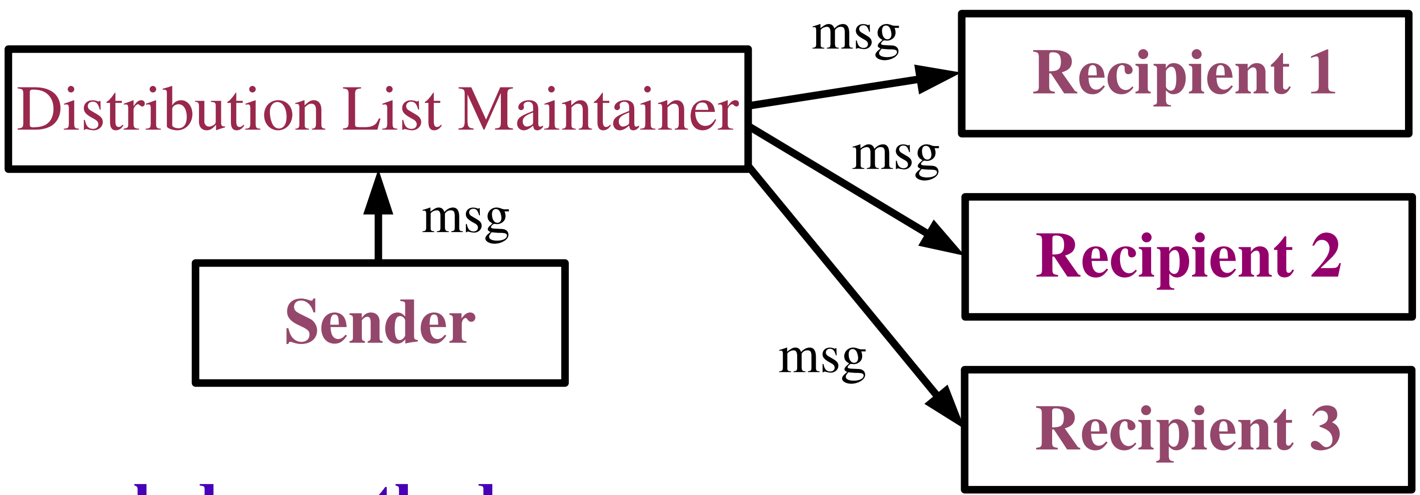
Email Security

Email Security

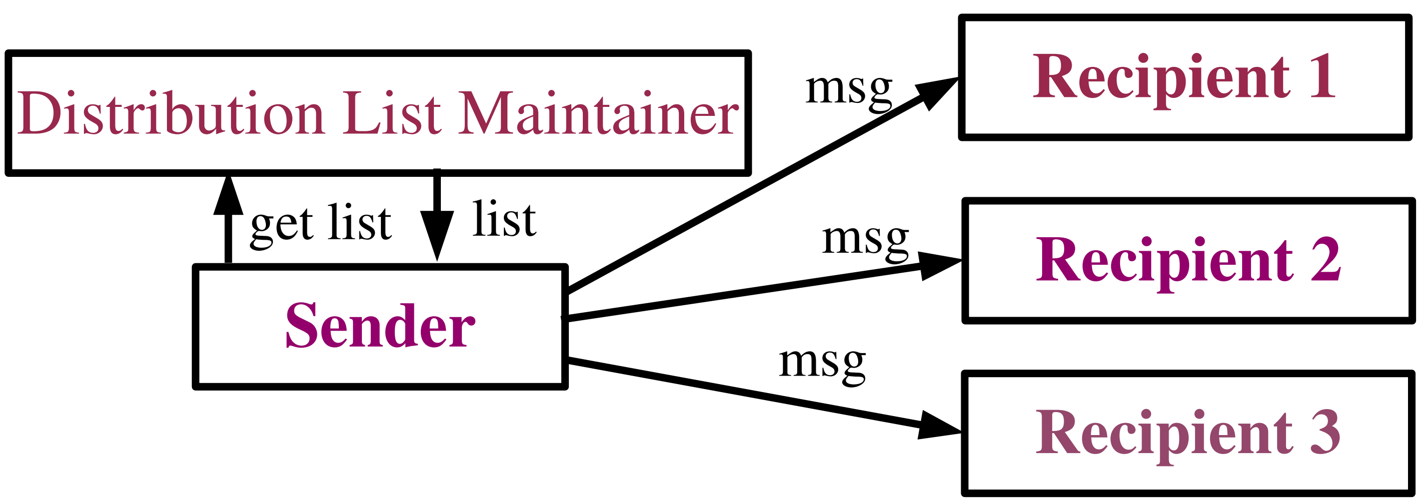
Issues:

- Not real-time, can afford to use public key cryptosystems more.
- Certification of keys is much harder because anyone can send anyone else some mail
- Strictly end-to-end, IPSec/firewalls might get in the way here
- A single message can be sent to many parties
- A single message can be sent to one or more distribution lists
- There can be message forwarding loops due to distribution lists or even someone's `.forward` file
- Duplicate copies can be sent to same individual
- Recipient or intermediate node may not be ready to receive mail

Email Security



Remote exploder method



Local exploder method

Email Security

Comparison:

Local exploder method has some advantages:

- Easier to prevent mail forwarding loops caused by the sender
- Sender may be able to prevent duplicate copies to same recipient
- Sender knows in advance what traffic will be generated (may be important if billing is based on traffic)

Email Security

Comparison:

Local exploder method has some advantages:

Easier to prevent mail forwarding loops caused by the sender

Sender may be able to prevent duplicate copies to same recipient

Sender knows in advance what traffic will be generated (may be important if billing is based on traffic)

Remote exploder method has some advantages:

You can send to a list of people you are not allowed to know

Lots of traffic may be generated *away* from sender's network

If distribution list is huge it is economical to have mail sent by a list maintainter

Distribution lists may explode to other lists – the number of recipients would be too hard for a sender to keep up with. Parallelism is exploited!

Email Security

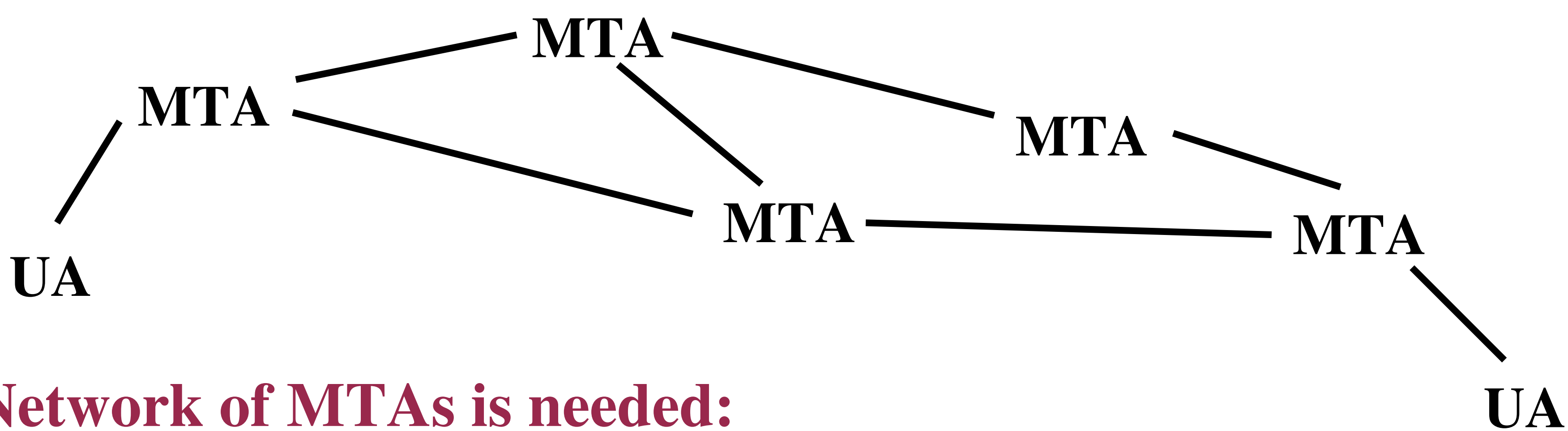
Store and Forward: users are not always ready to receive email

Message Transfer Agent:

Node whereby mail is forwarded to another node

User Agent: the email client

Node where mail is processed



Network of MTAs is needed:

One path from source to destination might be intermittent

MTAs may need to authenticate over MTAs (find trusted chain)

Company desires "security gateway" (only email allowed at node)

Different parts of network may use different protocols (TCP/OSI)

Email Security

Protocols:

Simple Mail Transfer Protocol (SMTP):

Text-based commands for forwarding email between UA-MSA (mail submission agent) MSA-MTA, MTA-MTA, MTA-MDA (mail delivery agent)

Internet Message Access Protocol (IMAP):

Allows UA to access mail stored by MDA. Supports

- Several clients can be connected to the same mailbox
- Separate retrieval of MIME parts of a message (e.g. attachment)
- IMAP over SSL (IMAPS)

Post Office Protocol:

Another popular mail retrieval protocol.

