryptographic Algorithms:**
  - 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 cryptographic 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.

E-mail Security

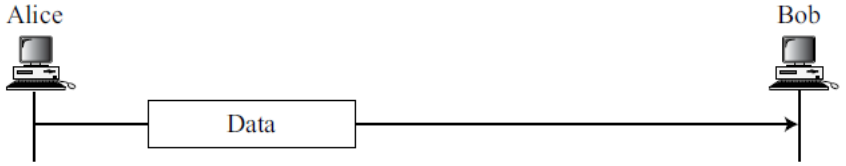
- **Cryptographic Secrets**
  - The same problem for the cryptographic algorithms applies to the cryptographic secrets (keys): If there is no negotiation, how can the two parties establish secrets between themselves?
  - Alice and Bob could use asymmetric-key algorithms for authentication and encryption, which do not require the establishment of a symmetric key. However, the use of asymmetric-key algorithms is very inefficient for the encryption/decryption of a long message.
  - 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.
  - But the secret key to decrypt the message is encrypted with the public key of the receiver and is 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.

E-mail Security

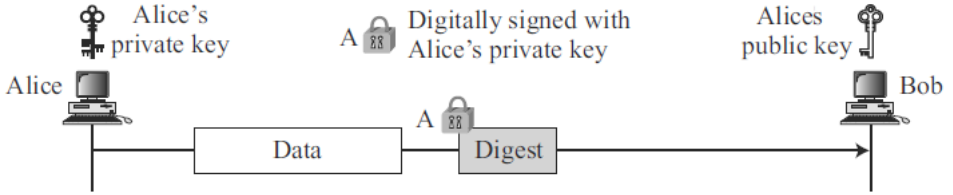
- **Certificates**
  - It is obvious that some public-key algorithms must be used for e-mail security. For example, we need to encrypt the secret key or sign the message.
  - To encrypt the secret key, Alice needs Bob's public key; to verify a signed message, Bob needs Alice's public key.
  - So, for 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.

PGP

- Pretty Good Privacy (PGP) was invented by Phil Zimmermann
- provides e-mail with privacy, integrity, and authentication.
- can be used to create a secure e-mail message or to store a file securely for future retrieval.
- **First Scenario: Plaintext**
  - no message integrity or confidentiality



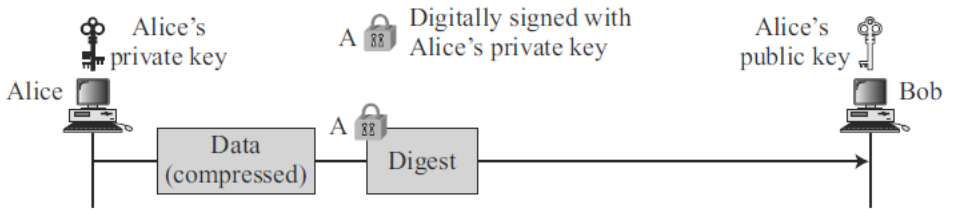
- **Next Improvement: Message Integrity**
  - Alice creates a digest of the message and signs it with her private key.
  - When Bob receives the message, he verifies the message by using Alice's public key.



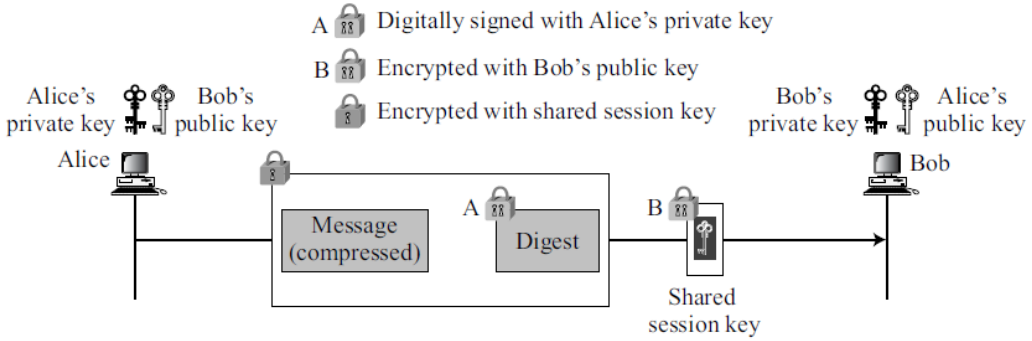
An authenticated message

PGP

- **Next Improvement: Compression**
  - to make the packet more compact
  - No security benefit



- **Confidentiality with 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.
  - When Bob receives the packet, he first decrypts the key, using his private key to remove the key.
  - He then uses the session key to decrypt the rest of the message.
  - After decompressing the rest of the message, Bob creates a digest of the message and checks to see if it is equal to the digest sent by Alice.
  - If it is, then the message is authentic.



PGP

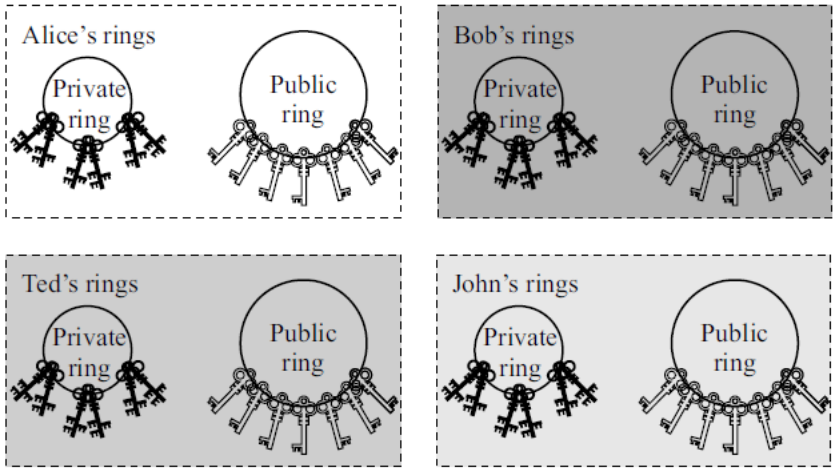
- **Code Conversion**
  - Another service provided by PGP is code conversion.
  - Most e-mail systems allow the message to consist of only ASCII characters.
  - To translate other characters not in the ASCII set, PGP uses Radix-64 conversion.
  - Each character to be sent (after encryption) is converted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

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

PGP: Key Rings

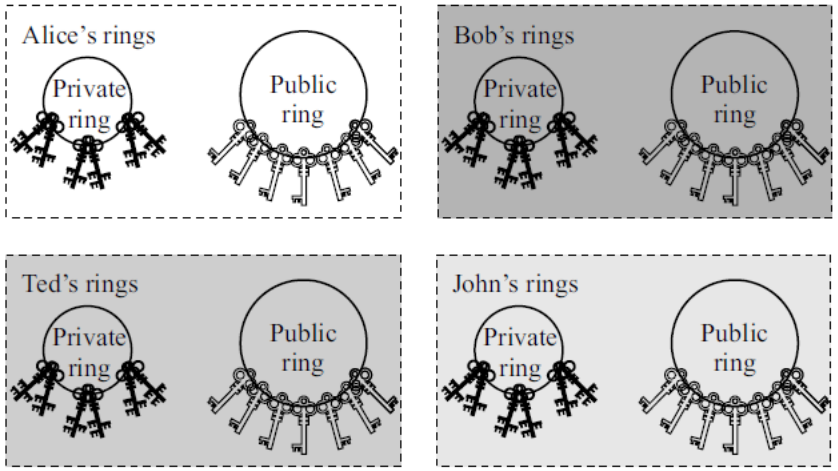
- Alice may need to send messages to many people; she needs key rings.
- In this case, Alice needs a ring of public keys, with a key belonging to each person with whom Alice needs to correspond (send or receive messages).
- In addition, the PGP designers specified a ring of private/public keys.
  - One 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. Alice may wish to use a different key pair for each group.

- Each user to have two sets of rings: a ring of private/public keys and a ring of public keys of other people
- Everyone can have more than one public key



PGP: Key Rings

- **Case 1:**
  - Alice needs to send a message to another person in the community.
  - She uses her private key to sign the digest.
  - She uses the receiver's public key to encrypt a newly created session key.
  - She encrypts the message and signed digest with the session key created.
- **Case2:**
  - Alice receives a message from another person in the community.
  - She uses her private key to decrypt the session key.
  - She uses the session key to decrypt the message and digest.
  - She uses her public key to verify the digest.



PGP Algorithms

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

Public-key algorithms

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

Symmetric-key algorithms

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

Hash Algorithms

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

Compression methods

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).
- **There is one single path from a fully trusted CA to a certificate.**

PGP Certificates

- In PGP, there is no need for CAs; anyone in the ring can sign a certificate for anyone else in ring.
- 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.
  - 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**.
- The entire operation of PGP is based on **introducer trust**, the **certificate trust**, and **the legitimacy of the public keys.**

PGP Certificates

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
- 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. Example: Alice may fully trust Bob, partially trust Anne, and not trust John at all.
- There is no mechanism in PGP to determine 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:
  - *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.*

PGP Certificates

Certificate Trust Levels

- *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.*
- *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.*
  - *John now has two certificates, one from Anne and one from Janette, each with a partial level of trust.*
- *John issues a certificate for Liz.*
  - *Alice can discard or keep this certificate with a signature trust of none.*

PGP Certificates

Key Legitimacy

- The purpose of using introducer and certificate trusts is to determine legitimacy of a public key.
- The procedure for determining key legitimacy in PGP is: 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.
- 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

- 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–> meeting or via call.
2. Bob can 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.
- 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

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

PGP Key Ring Tables

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

Private Key Ring Table

- **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 ID**. This column uniquely defines a public key among the user's public keys. In PGP, the key ID for each pair 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.
  - Sending just 8 bytes reduces the size of the message.
- **Public Key**. lists the public key belonging to a particular private key/public key pair.
- **Encrypted Private Key**. shows the encrypted value of the private key in the private key/public key pair. PGP saves only the encrypted version of the private key.
- **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.

