L8: Public Key Infrastructure

Hui Chen, Ph.D.
Dept. of Engineering & Computer Science
Virginia State University
Petersburg, VA 23806

Acknowledgement

- Many slides are from or are revised from the slides of the author of the textbook
  - Matt Bishop, Introduction to Computer Security, Addison-Wesley Professional, October, 2004, ISBN-13: 978-0-321-24774-5. Introduction to Computer Security @ VSU's Safari Book Online subscription
  - http://nob.cs.ucdavis.edu/book/book-intro/slides/

Outline

- Public key infrastructure
- Certificate signature chains
  - X.509 certification signature chains
  - PGP certificate signature chains

Cryptographic Key Infrastructure

- Goal: *bind identity to key*
- Classical cryptographic systems: not possible as all keys are shared
  - Two parties need to *agree on a shared key* (see earlier)
- Public key cryptographic systems: *bind identity to public key*
  - Crucial as people will use key to communicate with principal whose identity is bound to key
  - Erroneous binding means no secrecy between principals
  - Assume principal identified by an acceptable name

Binding Identity to Public Key

- Bob wants to communicate with Alice
  - Alice must bind identity to her public key
- Main idea:
  - Alice *signs* her public key (e_A) with her private key (d_A)

    $\{e_A\}_{d_A}$

    - Not sufficient, because Bob would only know that whoever generated the public key also signed and cannot verify it is Alice who generated and signed the public key.
  - Using a *certificate* (Kohnfelder, 1978)

Certificates

- Create token (message) containing
  - Identity of principal (e.g., Alice)
  - Corresponding public key (e.g., E_A)
  - Timestamp (when the public key is issued, denoted as T)
  - Other information (perhaps identity of signer)
- *Signed* by a *trusted authority* (e.g., Cathy), i.e., enciphered using Cathy's private key

$$C_A = \{ e_A \,||\, \text{Alice} \,||\, T \}\ d_C$$

- C_A is a certificate, a token that binds an identity to a cryptographic key

Using Certificate

- Bob wants to communicate with Alice
  - Bob obtains Alice's certificate

    $C_A = \{ e_A \| \text{Alice} \| T \}_{d_C}$
  - If he knows Cathy's public key, he can decipher the certificate
    - When was certificate issued?
    - Is the principal Alice?
  - Now Bob has Alice's public key

Problem of Using Certificate

- Problem: Bob now needs issuer's, i.e., Cathy's public key to validate certificate
- The certificate approach pushes the problem "up" a level
  - Solution: construct a tree-like hierarchy
    - Using certificate signature chains
    - Using Merkle trees (Merkle, 1979): as further reading

Certificate Signature Chains

- Issuer creates certificate
  - Generate hash of certificate
  - Encipher hash with issuer's private key
- Anyone can validate the certificate
  - Obtain issuer's public key
  - Decipher enciphered hash
  - Re-compute hash from certificate and compare
- Problem: how to obtain issuer's public key

Two Approaches

- Problem of Certificate Signature Chains: *getting issuer's public key*
- Two approaches
  - To construct a tree-like hierarchy with the public key of the root known out of band
    - e.g., X.509 certificate signature chains
  - To allow an arbitrary arrangement of certifiers and rely on each individual's knowledge of the certifiers
    - e.g., PGP certificate signature chains

X.509 Certificate Signature Chains

- ITU-T standard
- A public key infrastructure (PKI)

Certificate Authority (CA)

- Entity that issues certificates
- Multiple CAs exist in X.509

X.509 Certificate

- Some certificate components in X.509v3 certificate:
  - Version
  - Serial number
  - Signature algorithm identifier: hash algorithm
  - Issuer’s name: uniquely identifies issuer
  - Interval of validity
  - Subject’s name: uniquely identifies subject
  - Subject’s public key
  - Signature: enciphered hash
- Issued and signed using a CA’s private key

X.509 Certificate Validation

- Obtain issuer's public key
  - The one for the particular signature algorithm
- Decipher signature using the public key
  - Yields hash of certificate
- Re-compute hash from certificate and compare the two
  - If they differ, there is a problem
- Check interval of validity
  - which confirms that certificate is current

Multiple Certificate Issuers

- An example scenario
  - Alice wants to communicate with Bob
    - Alice’s local CA is Cathy and Alice has a certificate from Cathy
    - Bob’s local CA is Dan and Bob has a certificate from Dan
  - Validation problem caused by multiple CAs, i.e., certificate issuers
    - Alice and Bob need to validate each other’s certificates
      - Alice’s CA is Cathy; Bob’s CA is Don; how can Alice validate Bob’s certificate?
  - Solution
    - Have Cathy and Don cross-certify
      - Each issuers issues certificate for the other issuer

Validation and Cross-Certifying

- X<<Y>>: certificate that X generated for subject Y
- Certificates
  - Cathy<<Alice>>
  - Dan<<Bob>
  - Cathy<<Dan>>
  - Dan<<Cathy>>
- Alice validates Bob’s certificate
  - Alice obtains Cathy<<Dan>>
  - Alice uses (known) public key of Cathy to validate Cathy<<Dan>>
  - Alice uses Cathy<<Dan>> to validate Dan<<Bob>>