- Client connects, gets email, deletes messages on server
- One client can connect at a time
- POP3 over SSL (POP3S)

Email Security

Security Services over Email:

- **Privacy**: No one should read message except recipient
- **Authentication**: Recipient should know exactly who the sender is
- **Integrity**: Recipient should be able to tell whether message was altered in transit
- **Non-repudiation**: Recipient can prove that the sender really sent it
- **Proof of submission**: Verification to the sender that the mailer got it
- **Proof of delivery**: Verification to sender that the recipient got it
- **Message flow confidentiality**: Eavesdropper cannot determine the sender's ID
- **Anonimity**: Ability to send so recipient does not know sender
- **Containment**: Ability to keep secure messages from "leaking" out of a region
- **Audit**: Logging of events having relevance to security
- **Accounting**: Maintain usage statistics (might charge for service)
- **Self-destruct**: Message is destroyed on delivery
- **Message sequence integrity**: Sequence of messages have arrived in order, without loss

Email Security

Establishing Keys:

Most services are best provided using cryptographic means

But the email infrastructure may require many keys – where are they?

Email Security

Establishing Keys:

Most services are best provided using cryptographic means
But the email infrastructure may require many keys – where are they?

Establishing Public Keys:

Receiver may have sent it by some other means - say NY times
Receiver may have appended it to an email message (signed)
Receiver may have certified it though a CA
Receiver may have posted it on a Public Key Infrastructure

Email Security

Establishing Keys:

Most services are best provided using cryptographic means
But the email infrastructure may require many keys – where are they?

Establishing Public Keys:

Receiver may have sent it by some other means - say NY times
Receiver may have appended it to an email message (signed)
Receiver may have certified it though a CA
Receiver may have posted it on a Public Key Infrastructure

Establishing Secret Keys:

Both parties meet in private to set a key
Communicate on the phone
Sender gets a "ticket" from a KDC and includes it in the message

Email Security

Privacy – needed because:

Eavesdropper can easily listen - especially at "No Such Agency"

Mail can be rerouted - never to reach intended recipient

Conflicts - employee wants privacy, company wants assurance
employee is not giving away company secrets

End-to-end Privacy:

Problem: how to encrypt lots of copies to multiple recipients?

Secret key encryption is preferable since it is 1000 times faster

Not desirable to use a long-term key more than needed

Email Security

Privacy – needed because:

Eavesdropper can easily listen - especially at "No Such Agency"

Mail can be rerouted - never to reach intended recipient

Conflicts - employee wants privacy, company wants assurance
employee is not giving away company secrets

End-to-end Privacy:

Problem: how to encrypt lots of copies to multiple recipients?

Secret key encryption is preferable since it is 1000 times faster

Not desirable to use a long-term key more than needed

Hence: sender may choose a secret S only for encrypting message

msg encrypted with S + S encrypted with public key -> recipient

S encrypted multiple times with recipients' public keys

Email Security

Privacy – needed because:

Eavesdropper can easily listen - especially at "No Such Agency"

Mail can be rerouted - never to reach intended recipient

Conflicts - employee wants privacy, company wants assurance
employee is not giving away company secrets

End-to-end Privacy:

Problem: how to encrypt lots of copies to multiple recipients?

Secret key encryption is preferable since it is 1000 times faster

Not desirable to use a long-term key more than needed

Hence: sender may choose a secret S only for encrypting message

msg encrypted with S + S encrypted with public key -> recipient

S encrypted multiple times with recipients' public keys

```
To: prakash, masterblaster, cokane
From:  franco
Key-info: prakash-7567484385785467
Key-info: masterblaster-734478868274684
Key-info: cokane-9062346667642424
Msg-info: jkdiuqwydfkjhjdfreuigfkjsdfkjsyfuieihfuigf
```

Email Security

Privacy with Distribution List Exploders:

Problem: sender may not know public keys of recipients

Sender must have a key K shared with the distribution list exploder

Sender encrypts message with a secret S and sends it with S encrypted using K to the list exploder

Distribution list exploder decrypts S then re-encrypts it with the keys of recipients (without decrypting the email?) and sends the email forward (possibly to other distribution list exploders).

But now sender loses some assurance that the message arrives as intended

Email Security

Authentication of the Source:

Prevent `C` from sending mail to `B` with `'From: A'`

Email Security

Authentication of the Source:

Prevent C from sending mail to B with `'From: A'`

Public Keys:

- Sender signs hash of message with its private key
- Works on multiple messages (same signature!)
- Public key might be sent with the message with a chain of certificates

Email Security

Authentication of the Source:

Prevent `C` from sending mail to `B` with `'From: A'`

Public Keys:

Sender signs hash of message with its private key

Works on multiple messages (same signature!)

Public key might be sent with the message with a chain of certificates

Secret Keys:

Sender computes a MAC: one of

- CBC residue of the message computed with shared secret
- Hash of shared secret appended to message
- Encrypted message digest of message

Multiple emails: use 3rd method - compute MD once, then encrypt for each addressee

Email Security

Authentication of the Source - Distribution Exploders:

Public keys: Just forward the messages as is, use sender's public key to authenticate

Email Security

Authentication of the Source - Distribution Exploders:

<u>Public keys</u>: Just forward the messages as is, use sender's public key to authenticate

<u>Secret keys</u>: Sender cannot be assumed to share secrets with all recipients or know who all the recipients are

Distribution list exploder must remove sender's authentication information from emails and replace it with its own

Distribution list exploder must verify the source of the email because recipients cannot do that themselves - although they can authenticate the exploder

Exploder may need to include the name of the sender in the body of the encrypted email.

Email Security

Message Integrity:

Source authentication methods also provide message integrity

Email Security

Message Integrity:

Source authentication methods also provide message integrity

Does it make sense to provide integrity without authentication?

Email Security

Message Integrity:

Source authentication methods also provide message integrity

Does it make sense to provide integrity without authentication?

Application: send a ransom note

Email Security

Message Integrity:

Source authentication methods also provide message integrity

Does it make sense to provide integrity without authentication?
Application: send a ransom note

Message integrity without source authentication meaningless without encryption since someone could replace the message with a completely different one and the recipient could not tell

Email Security

Message Integrity:

Source authentication methods also provide message integrity

Does it make sense to provide integrity without authentication?
Application: send a ransom note

Message integrity without source authentication meaningless without encryption since someone could replace the message with a completely different one and the recipient could not tell

Can't do message integrity check without source authentication with secret key technology since both parties must know each other to be able to use the same secret

Email Security

Non-Repudiation:

With public keys easy to provide non-repudiation authentication

With secret keys easy to provide repudiable authentication

Email Security

Non-Repudiation:

With public keys easy to provide non-repudiation authentication

With secret keys easy to provide repudiable authentication

Public Keys: non-repudiation [hash(m)]sender

Sender includes signature (private key) on hash of message

Only someone with knowledge of private key could sign it

Email Security

Non-Repudiation:

With public keys easy to provide non-repudiation authentication
With secret keys easy to provide repudiable authentication