PGP Key Ring Tables

Public Key Ring Table

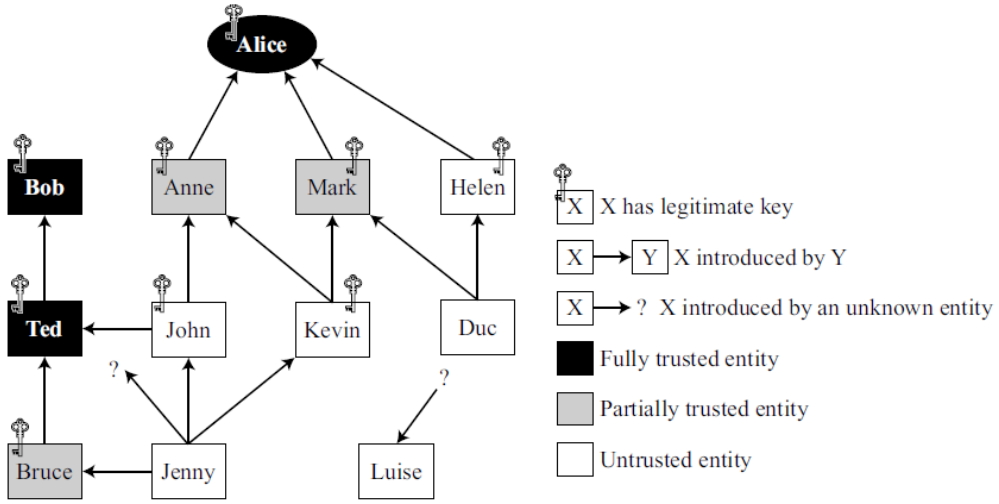
User ID	Key ID	Public key	Producer trust	Certificate(s)	Certificate trust(s)	Key Legitimacy	Timestamp
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

- **User ID, Key ID**. As in the private key ring table
- **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.
- **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. If Anne sends a certificate for John, PGP searches the row entry for Anne, finds the value of the producer trust for Anne, copies that value, and inserts it in the certificate trust field in the entry for John.
- **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.

Trust Model in PGP

- PGP can create a trust model for any user in a ring with the user as the center of activity.
- The diagram may change with any changes in the public key ring table.

- Nine entities have a legitimate key. Alice can encrypt a message to any one of these entities or verify a signature received from one of these entities (Alice's key is never used in this model).
- There are also three entities that do not have any legitimate keys with Alice.



Trust model for Alice

Web of Trust

- PGP can eventually make a web of trust between a group of people.
- If each entity introduces more entities to other entities, the public key ring for each entity gets larger and larger and entities in the ring can send secure e-mail to each other.

Example

A series of steps will show how a public key ring table is formed for Alice.

1. Start with one row, Alice herself, as shown in Table 16.6. Use N (none), P (partial), and F (full) for the levels of trust. For simplicity, also assume that everyone (including Alice) has only one user ID.

Table 16.6 *Example 2, starting table*

User ID	*Key ID*	*Public key*	*Prod. trust*	*Certificate*	*Cert. trust*	*Key legit.*	*Time-stamp*
Alice...	AB...	AB.....	F			F	.....

Note that, based on this table, we assume that Alice has issued a certificate for herself (implicitly). Alice of course trusts herself fully. The producer level of trust is also *full* and so is the key legitimacy. Although Alice never uses this first row, it is needed for the operation of PGP.

2. Now Alice adds Bob to the table. Alice fully trusts Bob, but to obtain his public key, she asks Bob to send the public key by e-mail as well as his fingerprint. Alice then calls Bob to check the fingerprint. Table 16.7 shows this new event.

Example

Table 16.7 *Example 2, after Bob is added to the table*

User ID	*Key ID*	*Public key*	*Prod. trust*	*Certificate*	*Cert. trust*	*Key legit.*	*Time-stamp*
Alice...	AB...	AB.....	F			F	.....
Bob...	12...	12.....	F			F	.....

Note that the value of the producer trust is *full* for Bob because Alice fully trusts Bob. The value of the certificate field is empty, which shows that this key has been received indirectly, and not by a certificate.

3. Now Alice adds Ted to the table. Ted is fully trusted. However, for this particular user, Alice does not have to call Ted. Instead, Bob, who knows Ted's public key, sends Alice a certificate that includes Ted's public key, as shown in Table 16.8.

Table 16.8 *Example 2, after Ted is added to the table*

User ID	*Key ID*	*Public key*	*Prod. trust*	*Certificate*	*Cert. trust*	*Key legit.*	*Time-stamp*
Alice...	AB...	AB.....	F			F	.....
Bob...	12...	12.....	F			F	.....
Ted...	48...	48.....	F	Bob's	F	F	.....

Example

Note that the value of certificate field shows that the certificate was received from Bob. The value of the certificate trust is copied by PGP from Bob's producer trust field. The value of the key legitimacy field is the value of the certificate trust multiplied by 1 (the weight).

4. Now Alice adds Anne to the list. Alice partially trusts Anne, but Bob, who is fully trusted, sends a certificate for Anne. Table 16.9 shows the new event.

Table 16.9 *Example 2, after Anne is added to the table*

User ID	*Key ID*	*Public key*	*Prod. trust*	*Certificate*	*Cert. trust*	*Key legit.*	*Time-stamp*
Alice...	AB...	AB.....	F			F	.....
Bob...	12...	12.....	F			F	.....
Ted...	48...	48.....	F	Bob's	F	F	.....
Anne...	71...	71.....	P	Bob's	F	F	.....

Note that the producer trust value for Anne is partial, but the certificate trust and key legitimacy is full.

Example

5. Now Anne introduces John, who is not trusted by Alice. Table 16.10 shows the new event.

Table 16.10 *Example 2, after John is added to the table*

User ID	*Key ID*	*Public key*	*Prod. Trust*	*Certificate*	*Cert. trust*	*Key legit.*	*Time-stamp*
Alice...	AB...	AB.....	F			F	.....
Bob...	12...	12.....	F			F	.....
Ted...	48...	48.....	F	Bob's	F	F	.....
Anne...	71...	71.....	P	Bob's	F	F	.....
John...	31...	31.....	N	Anne's	P	P	.....

Note that PGP has copied the value of Anne's producer trust (P) to the certificate trust field for John. The value of the key legitimacy field for John is 1/2 (P) at this moment, which means that Alice must not use John's key until it changes to 1 (F).

6. Now Janette, who is unknown to Alice, sends a certificate for Lee. Alice totally ignores this certificate because she does not know Janette.

Example

7. Now Ted sends a certificate for John (John, who is trusted by Ted, has probably asked Ted to send this certificate). Alice looks at the table and finds John's user ID with the corresponding key ID and public key. Alice does not add another row to the table; she just modifies the table as shown in Table 16.11.

   Because John has two certificates in Alice's table and his key legitimacy value is 1, Alice can use his key. But John is still untrustworthy. Note that Alice can continue to add entries to the table.

Table 16.11 *Example 2, after one more certificate received for John*

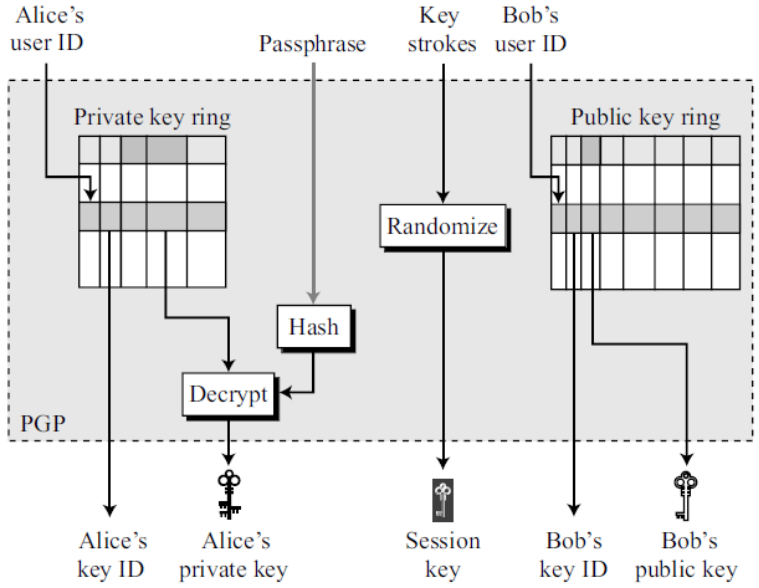
User ID	*Key ID*	*Public key*	*Prod. trust*	*Certificate*	*Cert. trust*	*Key legit.*	*Time-stamp*
Alice...	AB...	AB.....	F			F	.....
Bob...	12...	12.....	F			F	.....
Ted...	48...	48.....	F	Bob's	F	F	.....
Anne...	71...	71.....	P	Bob's	F	F	.....
John...	31...	31.....	N	Anne's Ted's	P F	F	.....

Key Revocation

- It may become necessary for an entity to revoke his or her public key from the ring.
- This may happen if the owner of the key feels that the key is compromised (stolen, for example) or just too old to be safe.
- To revoke a key, the owner can send a revocation certificate signed by herself.
- The revocation certificate must be signed by the old key and disseminated to all the people in the ring that use that public key.

