Identification and Public Key
• Every node has an ID
• Every node has a public key
• The association between the ID and the key is critical
• A central question: is this the public key for node X?
  – X is the ID

PKI

• PKI consists of those components that are used to securely distribute public keys
  – Certificates
  – A repository for retrieving certificates
  – A method for revoking certificates
  – A method for evaluating certificates

A Preliminary Solution
• A node encrypts (signs) its public key with its private key \( \{ e \} d \) recipient
• The recipient can only decrypt using the public key in question
  – Thus confirm that \( e \) is the public key of the signing guy
• But who is the signing guy?
• It won’t help by adding the ID, either
  \( \{ e, Alice \} d \) recipient
  since the \( e \) and \( d \) here can actually belong to Eve!

Certificate-Based Solution
• A certificate is a token that binds an identity to a cryptographic key
• A certificate authority (CA) issues certificates

Two Types of Certificates
• Signature-less certificate
  – Merkle’s Tree Authentication Scheme
  – Such a certificate contains an authentication path
• Signed certificate
  \[ C_{Alice} = \{ e_{Alice} \parallel Alice \parallel T \} d_{Cathy} \]
Merkle’s Tree Authentication Scheme

- All <id, public key> pairs are stored in a file
- A cryptographic hash function creates a digest of the file
  - The digest is known to the public
- If any pair is changed, it will be detected
  - Since the digest will be different

Digest Algorithm

- A tree-based algorithm

Y₁ Cathy 01389234789357
Y₂ Bob 89230378597823
Y₃ Alice 72384927894027
Y₄ David 32748902378240

Y₁ h(1,1) h(1,4)
| h(2,1) h(1,2) h(3,3) h(4,4)
| Y₂ Y₃ Y₄

Signature-Less Certificate Verification

- How can Bob verify whether or not Alice’s public key is 72384927894027.
- Bob will re-compute the digest, and compare that with the publicly known value of the digest
  - If Alice’s public key is not 72384927894027, a discrepancy will be detected

Authentication Path

- Bob knows Y₃
- Bob needs to know h(4,4) and h(1,2)
- Y₃, h(4,4) and h(1,2) is the authentication path for Alice’s public key
  - They can put together and used for certifying Alice’s public key
Verifying A Signed Certificate

• Suppose Bob knows Cathy’s public key $e_{Cathy}$
• When Bob obtains $C_{Alice}$
  – Deciphers $C_{Alice}$ using $e_{Cathy}$
  – Then knows that Cathy is vouching that $e_{Alice}$ is Alice’s public key, issued at time $T$
  – If Bob trusts what Cathy believes
  – Then Bob knows $e_{Alice}$ is Alice’s public key
• But, Bob Has to Know $e_{Cathy}$!
• We focus on the signed certificate below

PKI Trust Models

• Monopoly Model
• Monopoly + Registration Authorities (RA)
• Delegated CAs
• Oligarchy
• Anarchy

Monopoly Model

• One single CA for everybody
• There is no one universally trusted organization
• Hard to reconfigure once everybody uses a single CA
• Can be remote from many principals
• Entire world relies on a single entity!

Monopoly + RAs

• Well, one can contact a local RA for a certificate
• The local RA will verify identity, securely communicates with the CA, and then the CA issues a certificate
• CA actually just rubber-stamps

Delegated CAs

• A trusted CA can issue certificates to other CAs
  – Users can then obtain certificates from one of the delegated CAs, instead of just a single trusted CA

Oligarchy

• A product comes with MULTIPLE trusted CAs
• Often used in browsers
• If one is broken, security is broken
Anarchy Model

- Everyone has its own trusted CAs
  - Probably everyone has different ones

Certificate Signature Chains

- **X.509**
- **PGP**

  - Tree-like CA hierarchy employed
    - Every node has a local CA
    - A local CA has its CA, the parent
    - The parent CA has its parent
    - And there is a root CA
    - Together, a tree of CAs!

Certificate Chains

- Cathy certifies Dan’s public key
  - Cathy <<Dan>>
- If Dan <<Bob>>, Bob<<David>>, and Alice knows Cathy’s public key,
  - then a certificate chain is formed
  - Alice can validate Bob’s public key by going through the chain

X.509

- X.509 defines certificate formats and validation in generic context
  - X.509v3 is the current version
- Format:
  - Version, serial number
  - issuer’s name, id, signature algorithm id
  - subject’s name, id, public key, validity interval
  - extensions
  - Signature

Certificate Chains

PGP Certificate Chains

- PGP (Pretty Good Privacy) provides privacy for email
  - Can also be used to sign files
  - We look at OpenPGP below
- An OpenPGP certificate is a sequence of packets
  - A public key packet followed by 0+ signature packets
  - Each packet is a record with a tag describing its purpose

Public Key Packet

- Version
- Creation time
- Validity period
- Public key algorithms (and parameters)
- Public key (of course)
Signature Packet

- Version
- Signature type
  - Also encodes a level of trust
- Creation time
- Key identifier of the signer
- Public key algorithm
- Hash algorithm
- Part of signed hash value
- Signature (of course!)

PGP Certificate Features

- PGP certificate allows multiple signatures
- Each signature has a different level of “trust”

  Different from X.509

PGP Certificate Chain Example

Alice is verifying Bob’s public key
- Ellen, Fred, Giselle, Bob <<Bob>>
- Henry, Irene, Giselle <<Giselle>>
- Ellen, Henry <<Henry>>
- Jack, Ellen <<Ellen>>

Then: Henry<<Henry>>, Henry<<Giselle>>, Giselle<<Bob>>
  Jack<<Ellen>>, Ellen<<Bob>>