Public Keys: plausible deniability [{S}receiver]sender, MAC,msg

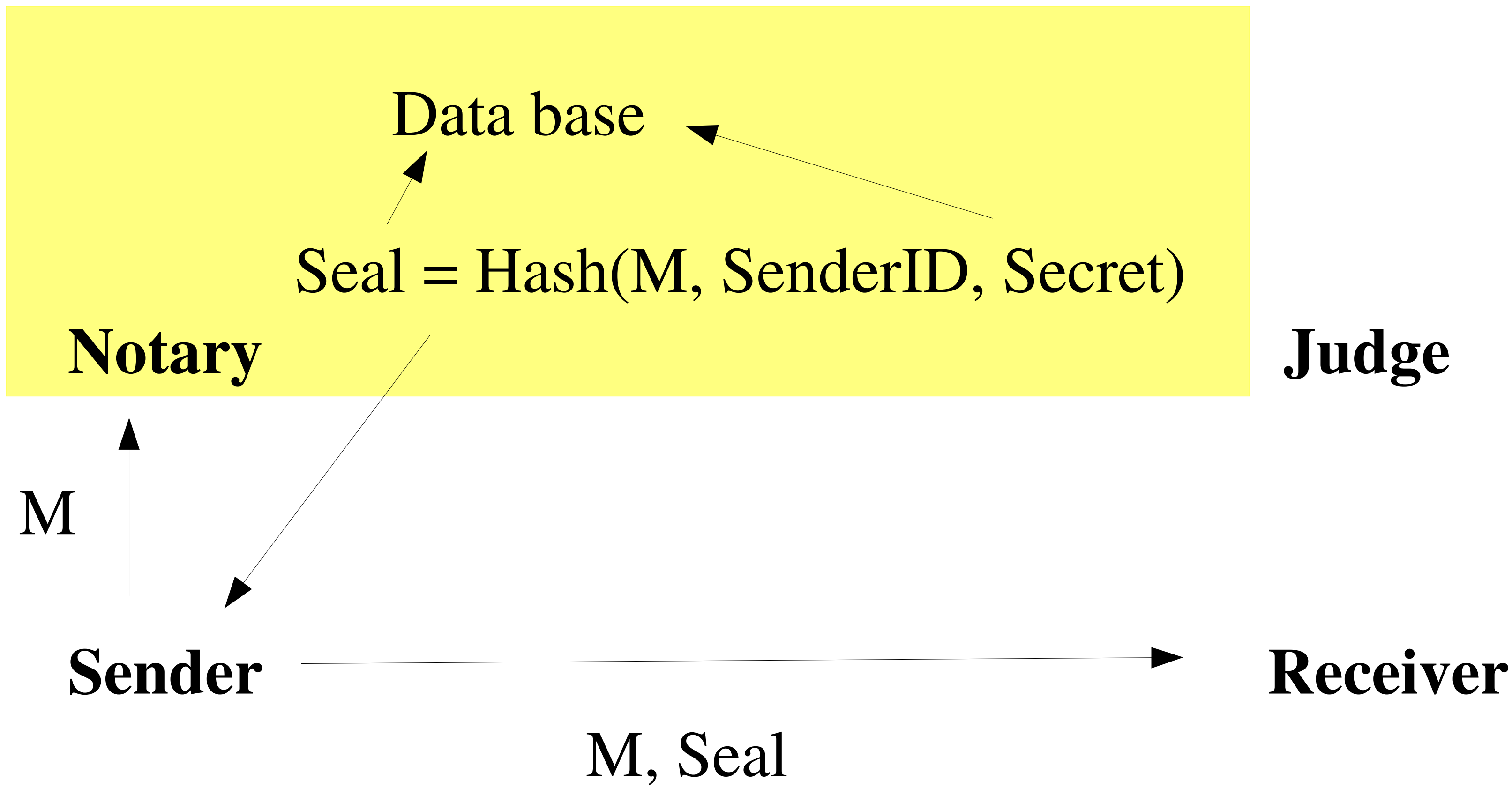
Sender chooses secret S and encrypts S with receiver's public key
Sender signs result with its private key. Then
Sender uses S to compute a MAC for the message
(e.g. CBC/DES residue)
Sender sends MAC, signed, encrypted S, message to recipient
Recipient authenticates the sender via signature and
computing the MAC of the message with S

Can prove a message was received from sender using S but
Can't prove the particular message was received because the
Recipient can use [{S}receiver]sender to MAC any message

Email Security

Non-Repudiation:

Secret Keys: non-repudiation

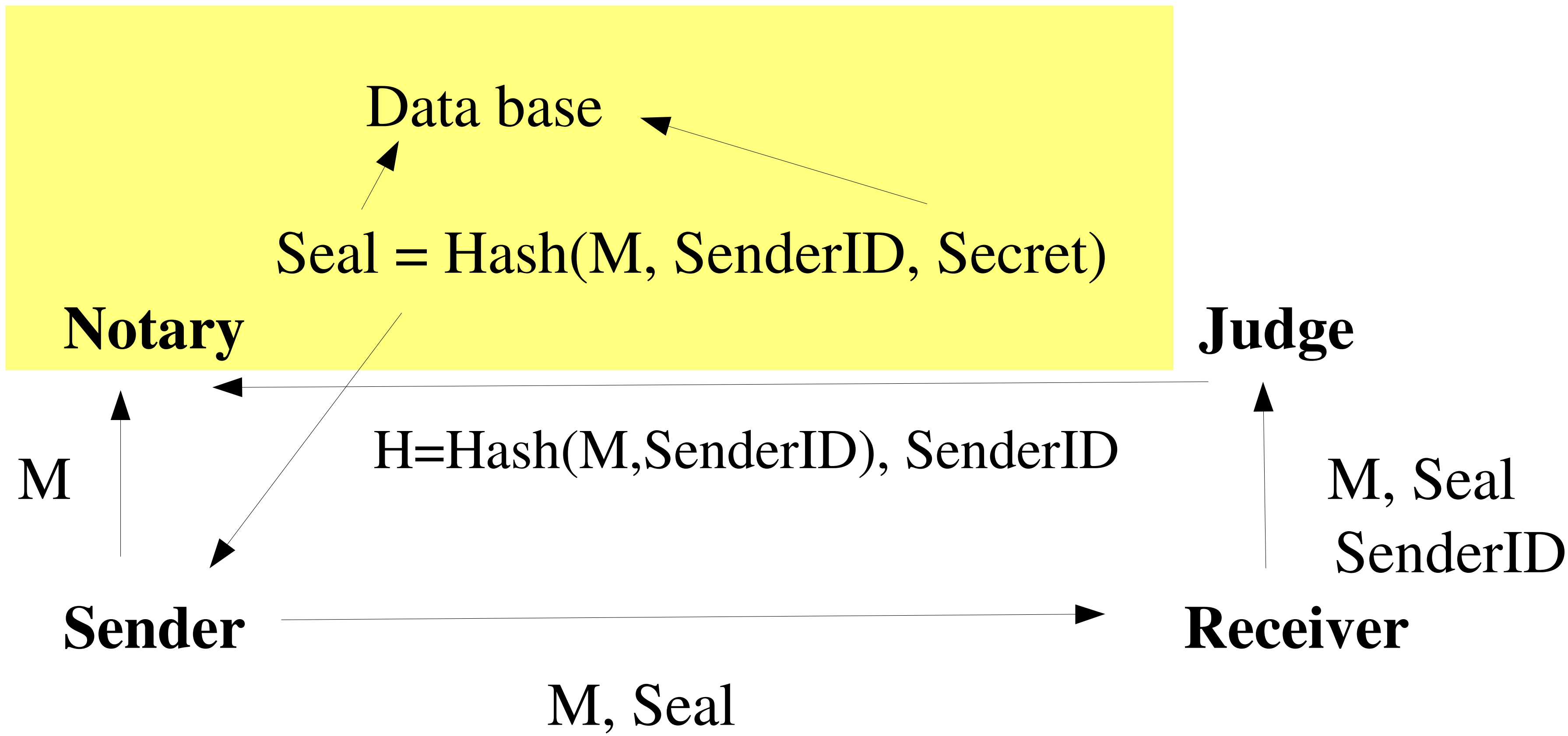


Note: notary authenticates sender in the usual way
the secret is known only to the notary
sender does not know the secret and can't tell if the seal is good

Email Security

Non-Repudiation:

Secret Keys: non-repudiation

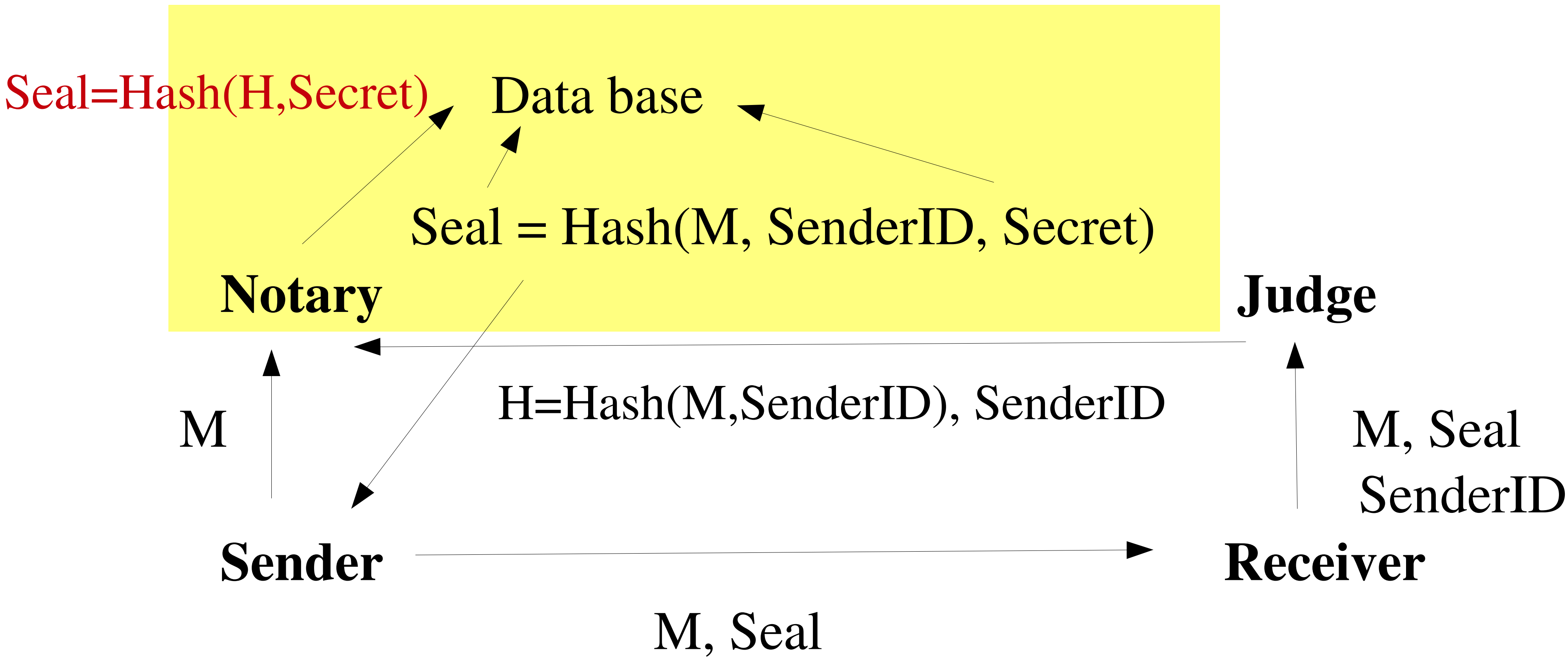


Proof: Receiver sends seal, M, sender ID to Judge
Judge takes Hash(M, SenderID), sends with SenderID to notary

Email Security

Non-Repudiation:

Secret Keys: non-repudiation



Proof: Receiver sends seal, M, sender ID to Judge
Judge takes Hash(M, SenderID), sends with SenderID to notary
Notary creates the seal and checks if it's in the database

Email Security

Non-Repudiation:

Secret Keys: plausible deniability

K_1, K_2 (two keys)

Sender ——→ **Receiver** (to read M_2)

K_2

$\{M_2\}K_2$

M_1 – benign message

M_2 – incriminating message

Keys are chosen so that

$\{M_1\}K_1 = \{M_2\}K_2$

How it works:

Recipient asks sender to decrypt $\{M_2\}K_2$

Sender decrypts with K_1 to get M_1

Email Security

Proof of Submission – public key:

Suppose mail system has a private key
It can sign a hash of the message, time of submission, intended recipient, and so on.

Sender proves message was handled by mail system by decrypting the hash with mail system public key then comparing with hash of all the above items.

Email Security

Proof of Delivery – public key:

Suppose recipient has a private key
It can sign a hash of the message + time + other things

Suppose mail system has private key
It can sign a hash of the message, etc., after recipient gets last packet and acknowledges this fact – otherwise, "proof" may be delivered but the mail is deemed "lost" by recipient
Requires cooperation of the recipient

Sender proves message was delivered by decrypting the hash with system/recipient public key then comparing with hash of all the above items.

Email Security

Message Flow Confidentiality:

Eavesdropper cannot determine that A sent message to B

Sender uses a series of intermediaries to forward the message

Each transmission is encrypted with the message of whom to forward it to next – the encrypted actual message winds up in the recipient's mailbox

Attacker can monitor the recipient but cannot determine who sent any message

Attacker can monitor the sender but cannot determine to whom a message is intended

Attacker can bribe one or more intermediaries to determine flow but must bribe them all

Email Security

Anonymity:

Harder than expected due to: Mailer automatically includes the sender's ID in a message. Further clues come from network addresses passed through

Can use a surrogate: example, mail.google.com

Anonymous mail service - provides legitimate services so as to prevent attacker or receiver from knowing exactly who the email from the service to the receiver has come

Email Security – Privacy Enhanced Mail

Email Security – Privacy Enhanced Mail

Developed as a means of adding encryption, source authentication, and integrity protection to ordinary text messages (before MS whatever attachments hit the big-time).

Email Security – Privacy Enhanced Mail

Developed as a means of adding encryption, source authentication, and integrity protection to ordinary text messages (before MS whatever attachments hit the big-time).

Smart software only at the source and destination (assume simple mail infrastructure)

Email Security – Privacy Enhanced Mail

Developed as a means of adding encryption, source authentication, and integrity protection to ordinary text messages (before MS whatever attachments hit the big-time).

Smart software only at the source and destination (assume simple mail infrastructure)

MIME (Multipurpose Internet Mail Extensions): took PEM principles into MIME structure

Email Security – Privacy Enhanced Mail

Developed as a means of adding encryption, source authentication, and integrity protection to ordinary text messages (before MS whatever attachments hit the big-time).

Smart software only at the source and destination (assume simple mail infrastructure)

MIME (Multipurpose Internet Mail Extensions): took PEM principles into MIME structure

Application layer: end-to-end, do not use any mailer tricks - PEM/MIME message will pass unchanged through any mailer

Email Security – Privacy Enhanced Mail

Developed as a means of adding encryption, source authentication, and integrity protection to ordinary text messages (before MS whatever attachments hit the big-time).

Smart software only at the source and destination (assume simple mail infrastructure)

MIME (Multipurpose Internet Mail Extensions): took PEM principles into MIME structure

Application layer: end-to-end, do not use any mailer tricks - PEM/MIME message will pass unchanged through any mailer

Supports public key and secret key technology

Email Security – Privacy Enhanced Mail

Developed as a means of adding encryption, source authentication, and integrity protection to ordinary text messages (before MS whatever attachments hit the big-time).

Smart software only at the source and destination (assume simple mail infrastructure)

MIME (Multipurpose Internet Mail Extensions): took PEM principles into MIME structure

Application layer: end-to-end, do not use any mailer tricks - PEM/MIME message will pass unchanged through any mailer

Supports public key and secret key technology

S/MIME supports encrypted messages via public key technology

Email Security - PEM

Message Structure:

Processed sections of email are marked:

```
-----BEGIN PRIVACY-ENHANCED MESSAGE-----
     ...
-----END PRIVACY-ENHANCED MESSAGE----
```

Pieces of a message are:

- Ordinary, unsecured data
- Integrity protected, unmodified data (`MIC-CLEAR`)
- Integrity protected, encoded data (`MIC-ONLY`)
- Encoded, encrypted, integrity-protected data (`ENCRYPTED`)

Email Security - PEM

Message Structure – fields (RFC 1421 - 1993):

Recipient-ID-asymmetric: cert: issuing authority, vers/exp
Originator-ID-asymmetric: cert: issuing authority, vers/exp

DEK-info: Identifies the message text encryption algorithm and mode, and carries any cryptographic parameters (e.g., IVs) used for message encryption.