Extracting Information from Rings

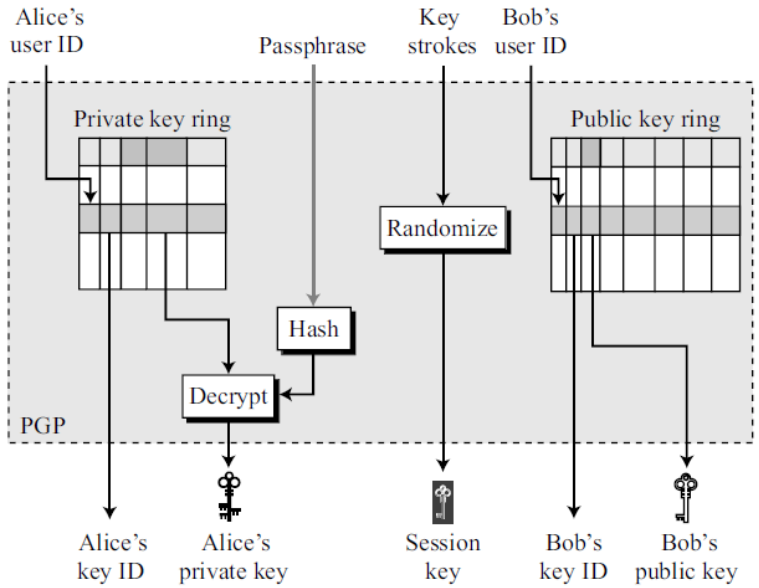
- **Sender Site**
  - Alice needs five pieces of information: the key ID of the public key she is using, her private key, the session key, Bob's public key ID, and Bob's public key.
  - To obtain these five pieces of information, Alice needs to feed four pieces of information to PGP: her user ID (for this e-mail), her passphrase, a sequence of key strokes with possible pauses, and Bob's user ID.



  - Alice's public-key ID (to be sent with the message) and her private key (to sign the message) are stored in the private key ring table. Alice selects the user ID (her e-mail address) that she wants to use as an index to this ring.
  - PGP extracts the key ID and the encrypted private key. PGP uses the predefined decryption algorithm and her hashed passphrase (as the key) to decrypt this private key.

Extracting Information from Rings

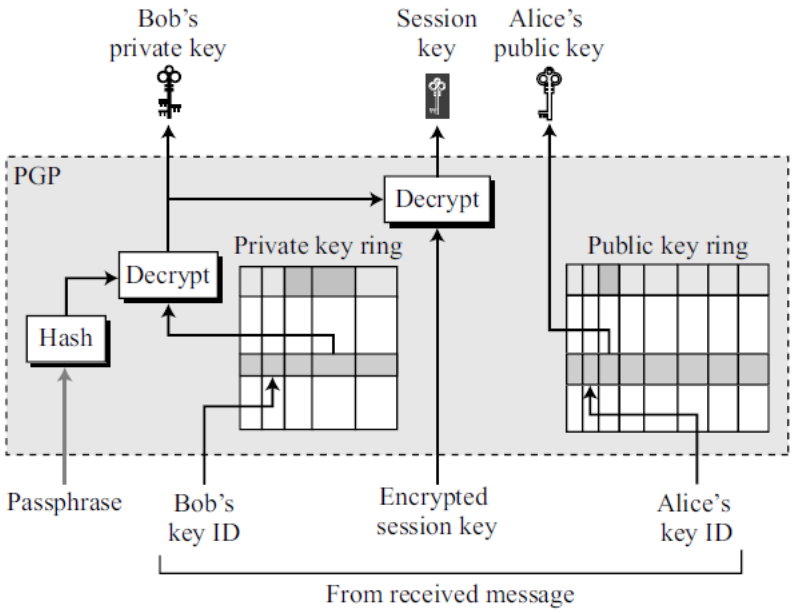
- **Sender Site**
  - The session key in PGP is a random number with a size defined in the encryption/decryption algorithm. PGP uses a random number generator to create a random session key; the seed is a set of arbitrary keystrokes typed by Alice on her keyboard.
  - Each key stroke is converted to 8 bits and each pause between the keystrokes is converted to 32 bits. The combination goes through a complex random number generator to create a very reliable random number as the session key. Note that the session key in PGP is a one-time random key and used only once.



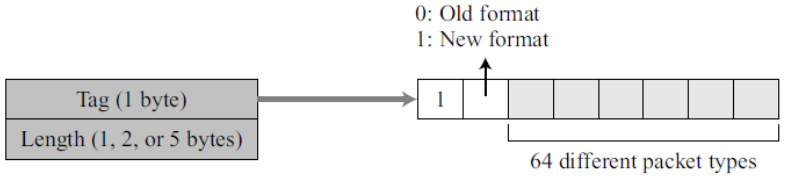
  - Alice also needs Bob's key ID (to be sent with the message) and Bob's public key (to encrypt the session key). These two pieces of information are extracted from the public key ring table using Bob's user ID (his e-mail address).

Extracting Information from Rings

- **Receiver Site**
  - At the receiver site, Bob needs three pieces of information: Bob's private key (to decrypt the session key), the session key (to decrypt the data), and Alice's public key (to verify the signature).
  - Bob uses the key ID of his public key sent by Alice to find his corresponding private key needed to decrypt the session key. This piece of information can be extracted from Bob's private key ring table.
  - The private key is encrypted when stored. Bob needs to use his passphrase and the hash function to decrypt it.
  - The encrypted session key is sent with the message; Bob uses his decrypted private key to decrypt the session key.
  - Bob uses Alice's key ID sent with the message to extract Alice's public key, which is stored in Bob's public key ring table



PGP Packets: Header



- **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 for latest version. The remaining six bits can define up to 64 different packet types
- **Length.** It is length of entire packet in bytes. Size 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
```

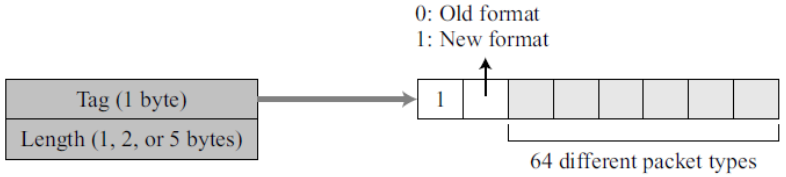
b. If the value of the byte after the tag field is between 192 and 223 (inclusive), the length field is two bytes. The length of the body can be calculated as:

```
body length = (first byte - 192) << 8 + second byte + 192
```

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

Some commonly used packet types

PGP Packets: Header



- **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 for latest version. The remaining six bits can define up to 64 different packet types
- **Length.** It is length of entire packet in bytes. Size 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.

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:

$$\text{partial body length} = 1 \ll (\text{first byte} \ \& \ 0x1F)$$

d. If the value of the byte after the tag field is 255, the length field consists of five bytes. The length of the body is calculated as:

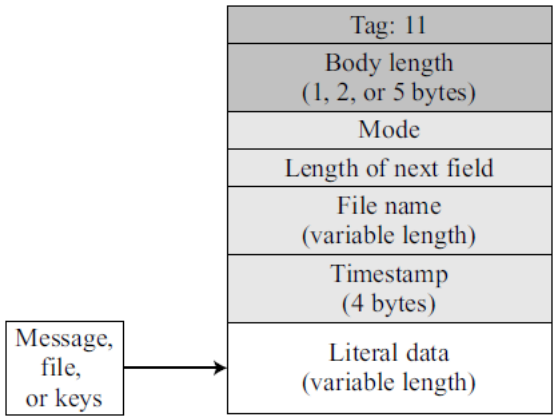
$$\text{Body length} = \text{second byte} \ll 24 \mid \text{third byte} \ll 16 \mid \text{fourth byte} \ll 8 \mid \text{fifth byte}$$

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

Some commonly used packet types

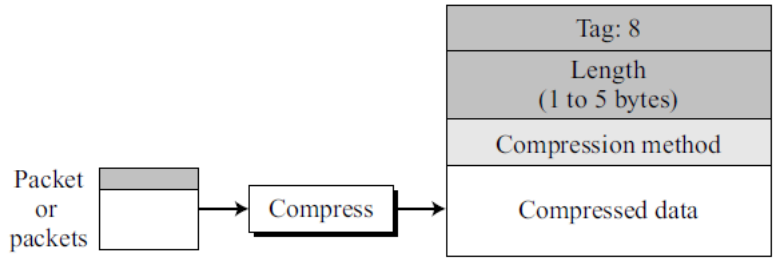
PGP Packets: Literal Data Packet

- This carries or holds the actual data that is being transmitted or stored.
- It is the most elementary type of message; that is, it cannot carry any other 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).



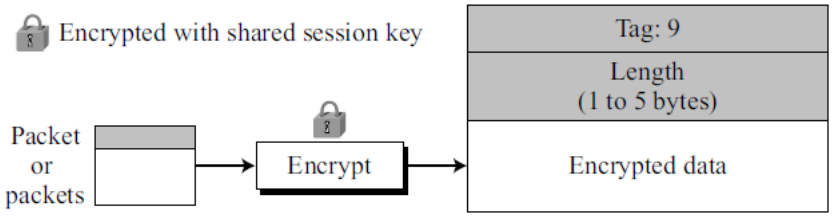
PGP Packets: Compressed Data Packet

- **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. 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.



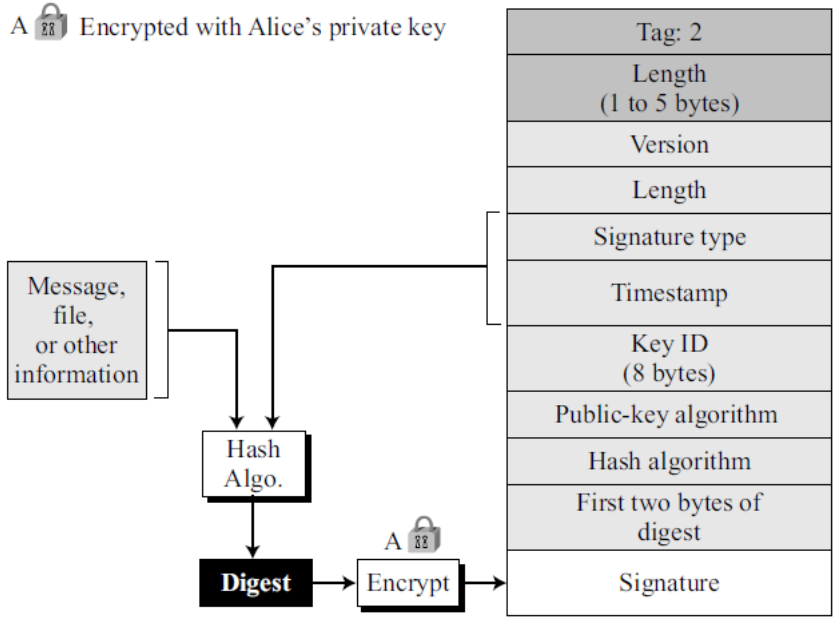
PGP Packets: Data Packet Encrypted with Secret Key

- Carries data from one packet or a combination of packets that have been encrypted using a conventional symmetric key algorithm.
- A packet carrying the one-time session key must be sent before this packet



PGP Packets: Signature Packet

- **Version**. one-byte field defines the PGP version 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**. one-byte field defines purpose of the signature, the document it signs.
- **Timestamp**. 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.

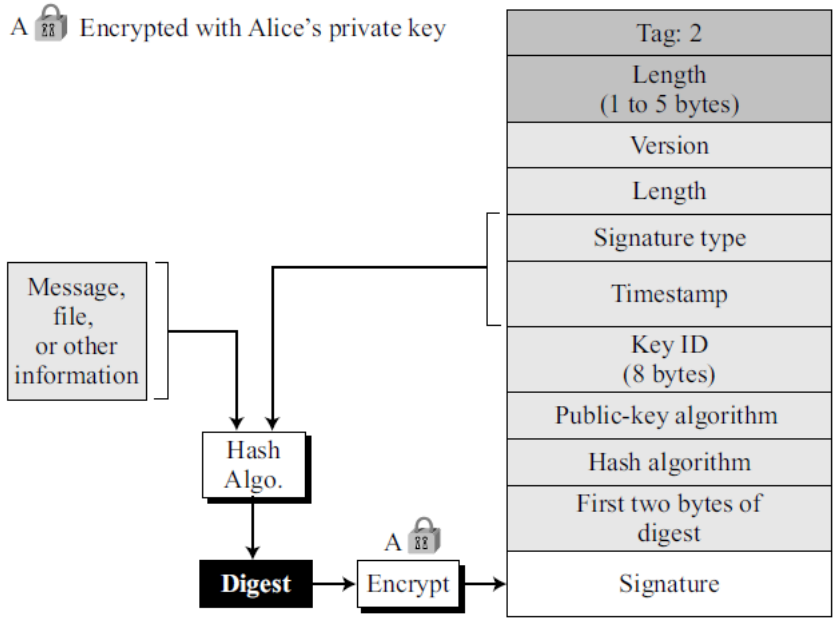


- **First two bytes of message digest.** These two bytes are used as a kind of checksum. They ensure that the receiver is using the right key ID to decrypt the digest.
- **Signature**. variable-length field, it is encrypted digest signed by the sender

PGP Packets: Signature Packet

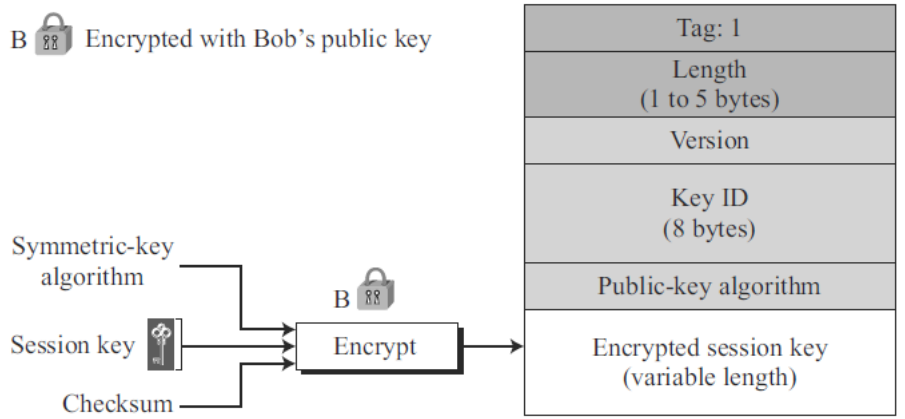