Cross-Verifying and Certificate Chain

- Cross-Verifying: Two CAs are cross-verified if each has issued a certificate for the other
- Signature Chain
  - Cathy is Alice’s local CA and Alice has Cathy’s public key. Alice can obtain certificate Cathy<<Dan>> and form the signature chain
    - Cathy<<Dan>> Dan<<Bob>
  - Similar argument can be made for Bob and Dan (Bob’s CA)
    - Dan<<Cathy>> Cathy<<Alice>>

Certificate Chain

- Signature chains can be of arbitrary length
- Each certificate can be validated by the one before it in the chain
- X.509 suggests organize CAs into a hierarchy to minimize the lengths of certificate signature chains
- Certificates can be revoked, or canceled
  - A list of such certificates enables a user to detect and reject invalidated certificates

Lab L8-1

- Experimenting X.509 PKI with OpenSSL

PGP Certificate Signature Chains

- PGP: Pretty Good Privacy
- Widely used to provide privacy for e-mail through the Internet and to sign files digitally
- PGP uses a certificate-based key management infrastructure for users' public keys
- OpenPGP
  - http://openpgp.org

OpenPGP Certificate

- OpenPGP certificates are structured into packets
- Packet: a record with a tag describing its purpose
- A certificate consists of
  - One public key packet
  - Zero or more signature packets

OpenPGP Public Key Packet

- Version
  - 3 or 4
  - 3 compatible with all versions of PGP
  - 4 not compatible with older versions of PGP)
- Creation time
- Validity period (not present in version 3)
- Public key algorithm, associated parameters
- Public key

OpenPGP Signature Packet

- Version 3 signature packet
  - Version (3)
  - Signature type (level of trust)
  - Creation time (when next fields hashed)
  - Signer's key identifier (identifies key to encipher hash)
  - Public key algorithm (used to encipher hash)
  - Hash algorithm
  - Part of signed hash (used for quick check)
  - Signature (enciphered hash)
- Version 4 packet more complex

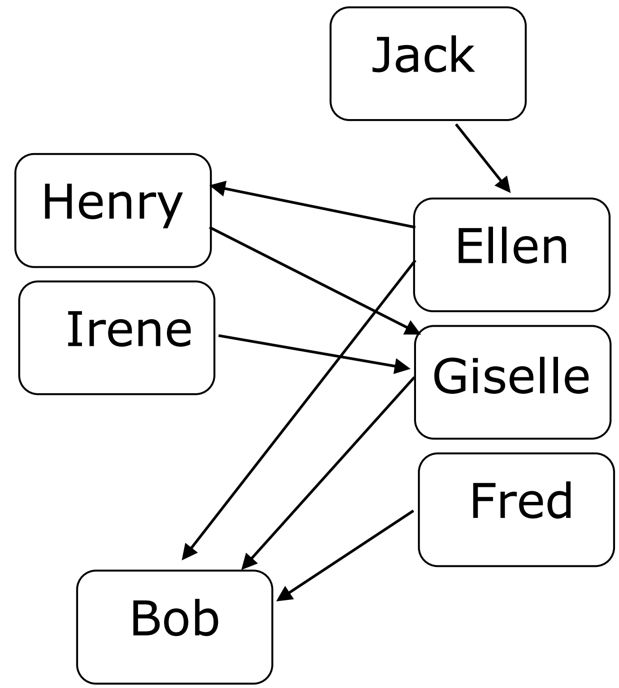
Signing

- Single certificate may have multiple signatures
  - Ellen,Fred,Giselle,Bob<<Bob>>
  - Certificate that Ellen, Fred, Giselle, and Bob generated for Bob
    - The certificate is signed by Ellen, Fred, Giselle, and Bob
- Notion of “trust” embedded in each signature
  - Range from “untrusted” to “ultimate trust”
  - Signer defines meaning of trust level (no standards!)
- All version 4 keys signed by subject
  - Called “self-signing”

Validating Certificates

- Scenario: Alice wants to communicate with Bob
- Alice obtains Bob's PGP certificate and needs to validate it
  - Ellen,Fred,Giselle,Bob<<Bob>>
  - Alice does not know Fred, Giselle, or Ellen
- Alice gets Giselle's PGP certificate
  - Henry,Irene,Giselle<<Giselle>>
  - Alice knows Henry slightly
- Alice gets Henry's PGP certificate
  - Ellen,Henry<<Henry>>
  - Use it to verifies Giselle's certificate
  - But Henry's signature is at "casual" level of trust
- Alice gets Ellen's PGP certificate
  - Jack,Ellen<<Ellen>>
  - Knows Jack *well*, so uses his cert to validate Ellen's, then use Ellens to validate Bob's

- Arrows show signatures
- Self signatures not shown



Certificate Chains

- In the above example, Alice followed two signature chains
  - Henry<<Henry>> Henry<<Giselle>> Giselle<<Bob>>
  - Jack<<Ellen>> Ellen<<Bob>>

Trust in X.509 and PGP

- X.509 certificates include an element of trust, but the trust is not indicated in the certificate
- PGP certificate indicate the level of trust, but the same level of trust may have different meanings to different signers

Lab 3

- Experimenting OpenPGP with GnuPG (GPG)

Summary

- Public key infrastructure
- Certificate signature chains
  - X.509 certification signature chains
  - PGP certificate signature chains
- Future reading
  - Merkel tree