Proc-type: Identifies the type of processing performed on the transmitted message (MIC-CLEAR etc)

Key-info: Follows ID field – for public key provides algorithm + DEK encrypted with recipient's public key

Mic-info: Algorithm under which MIC is computed,
Algorithm under which MIC is signed,
MIC signed with sender's private key

Email Security - PEM

Example:

```
From: franco
To: masterblaster
Subject: Blasted?
Date: Wed, 23 Nov 11 11:31:37 -0400
You are dead meat!
```

Email Security - PEM

Example: (MIC-CLEAR):

```
From: franco
To: masterblaster
Subject: Blasted?
Date: Wed, 23 Nov 11 11:31:37 -0400
You are dead meat!
    — — — — — — — — — — — — —

From: franco
To: masterblaster
Subject: Blasted?
Date: Wed, 23 Nov 11 11:31:37 -0400

----BEGIN PRIVACY-ENHANCED MESSAGE----
Proc-Type: 4,MIC-CLEAR
Content-Domain: RFC822
Originator-ID-Asymmetric: MEMxCzAJBgNVBytUEYWhUYu1jHKSjdkueyuHHGDGHHDH
 jjJHDJHEUEU[poeoidUIYDUIBYUIEYuimuiyUIEYYETFJHDGakjhjybyjxjghf,02
MIC-Info: RSA-MD5,RSA,u1OHP1RweHGfjjfkUIDlWUIhhdjkHHFGJWOK8DPNVSKjdhde
 MKGhdhyweIFSjdgtweHHDg==

You are dead meat!
----END PRIVACY-ENHANCED MESSAGE---
```

Email Security - PEM

Example: (MIC-ONLY):

```
From: franco
To: masterblaster
Subject: Blasted?
Date: Wed, 23 Nov 11 11:31:37 -0400
You are dead meat!
    — — — — — — — — — — — — —

From: franco
To: masterblaster
Subject: Blasted?
Date: Wed, 23 Nov 11 11:31:37 -0400

----BEGIN PRIVACY-ENHANCED MESSAGE----
Proc-Type: 4,MIC-ONLY
Content-Domain: RFC822
Originator-ID-Asymmetric: MEMxCzAJBgNVBytUEYWhUYu1jHKSjdkueyuHHGDGHHDH
 jjJHDJHEUEU[poeoidUIYDUIBYUIEYuimuiyUIEYYETFJHDGakjhjybyjxjghf,02
MIC-Info: RSA-MD5,RSA,u1OHP1RweHGfjjfkUIDlWUIhhdjkHHFGJWOK8DPNVSKjdhde
 MKGhdhyweIFSjdgtweHHDg==

8uYT6rw5x^ydio/+ueyTEuiieycbhejehfgeukfyuhgdfh
----END PRIVACY-ENHANCED MESSAGE---
```

Email Security - PEM

Example: (ENCRYPTED):

Encrypted integrity check
Encrypted message
Encrypted per-message key

```
From: franco
To: masterblaster
Subject: Blasted?
Date: Wed, 23 Nov 11 11:31:37 -0400
You are dead meat!
    - - - - - - - - - - - - -
From: franco
To: masterblaster
Subject: Blasted?
Date: Wed, 23 Nov 11 11:31:37 -0400

----BEGIN PRIVACY-ENHANCED MESSAGE----
Proc-Type: 4,ENCRYPTED
Content-Domain: RFC822
DEK-Info: DES-CBC,31747476B48331B1D
Originator-ID-Asymmetric: MEMxCzAJBgNVBytUEYWhUYu1jHKSjdkueyuHHGDGHHDH
 jjJHDJHEUEU[poeoidUIYDUIBYUIEYuimuiyUIEYYETFJHDGakjhjybyjxjghf,02
Key-Info: RSA,OEPURhdjIye7HEhgshjkdhfdhfhdjhgdgvdhjsgdjhgshjgjgjhgd
 kdfjoeiofkdhjfdhmdjjsf==
MIC-Info: RSA-MD5,RSA,u1OHP1RweHGfjjfkUIDlWUIhhdjkHHFGJWOK8DPNVSKjdhde
 MKGhdhyweIFSjdgtweHHDg==
Recipient-ID-Asymmetric: MEMkdjkkhadjkkadhbhjfhjo9li7jshwhh3tsttddjeyg
 uskkuuySuklMDKJMLKJEUkjstfoewjfkuuyedyuYYWTYTWytbuytWYTWYT,05
Key-Info: RSA,uiNUYUDYFgNKJdo;s[pioeidyuiYYUBDYEKJMKfjekjfnjfnjhj==

8uY*#83pjkdfuiqe89368&*#*'kldfh\pdjkd/jehjkuy3udfhas
----END PRIVACY-ENHANCED MESSAGE---
```

Email Security - PEM

Establishing Keys:

Interchange key: long-term (recipient's public key | shared secret)

Per-message key: randomly selected number

Interchange key is used to encrypt per-message key

If interchange key is shared secret it is obtained "out-of-band"

Certificates: recipient sends series of certificates to sender in unencrypted message header

Email Security - PEM

Certificate Hierarchy:

Single root CA called IPRA (Internet Policy Registration Auth.)

Certified by IRPA: PCAs (Policy Certification Authorities)

each has a written policy it will enforce when issuing certs

Email Security - PEM

Certificate Hierarchy:

Single root CA called IPRA (Internet Policy Registration Auth.)

Certified by IRPA: PCAs (Policy Certification Authorities)

each has a written policy it will enforce when issuing certs

- High Assurance (HA) super-secure, CA implemented on special hardware to be tamper resistant, managed by geeks, paranoid about handing out certs

Email Security - PEM

Certificate Hierarchy:

Single root CA called IPRA (Internet Policy Registration Auth.)

Certified by IRPA: PCAs (Policy Certification Authorities)

each has a written policy it will enforce when issuing certs

- High Assurance (HA) super-secure, CA implemented on special hardware to be tamper resistant, managed by geeks, paranoid about handing out certs
- Discretionary Assurance (DA) managed by gooks, no rules imposed on who gets issued a cert (except they verify ID)

Email Security - PEM

Certificate Hierarchy:

Single root CA called IPRA (Internet Policy Registration Auth.)

Certified by IRPA: PCAs (Policy Certification Authorities)

each has a written policy it will enforce when issuing certs

- High Assurance (HA) super-secure, CA implemented on special hardware to be tamper resistant, managed by geeks, paranoid about handing out certs
- Discretionary Assurance (DA) managed by gooks, no rules imposed on who gets issued a cert (except they verify ID)
- No Assurance (NA) can be managed by goons. Not allowed to issue two certs with same name

Email Security - PEM

Certificate Hierarchy:

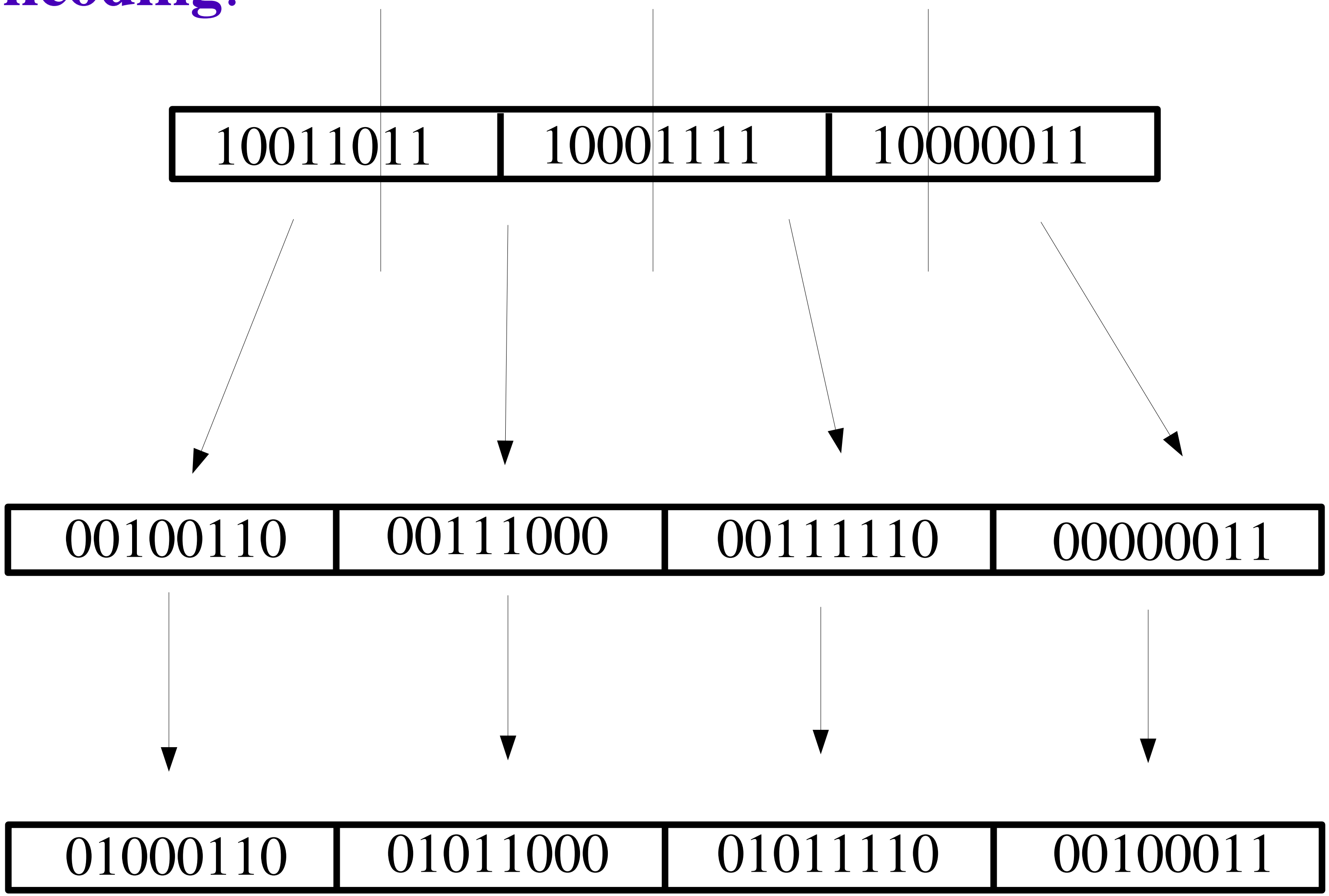
Single root CA called IPRA (Internet Policy Registration Auth.)
Certified by IRPA: PCAs (Policy Certification Authorities)
each has a written policy it will enforce when issuing certs

- High Assurance (HA) super-secure, CA implemented on special hardware to be tamper resistant, managed by geeks, paranoid about handing out certs
- Discretionary Assurance (DA) managed by gooks, no rules imposed on who gets issued a cert (except they verify ID)
- No Assurance (NA) can be managed by goons. Not allowed to issue two certs with same name

Must be single path from IRPA to any individual (no cross-certs)
Proper chain to give someone is predictable and easy to figure out.
HA below HA; DA below HA, DA; NA below HA, DA, NA

Email Security - PEM

Data Encoding:



Email Security - PEM

Unprotected Information:

Subject, to, from: mailers need to see them
No integrity protection on the to,from fields
No protection on the timestamp
But header info can be included in the text

Problem: Assume secret-key-based interchange keys,
Sender sends message to distribution list exploder
Exploder checks sender MIC and adds its own per-recipient MIC
but lacks protected means to assert MIC was checked and OKed

Email Security - PGP

Email Security - PGP

Combines best features of secret and public key cryptography.

Compresses plaintext before encryption.

Compression saves transmission time and disk space and strengthens cryptography: thwarts cryptanalysis techniques which exploit patterns found in plaintext to crack the cipher.

Creates session key: random, one-time number, and encrypts.

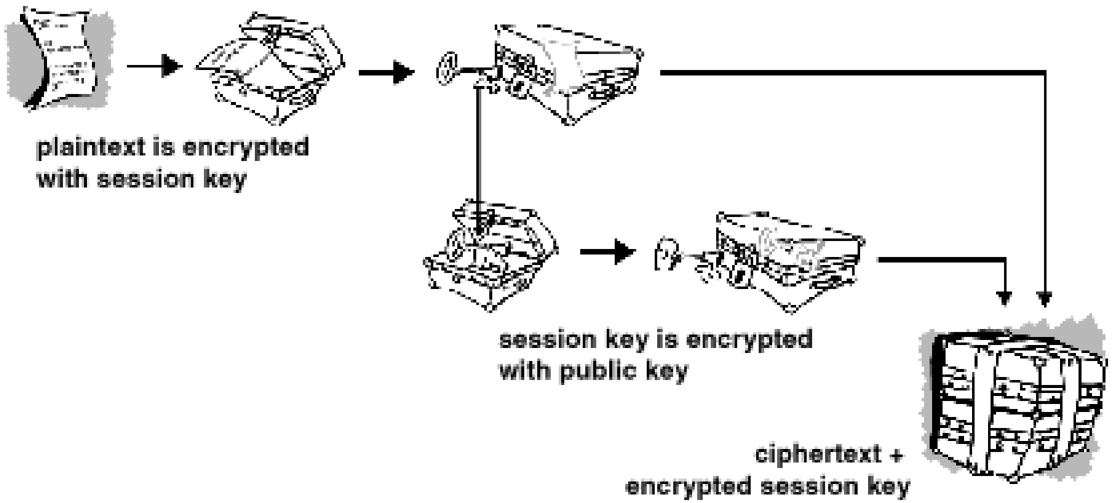
Encrypts session key with recipient's public key.

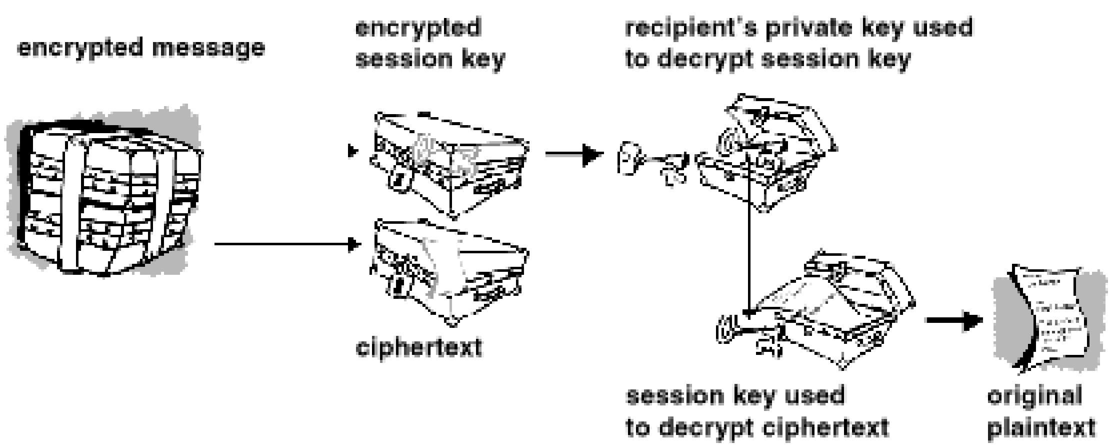
Encrypted session key and message are sent to recipient.

Recipient applies private key to encrypted session key.

Recipient uses session key to decrypt message, then decompresses.