Some signature values

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 (0*x*10 through 0*x*13).



PGP Packets: Session-Key Packet Encrypted with Public Key

- used to send the session key encrypted with the receiver public key.
- **Version**. one-byte, defines the PGP version being used.
- **Key ID**. 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.
- **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:

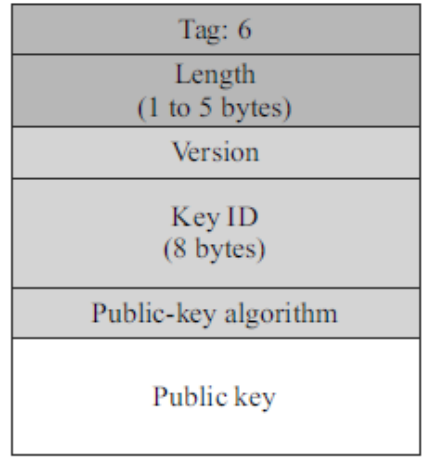


1. One-octet symmetric encryption algorithm
2. The session key
3. A two-octet checksum equal to the sum of the preceding session-key octets

PGP Packets: Public-Key & User ID

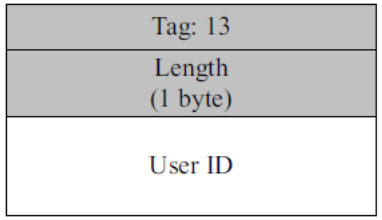
Public Key

- **Version**. one-byte field defines PGP version being used.
- **Timestamp**. four-byte field defines the time the key was created.
- **Validity**. 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.** variable-length field holds the public key itself. Its contents depend on the public-key algorithm used.



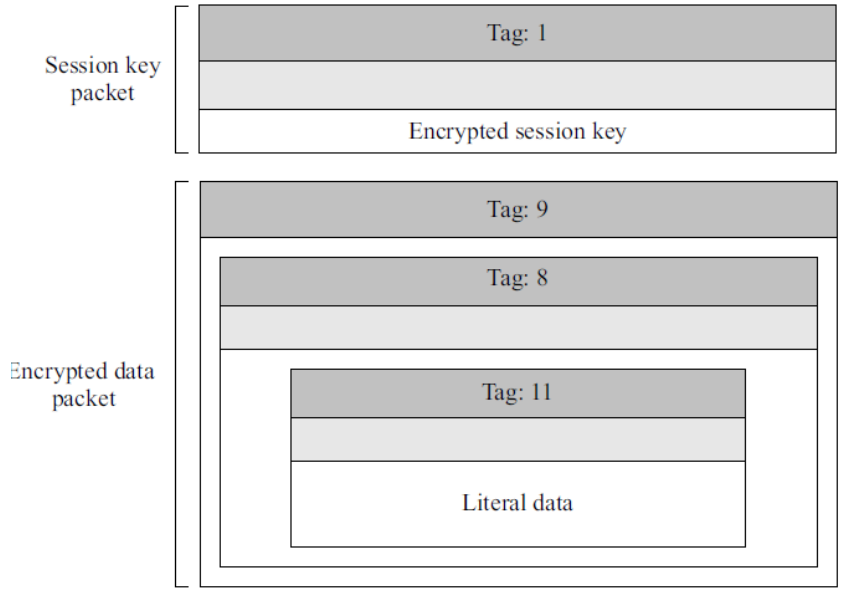
User ID

- This packet identifies a user and can normally associate the user ID contents with a public key of the sender. User ID is normally the name of the user followed by an e-mail address.



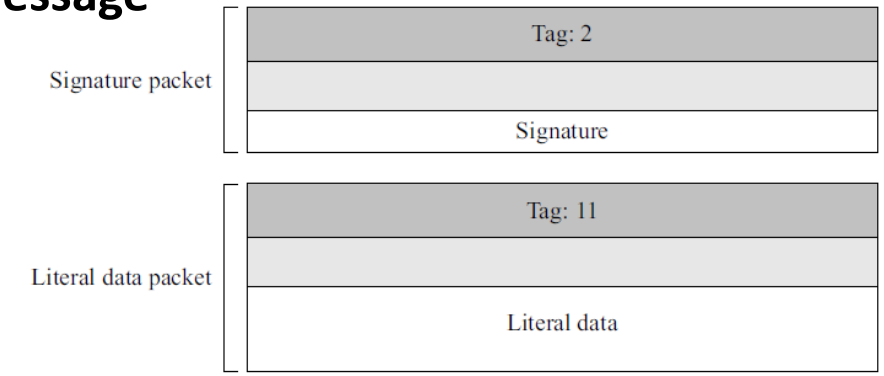
PGP Messages

- **Encrypted Message**

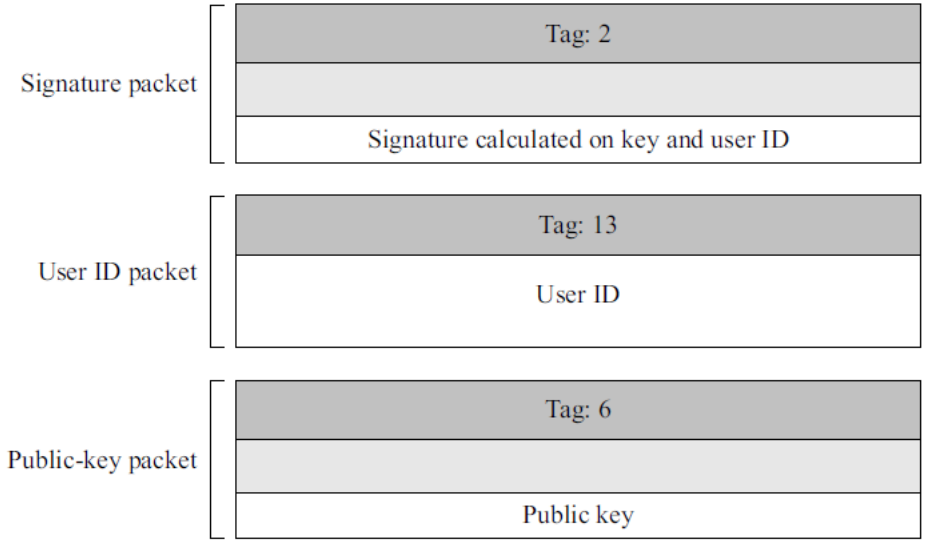


  - session-key packet is just a single packet. The encrypted data packet is made of a compressed packet.
  - The compressed packet is made of a literal data packet. The last one holds the literal data.

- **Signed Message**

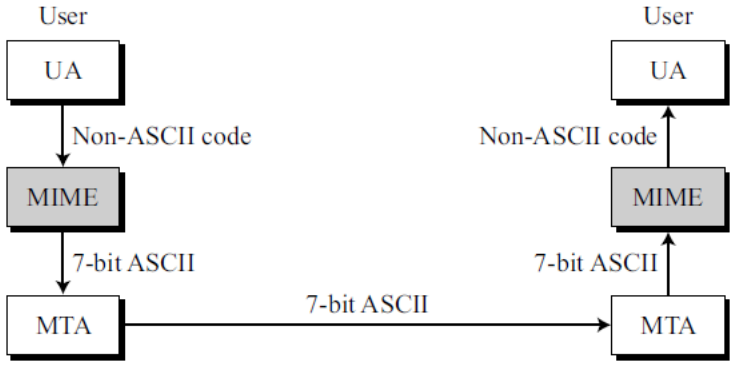


- **Certificate Message**



S/MIME

- Another security service designed for electronic mail
- It is an enhancement of the Multipurpose Internet Mail Extension (MIME) protocol
- **MIME**
  - supplementary protocol that allows non-ASCII data to be sent through e-mail.
  - transforms non-ASCII data at the sender site to NVT ASCII data and delivers it to the client MTA to be sent through the Internet.
  - The message at the receiving side is transformed back to the original data.
  - It is a set of software functions that transform non-ASCII data to ASCII data, and vice versa



S/MIME: Cryptographic Message Syntax (CMS)

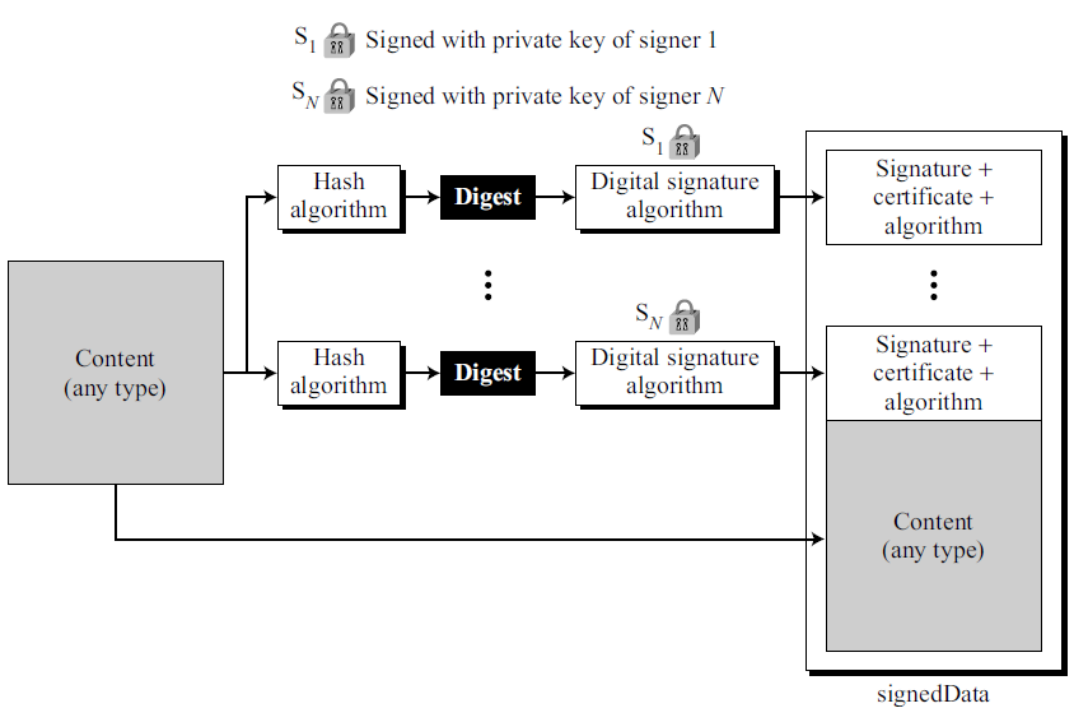
- S/MIME adds some new content types to include security services to the MIME. All of these new types include the parameter “application/pkcs7-mime,” in which “pkcs” defines “Public Key Cryptography Specification.”