Email Security - PGP

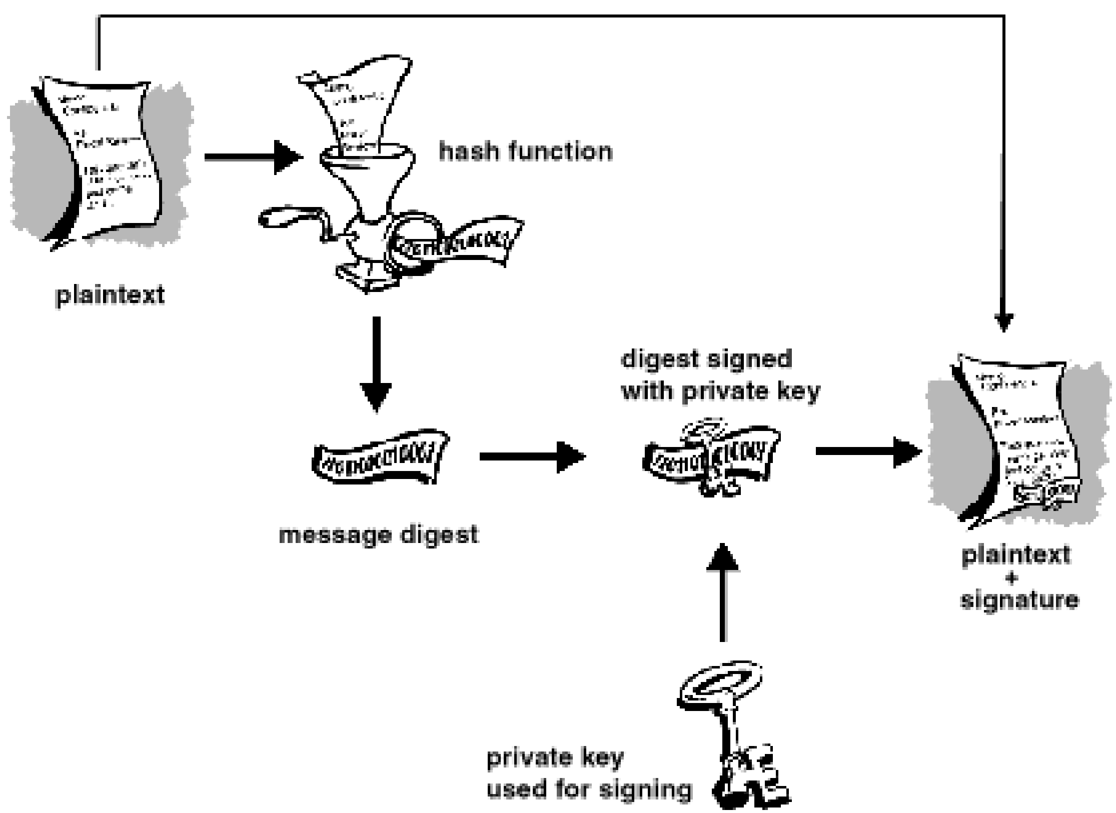




Email Security - PGP

Digital Signatures:

Authentication, integrity, non-repudiation, psbl non-encrypted.
Sender sends hash of message, signed with private key.
Recipient applies public key of sender to compute the hash of the message, then takes the hash of the message received and checks for agreement.

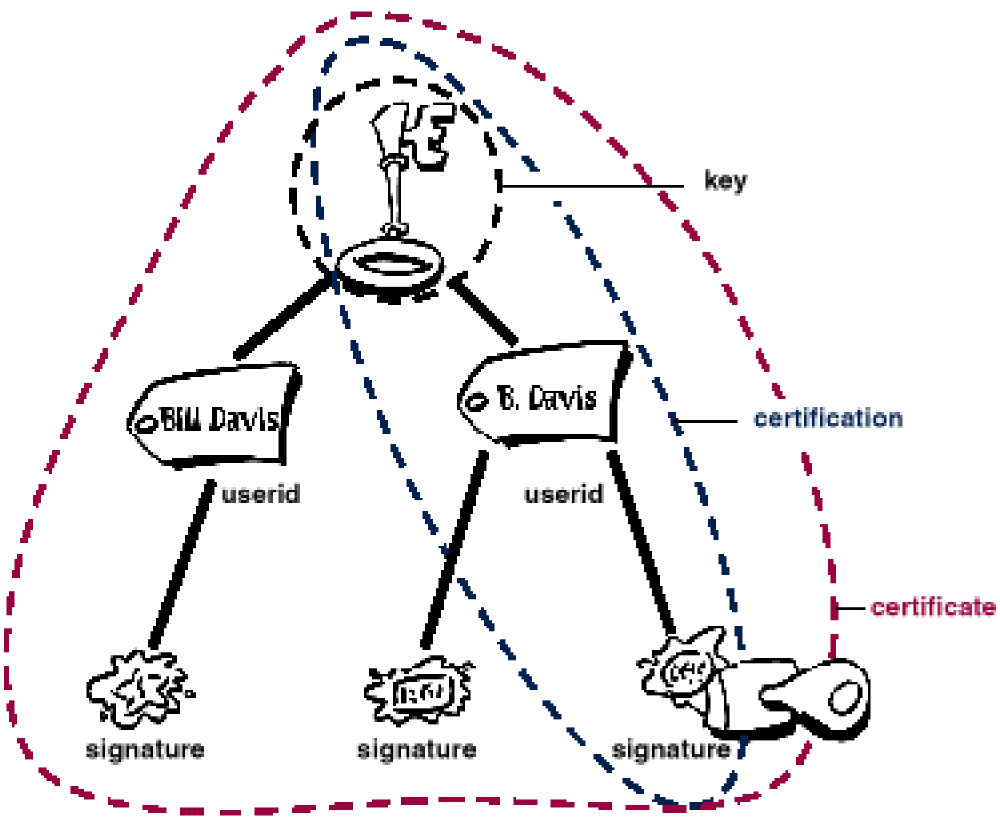


Email Security - PGP

Digital Certificates (certs):

Identity info, public key, signature of "trusted" CA

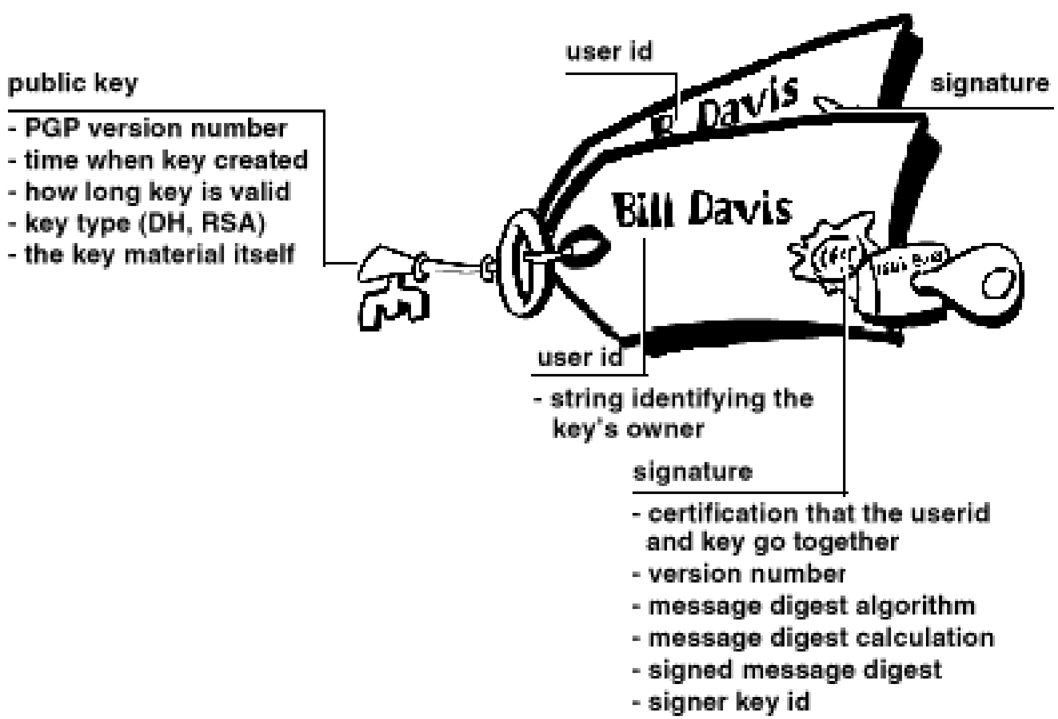
Two certificate formats: X.509 and PGP



Email Security - PGP

PGP certificate:

Version number
Certificate holder's public key
Certificate holder's identity (name, address, email, etc.)
Digital signature of the certificate holder (using private key)
Certificate validity period
Preferred symmetric encryption algorithm (3-DES,IDEA,CAST)



Email Security - PGP

X.509 certificate:

PGP certificates: anyone can play the role of validator.

X.509 certificates: the validator is always a CA or CA designate

Contains:

X.509 version number

The certificate holder's public key + algo for key + parms

The serial number of the certificate (for revocation)

The certificate holder's unique identifier (unique across internet)

distinguished name, organizational unit, organization, country

The certificate's validity period

The unique name of the certificate issuer (normally CA)

The digital signature of the issuer

The signature algorithm identifier

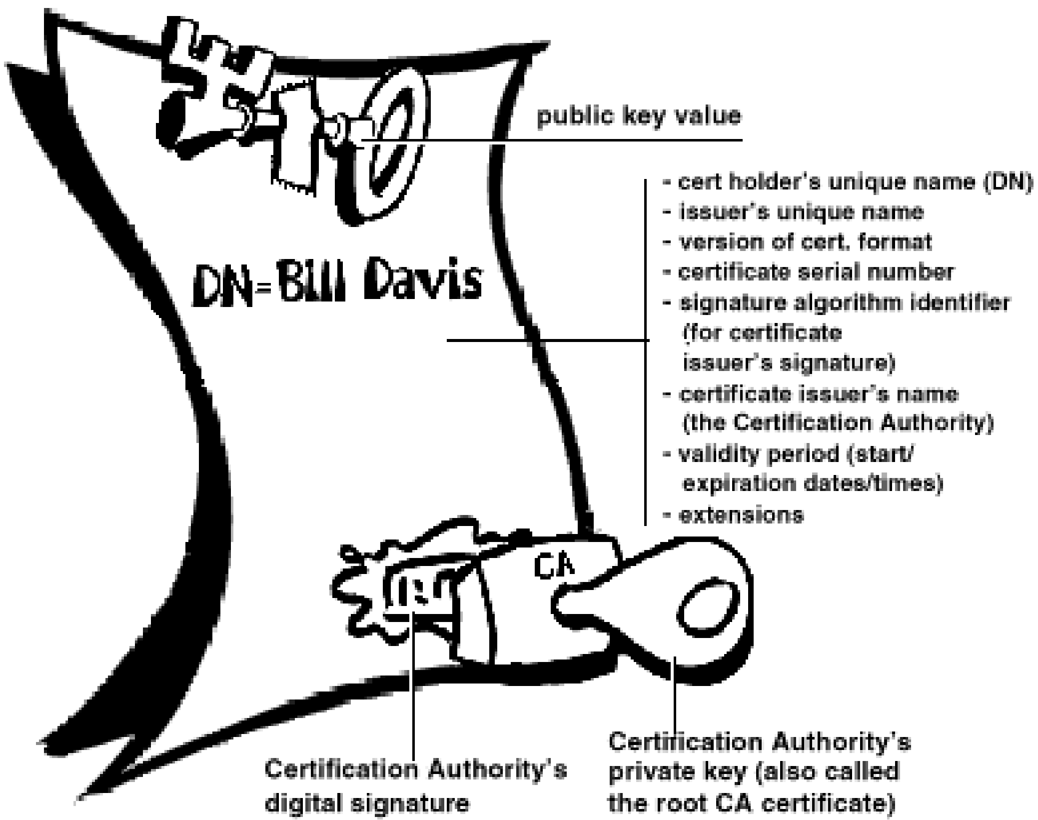
Email Security - PGP

Differences between PGP and X.509 certificates:

- User can create own PGP certificate; but user must request and be issued an X.509 certificate from a Certification Authority
  - Requestor provides public key, proof of possession of the corresponding private key, and some specific information about requestor. Send the *certificate request* signed to the CA.
- X.509 certificates natively support only a single name for the key's owner
- X.509 certificates support only a single digital signature to attest to the key's validity
- Most common use of X.509 certificates: Web Browsers

Email Security - PGP

An X.509 certificate:



Email Security - PGP

Validity:

When certificate is verified as authentic, it may be added to key ring

Can export a person's key to CA with your signature attesting validity

CA must use care in issuing certificates since it is trusted by a large body of users

Email Security - PGP

Validity:

When certificate is verified as authentic, it may be added to key ring

Can export a person's key to CA with your signature attesting validity

CA must use care in issuing certificates since it is trusted by a large body of users

Checking Validity:

Physically hand the public key to sender (not terribly practical).

Fingerprint: hash of the certificate, included in cert's properties

use phenomes to make it easy to comprehend by humans

call the key's owner and ask them to say it

or get it from their business cards

Check that certificate has not been revoked (if issued by CA)

Email Security - PGP

Establishing Trust:

Problem: When lots of users are involved - how to trust?

Meta-introducer: bestows validity on keys and ability to trust keys on others (acting as *trusted introducers*)

Root Certification Authority: same as above in X.509 lang.

Root CA uses private key of special certificate type to sign.

Certificate signed by root CA certificate is considered valid by any other certificate signed by the root CA, even if signed by other CAs as long as their certificates were signed by root

Certification chain (certification path): valid certs leading to root

Email Security - PGP

Trust Models:

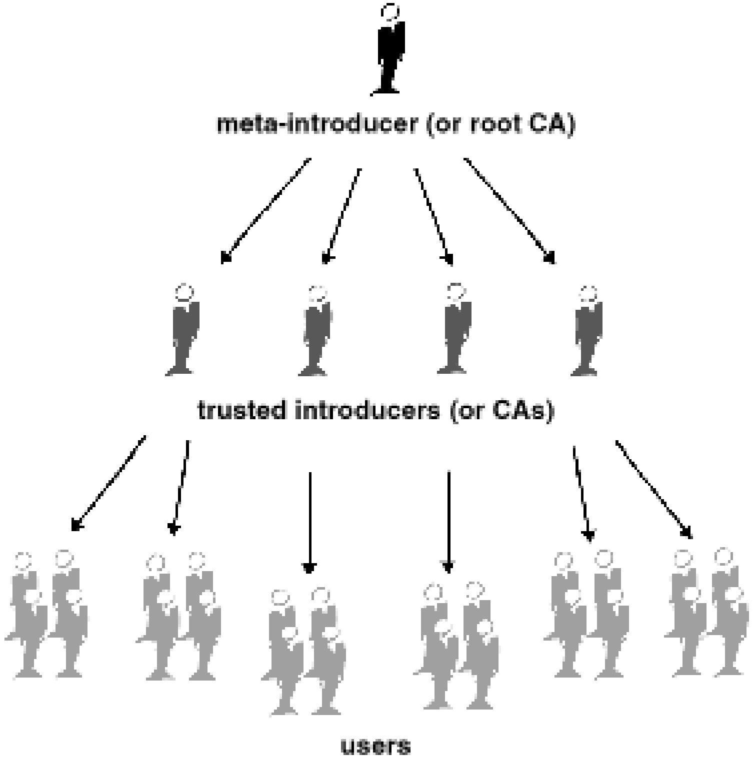
Problem: When lots of users are involved - how to find chain of certificates leading to a root CA?

Direct Trust: a key is valid because other party is known, trusted

ex: Web Browser: root keys trusted since shipped by manf. (ha)

Hierarchical: Trust tree

Web of Trust: more people signing a certificate makes it more trust-worthy. PGP model of trust. Any user can be a CA but caveat emptor. On keyring: is key valid? how much trust on user to sign other keys.



Email Security - PGP

Levels of Trust in PGP:

Implicit Trust: Trust in your own key pair. Any key signed by your implicitly trusted key is considered valid.

Complete Trust:
Marginal Trust:
Untrusted:

} Trust in someone else's key

Valid:
Marginally Valid:
Invalid:

} Validity assigned to someone else's key

Procedure for trusting a key: start with valid key, then set the level of trust you feel is deserved. A trusted key is suddenly a CA, so to speak. If trusted party signs a key, it shows up as valid on your keyring.

PGP: one completely or two marginally trusted keys to validate

Email Security - PGP

Certificate Revocation:

When a certificate expires it may still be used to decrypt, etc. things that were done during the validity period.

But a revoked certificate is to be treated with great suspicion

In PGP one can revoke signatures on a certificate or the whole certificate itself.

Only the certificate's issuer or someone whom the issuer has designated as a revoker can revoke a PGP certificate.

Reasonable because reason for revocation may be loss of passphrase for the certificate's private key

Only the certificate's issuer can revoke a X.509 certificate

In PGP use a certificate server to communicate revoked certs

Email Security - PGP

Passphrase:

Access to one's own private key is through a passphrase

A passphrase is more secure than a password because it is typically composed of multiple words and crazy symbols

A passphrase is used to encrypt the private key on a possibly public computer

Key Splitting:

Sharing private key is sometimes necessary

Then any of many people may operate on behalf of company!

Split key into three pieces so any two pieces are enough to reconstruct the key

Email Security - PGP

Vulnerability:

Message intercepted by eavesdropper

Eavedropper applies certain mathematical functions to the message corrupting it

Eavesdropper sends the corrupted message to intended recipient

Recipient decrypts and gets garbage

Recipient sends garbage message back to sender to show what happened

Eavedropper observes the message applies the inverse math function decrypting the message