- **Cryptographic Message Syntax (CMS)**
  - To define how security services, such as 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.

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

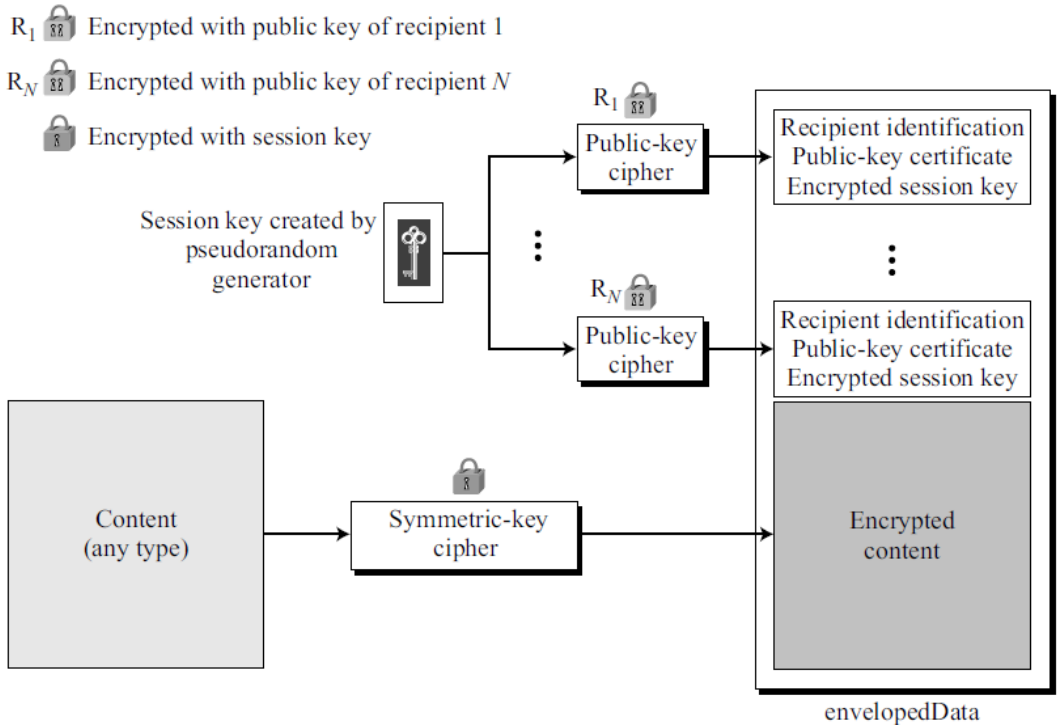
    Signed-Data Content Type This type provides only integrity of data. It contains any type and zero or more signature values. The encoded result is an object called **signedData**.

    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



Signed-data content type



Enveloped-data content type

S/MIME: Cryptographic Message Syntax (CMS)

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**.
- Steps:

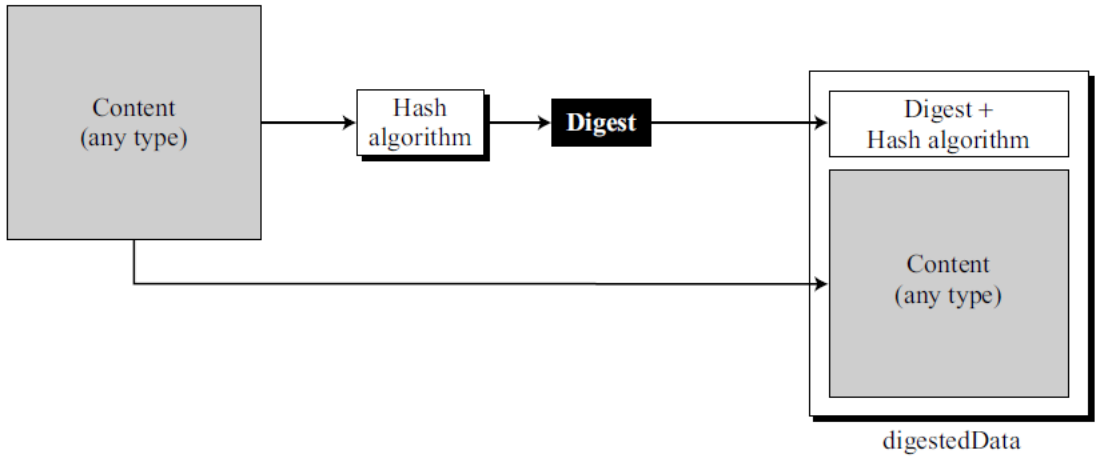
1. A pseudorandom session key is created for the symmetric-key algorithms to be used.
2. For each recipient, a copy of the session key is encrypted with the public key of each recipient.
3. The content is encrypted using the defined algorithm and created session key.
4. The encrypted contents, encrypted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

S/MIME: Cryptographic Message Syntax (CMS)

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**.
- Steps:

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: Cryptographic Message Syntax (CMS)

Encrypted-Data Content Type

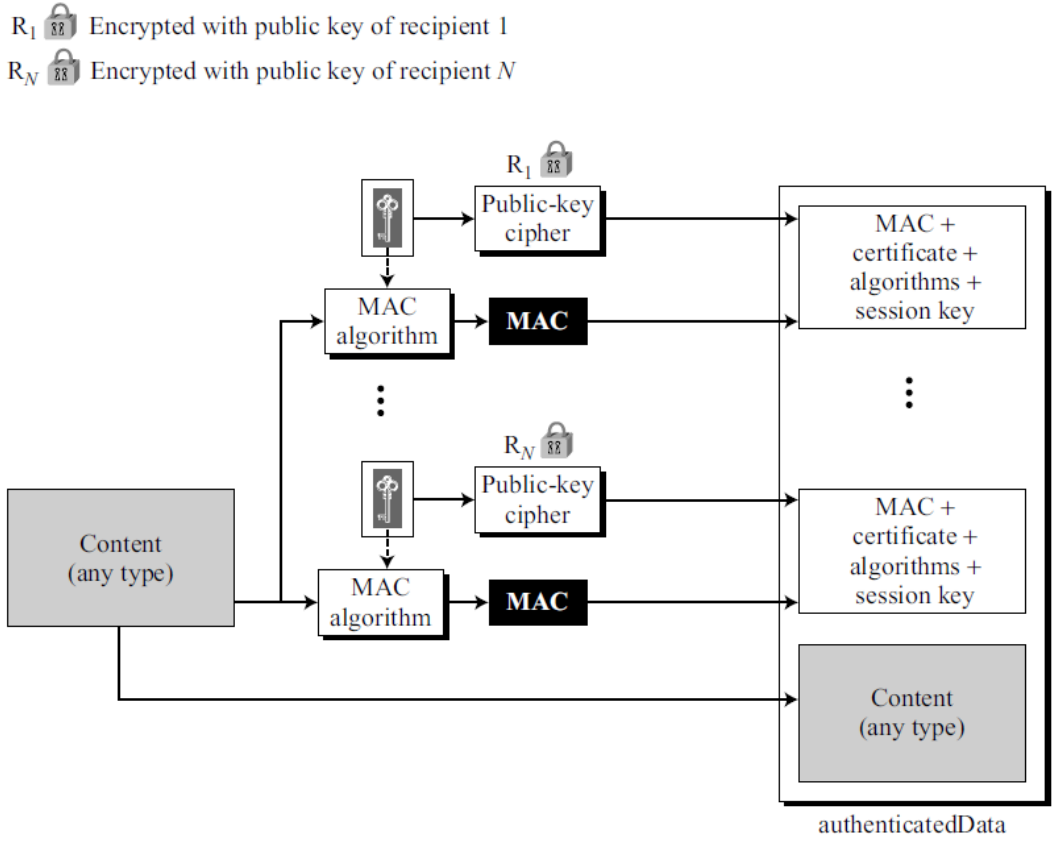
- This type is used to create an encrypted version of any content type.
- Although this looks like the enveloped-data content type, the encrypted-data content type has no recipient.
- It can be used to store the encrypted data instead of transmitting it.
- The process is very simple, the user employs 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 **encryptedData**.

S/MIME: Cryptographic Message Syntax (CMS)

Authenticated-Data Content Type

- This type is used to provide authentication of the data. The object is called **authenticatedData**.
- Steps:

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 information are collected together to form the authenticatedData object.



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.

Cryptographic Algorithms

- S/MIME defines several cryptographic algorithms. 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		

S/MIME

- An example of an enveloped-data in which a small message is encrypted using triple DES

```
Content-Type: application/pkcs7-mime; mime-type=enveloped-data
Content-Transfer-Encoding: Radix-64
Content-Description: attachment
name="report.txt";
cb32ut67f4bhijHU21oi87eryb0287hmnklsgFDoY8bc659GhIGfH6543mhjkdsaH23YjBnmN
ybmlkzjhgfdyhGe23Kjk34XiuD678Es16se09jy76jHuytTMDcbnmlkjgfFdiuyu678543m0n3h
G34un12P2454Hoi87e2ryb0H2MjN6KuyrlsgFDoY897fk923jljk1301XiuD6gh78EsUyT23y
```

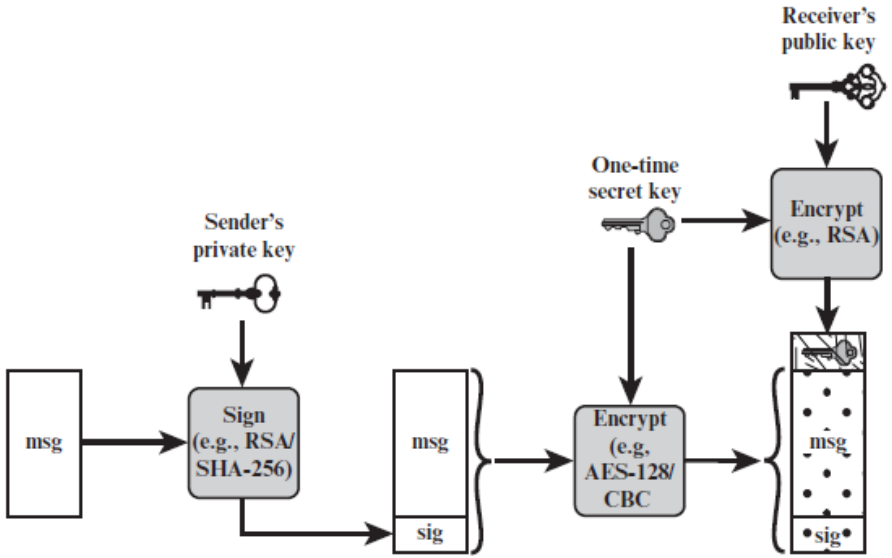
- **Applications of S/MIME**
  - It is predicted that S/MIME will become the industry choice to provide security for commercial e-mail.

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

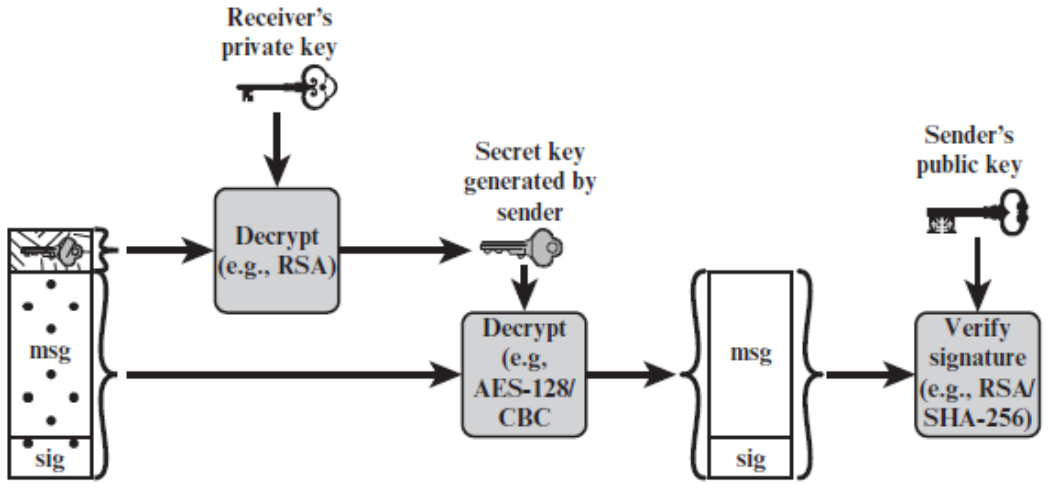
S/MIME Services

Function	Typical Algorithm	Typical Action
Digital signature	RSA/SHA-256	A hash code of a message is created using SHA-256. This message digest is encrypted using SHA-256 with the sender's private key and included with the message.
Message encryption	AES-128 with CBC	A message is encrypted using AES-128 with CBC with a one-time session key generated by the sender. The session key is encrypted using RSA with the recipient's public key and included with the message.
Compression	unspecified	A message may be compressed for storage or transmission.
Email compatibility	Radix-64 conversion	To provide transparency for email applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.

Simplified S/MIME Functional Flow



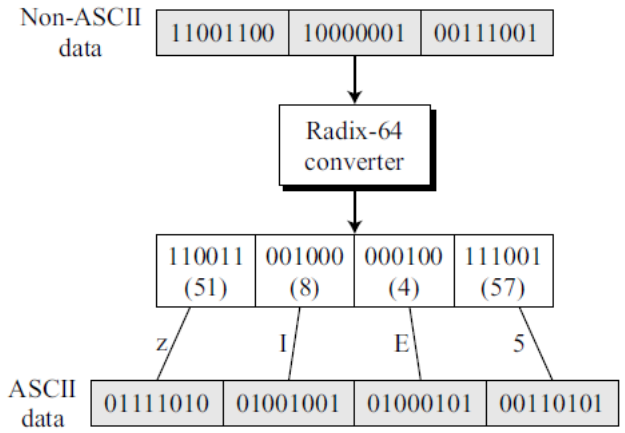
(a) Sender signs, then encrypts message



(b) Receiver decrypts message, then verifies sender's signature

Radix-64

- Radix-64 divides the binary data (made of streams of bits) into 24-bit blocks. Each block is then divided into four sections, each made of 6 bits
- Each 6-bit section is interpreted as one character according to the following table:



Radix-64 conversion

Value	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*	*Value*	*Code*
0	**A**	11	**L**	22	**W**	33	**h**	44	**s**	55	**3**
1	**B**	12	**M**	23	**X**	34	**i**	45	**t**	56	**4**
2	**C**	13	**N**	24	**Y**	35	**j**	46	**u**	57	**5**
3	**D**	14	**O**	25	**Z**	36	**k**	47	**v**	58	**6**
4	**E**	15	**P**	26	**a**	37	**l**	48	**w**	59	**7**
5	**F**	16	**Q**	27	**b**	38	**m**	49	**x**	60	**8**
6	**G**	17	**R**	28	**c**	39	**n**	50	**y**	61	**9**
7	**H**	18	**S**	29	**d**	40	**o**	51	**z**	62	+
8	**I**	19	**T**	30	**e**	41	**p**	52	**0**	63	/
9	**J**	20	**U**	31	**f**	42	**q**	53	**1**		
10	**K**	21	**V**	32	**g**	43	**r**	54	**2**		

Radix-64 encoding table

DNSSEC

- DNS Security Extensions (DNSSEC) are used by several protocols that provide email security.

Domain Name System

- DNS is a directory lookup service that provides a mapping between the name of a host on the Internet and its numeric IP address.
- Essential to the functioning of the Internet.
- DNS is used by UAs and MTAs to find the address of the next hop server for mail delivery.
- Sending MTAs query DNS for the Mail Exchange Resource Record (MX RR) of the recipient's domain (the right hand side of the"@" symbol) in order to find the receiving MTA to contact.
- Four elements comprise the DNS:
  - **Domain name space, DNS database, Name servers and Resolvers**

DNS Components

- **Domain name space**
  - DNS uses a tree-structured name space to identify resources on the Internet.
- **DNS database**
  - Conceptually, each node and leaf in the name space tree structure names a set of information (e.g., IP address, name server for this domain name) that is contained in resource record.
  - The collection of all RRs is organized into a distributed database.
- **Name servers**
  - These are server programs that hold information about a portion of the domain name tree structure and the associated RRs.
- **Resolvers**
  - These are programs that extract information from name servers in response to client requests. A typical client request is for an IP address corresponding to a given domain name.

The DNS Database

- DNS is based on a hierarchical database containing resource records (RRs) that include the **name**, **IP address**, and other information about hosts. The key features of the database are as follows:
- **Variable-depth hierarchy for names**
  - DNS allows essentially unlimited levels and uses the period (.) as the level delimiter in printed names
- **Distributed database**
  - The database resides in DNS servers scattered throughout the Internet.
- **Distribution controlled by the database**
  - The DNS database is divided into thousands of separately managed zones, which are managed by separate administrators.
  - Distribution and update of records is controlled by the database software.

The DNS Database

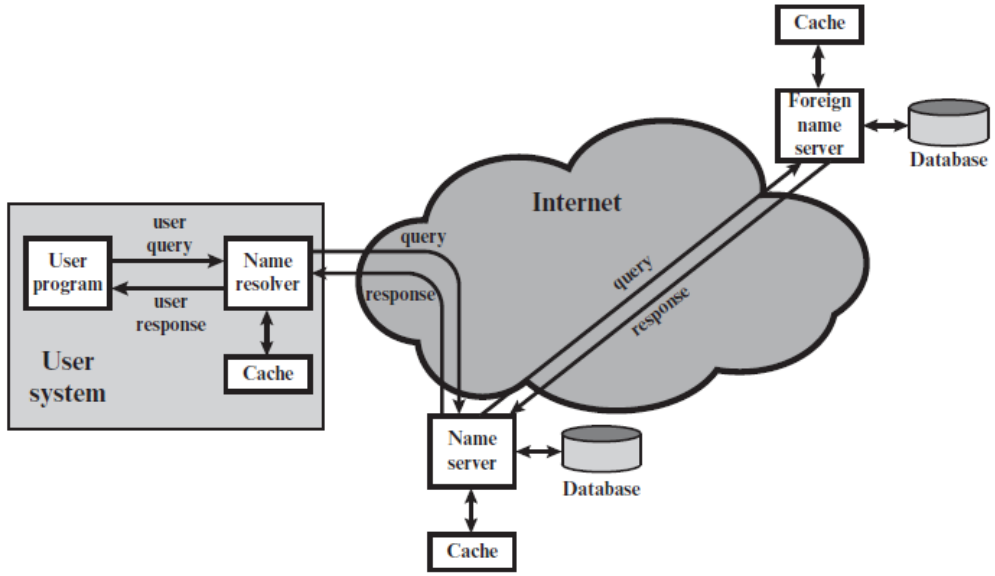
- Using this database, DNS servers provide a name-to-address directory service for network applications that need to locate specific servers.
- Example: every time an email message is sent or a Web page is accessed, there must be a DNS name lookup to determine the IP address of the email server or Web server.
- The distributed DNS database that supports the DNS functionality must be updated frequently because of the rapid and continued growth of the Internet.
- DNS must cope with dynamic assignment of IP addresses, such as is done for home DSL users by their ISP. Accordingly, dynamic updating functions for DNS have been defined.
- DNS name servers automatically send out updates to other relevant name servers as conditions warrant.

Type	Description
A	A host address. This RR type maps the name of a system to its IPv4 address. Some systems (e.g., routers) have multiple addresses, and there is a separate RR for each.
AAAA	Similar to A type, but for IPv6 addresses.
CNAME	Canonical name. Specifies an alias name for a host and maps this to the canonical (true) name.
HINFO	Host information. Designates the processor and operating system used by the host.
MINFO	Mailbox or mail list information. Maps a mailbox or mail list name to a host name.
MX	Mail exchange. Identifies the system(s) via which mail to the queried domain name should be relayed.
NS	Authoritative name server for this domain.
PTR	Domain name pointer. Points to another part of the domain name space.
SOA	Start of a zone of authority (which part of naming hierarchy is implemented). Includes parameters related to this zone.
SRV	For a given service provides name of server or servers in domain that provide that service.
TXT	Arbitrary text. Provides a way to add text comments to the database.
WKS	Well-known services. May list the application services available at this host.

Resource Record Types

DNS Operation

1. A user program requests an IP address for a domain name.
2. A resolver module in the local host or local ISP queries a local name server in the same domain as the resolver.
3. The local name server checks to see if the name is in its local database or cache, and, if so, returns the IP address to the requestor. Otherwise, the name server queries other available name servers, if necessary going to the root server.
4. When a response is received at the local name server, it stores the name/ address mapping in its local cache and may maintain this entry for the amount of time specified in the time-to-live field of the retrieved RR.
5. The user program is given the IP address or an error message.



DNSSEC

- DNSSEC provides end-to-end protection
- Uses digital signatures that are created by responding zone administrators and verified by a recipient's resolver software.
- DNSSEC avoids the need to trust intermediate name servers and resolvers that cache or route the DNS records originating from the responding zone administrator before they reach the source of the query.
- It consists of a set of new resource record types and modifications to the existing DNS protocol, and is defined in the following documents:
  - **RFC 4033, DNS Security Introduction and Requirements:** Introduces the DNS security extensions and describes their capabilities and limitations. The document also discusses the services that the DNS security extensions do and do not provide.
  - **RFC 4034, Resource Records for the DNS Security Extensions**: Defines four new resource records that provide security for DNS.
  - **RFC 4035, Protocol Modifications for the DNS Security Extensions:** Defines the concept of a signed zone, along with the requirements for serving and resolving by using DNSSEC. These techniques allow a security-aware resolver to authenticate both DNS resource records and authoritative DNS error indications

DNSSEC Operation

DNSSEC is designed to protect DNS clients from accepting forged or altered DNS resource records. It does this by using digital signatures to provide:

- **Data origin authentication:** Ensures that data has originated from the correct source.
- **Data integrity verification:** Ensures that the content of a RR has not been modified.
- The DNS zone administrator digitally signs every Resource Record set (RRset) in the zone, and publishes this collection of digital signatures, along with the zone administrator's public key, in the DNS itself.
- In DNSSEC, trust in the public key (for signature verification) of the source is established not by going to a third party or a chain of third parties (as in public key infrastructure [PKI] chaining)
- It is done by starting from a trusted zone (such as the root zone) and establishing the chain of trust down to the current source of response through successive verifications of signature of the public key of a child by its parent.
- The public key of the trusted zone is called the *trust anchor*.

DNSSEC Resource Records

- **DNSKEY**: Contains a public key.
- **RRSIG**: A resource record digital signature.
- **NSEC**: Authenticated denial of existence record.
- **DS**: Delegation signer.
- An RRSIG is associated with each RRset, where an RRset is the set of resource records that have the same label, class, and type.
- When a client requests data, an RRset is returned, together with the associated digital signature in an RRSIG record.
- The client obtains the relevant DNSKEY public key and verifies the signature for this RRset.

DNSSEC

- DNSSEC depends on establishing the authenticity of the DNS hierarchy leading to the domain name in question
- Its operation depends on beginning the use of cryptographic digital signatures in the root zone.
- The DS resource record facilitates key signing and authentication between DNS zones to create an authentication chain, or trusted sequence of signed data, from the root of the DNS tree down to a specific domain name.
- To secure all DNS lookups, including those for non-existent domain names and record types, DNSSEC uses the NSEC resource record to authenticate negative responses to queries.
- NSEC is used to identify the range of DNS names or resource record types that do not exist among the sequence of domain names in a zone.

Email Treats and Mitigations

Threat	Impact on Purported Sender	Impact on Receiver	Mitigation
Email sent by unauthorized MTA in enterprise (e.g., malware botnet)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered into user inboxes.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email message sent using spoofed or unregistered sending domain	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered into user inboxes.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email message sent using forged sending address or email address (i.e., phishing, spear phishing)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered. Users may inadvertently divulge sensitive information or PII.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email modified in transit	Leak of sensitive information or PII.	Leak of sensitive information, altered message may contain malicious information.	Use of TLS to encrypt email transfer between servers. Use of end-to-end email encryption.
Disclosure of sensitive information (e.g., PII) via monitoring and capturing of email traffic	Leak of sensitive information or PII.	Leak of sensitive information, altered message may contain malicious information.	Use of TLS to encrypt email transfer between servers. Use of end-to-end email encryption.
Unsolicited Bulk Email (UBE) (i.e., spam)	None, unless purported sender is spoofed.	UBE and/or email containing malicious links may be delivered into user inboxes.	Techniques to address UBE.
DoS/DDoS attack against an enterprises' email servers	Inability to send email.	Inability to receive email.	Multiple mail servers, use of cloud-based email providers.

SP 800-177 recommends use of a variety of standardized protocols as a means for countering these threats.

- **STARTTLS:**
  - An SMTP security extension that provides authentication, integrity, non-repudiation (via digital signatures) and confidentiality (via encryption) for the entire SMTP message by running SMTP over TLS.
- **S/MIME:**
  - Provides authentication, integrity, non-repudiation (via digital signatures) and confidentiality (via encryption) of the message body carried in SMTP messages.

Email Treats and Mitigations

Threat	Impact on Purported Sender	Impact on Receiver	Mitigation
Email sent by unauthorized MTA in enterprise (e.g., malware botnet)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered into user inboxes.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email message sent using spoofed or unregistered sending domain	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered into user inboxes.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email message sent using forged sending address or email address (i.e., phishing, spear phishing)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered. Users may inadvertently divulge sensitive information or PII.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email modified in transit	Leak of sensitive information or PII.	Leak of sensitive information, altered message may contain malicious information.	Use of TLS to encrypt email transfer between servers. Use of end-to-end email encryption.
Disclosure of sensitive information (e.g., PII) via monitoring and capturing of email traffic	Leak of sensitive information or PII.	Leak of sensitive information, altered message may contain malicious information.	Use of TLS to encrypt email transfer between servers. Use of end-to-end email encryption.
Unsolicited Bulk Email (UBE) (i.e., spam)	None, unless purported sender is spoofed.	UBE and/or email containing malicious links may be delivered into user inboxes.	Techniques to address UBE.
DoS/DDoS attack against an enterprises' email servers	Inability to send email.	Inability to receive email.	Multiple mail servers, use of cloud-based email providers.

- **DNS Security Extensions (DNSSEC):**
  - Provides authentication and integrity protection of DNS data, and is an underlying tool used by various email security protocols.
- **DNS-based Authentication of Named Entities (DANE):**
  - Is designed to overcome problems in the certificate authority (CA) system by providing an alternative channel for authenticating public keys based on DNSSEC, with the result that the same trust relationships used to certify IP addresses are used to certify servers operating on those addresses.

Email Treats and Mitigations

Threat	Impact on Purported Sender	Impact on Receiver	Mitigation
Email sent by unauthorized MTA in enterprise (e.g., malware botnet)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered into user inboxes.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email message sent using spoofed or unregistered sending domain	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered into user inboxes.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email message sent using forged sending address or email address (i.e., phishing, spear phishing)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered. Users may inadvertently divulge sensitive information or PII.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email modified in transit	Leak of sensitive information or PII.	Leak of sensitive information, altered message may contain malicious information.	Use of TLS to encrypt email transfer between servers. Use of end-to-end email encryption.
Disclosure of sensitive information (e.g., PII) via monitoring and capturing of email traffic	Leak of sensitive information or PII.	Leak of sensitive information, altered message may contain malicious information.	Use of TLS to encrypt email transfer between servers. Use of end-to-end email encryption.
Unsolicited Bulk Email (UBE) (i.e., spam)	None, unless purported sender is spoofed.	UBE and/or email containing malicious links may be delivered into user inboxes.	Techniques to address UBE.
DoS/DDoS attack against an enterprises' email servers	Inability to send email.	Inability to receive email.	Multiple mail servers, use of cloud-based email providers.

- **Sender Policy Framework (SPF):**
  - Uses the Domain Name System (DNS) to allow domain owners to create records that associate the domain name with a specific IP address range of authorized message senders.
  - It is a simple matter for receivers to check the SPF TXT record in the DNS to confirm that the purported sender of a message is permitted to use that source address and reject mail that does not come from an authorized IP address.

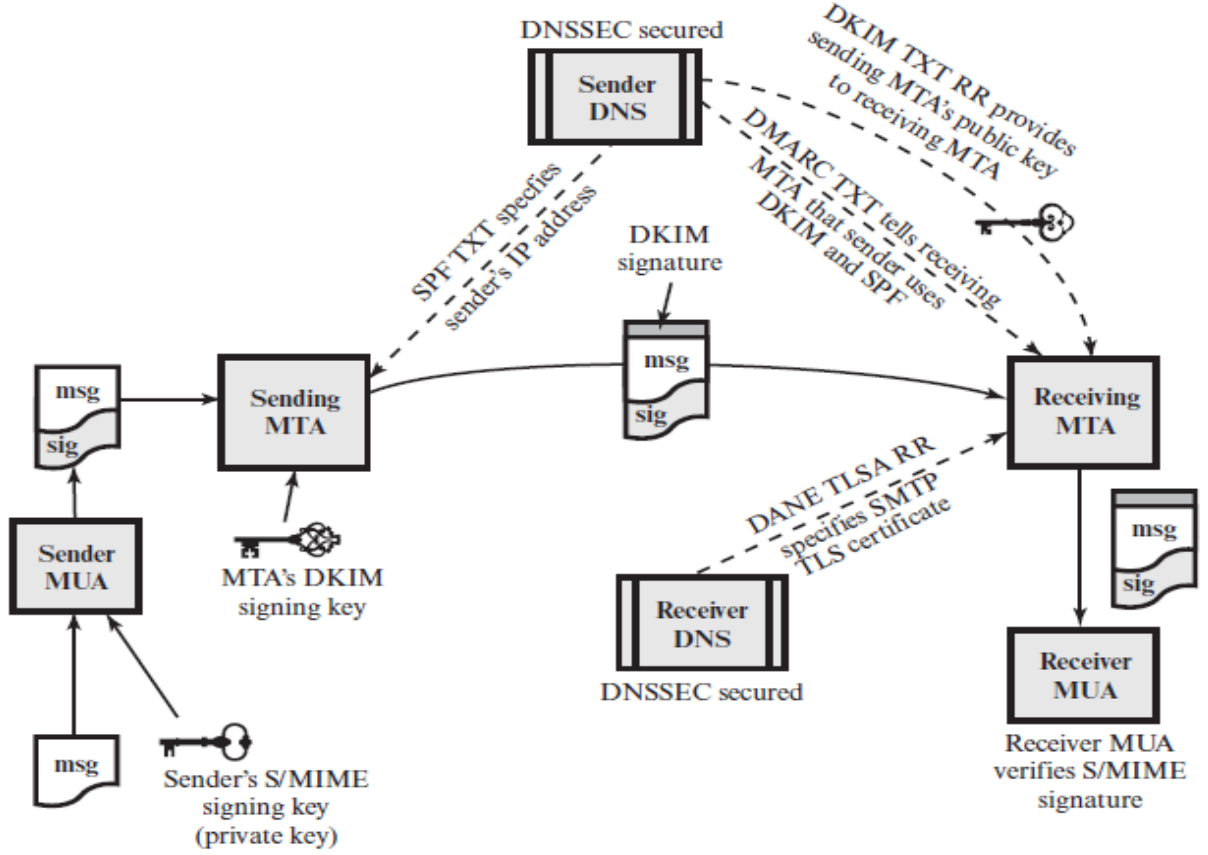
Email Treats and Mitigations

Threat	Impact on Purported Sender	Impact on Receiver	Mitigation
Email sent by unauthorized MTA in enterprise (e.g., malware botnet)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered into user inboxes.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email message sent using spoofed or unregistered sending domain	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered into user inboxes.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email message sent using forged sending address or email address (i.e., phishing, spear phishing)	Loss of reputation, valid email from enterprise may be blocked as possible spam/phishing attack.	UBE and/or email containing malicious links may be delivered. Users may inadvertently divulge sensitive information or PII.	Deployment of domain-based authentication techniques. Use of digital signatures over email.
Email modified in transit	Leak of sensitive information or PII.	Leak of sensitive information, altered message may contain malicious information.	Use of TLS to encrypt email transfer between servers. Use of end-to-end email encryption.
Disclosure of sensitive information (e.g., PII) via monitoring and capturing of email traffic	Leak of sensitive information or PII.	Leak of sensitive information, altered message may contain malicious information.	Use of TLS to encrypt email transfer between servers. Use of end-to-end email encryption.
Unsolicited Bulk Email (UBE) (i.e., spam)	None, unless purported sender is spoofed.	UBE and/or email containing malicious links may be delivered into user inboxes.	Techniques to address UBE.
DoS/DDoS attack against an enterprises' email servers	Inability to send email.	Inability to receive email.	Multiple mail servers, use of cloud-based email providers.

- **Domain Keys Identified Mail (DKIM):**
  - Enables an MTA to sign selected headers and the body of a message.
  - This validates the source domain of the mail and provides message body integrity.
- **Domain-based Message Authentication, Reporting, and Conformance (DMARC):**
  - Lets senders know the proportionate effectiveness of their SPF and DKIM policies, and signals to receivers what action should be taken in various individual and bulk attack scenarios.

The Interrelationship between Protocols

- The Interrelationship of DNSSEC, SPF, DKIM, DMARC, DANE, and S/MIME for Assuring Message Authenticity and Integrity



DANE = DNS-based Authentication of Named Entities
DKIM = DomainKeys Identified Mail
DMARC = Domain-based Message Authentication, Reporting, and Conformance
DNSSEC = Domain Name System Security Extensions
SPF = Sender Policy Framework
S/MIME = Secure Multi-Purpose Internet Mail Extensions
TLSA RR = Transport Layer Security Authentication Resource Record