Lecture 10: Public-Key Distribution

Stephen Huang

Contents

- Symmetric Key distribution

1. Key Distribution using Public-Key Cryptography
2. Distribution of Public Keys

Cryptographic key distribution involves cryptographic, protocol, and management considerations. This lecture gives the reader a touch on the issues involved and a broad survey of key management and distribution aspects.

Continue from the last lecture.

Reminder: Key Distribution

- Symmetric-key cryptography: efficient, but requires frequently setting up fresh secret keys
  - distribute short-term session keys using long-term master keys

- How to set up master keys?
  - deliver manually or using some secure channel (difficult or impossible)
  - use public-key cryptography to set up secret keys, communication parties do not need to have a shared secret

1. Key Distribution: Public-Key Cryptography

- Diffie-Hellman Key Exchange
  - Designed by Whitfield Diffie and Martin Hellman in 1976
    - first published public-key algorithm/protocol
  - Very widely used
    - example: SSL/TLS, SSH
  - ElGamal (and similar crypto primitives) are based on the idea of Diffie-Hellman.
  - Discussed in Lecture 6.

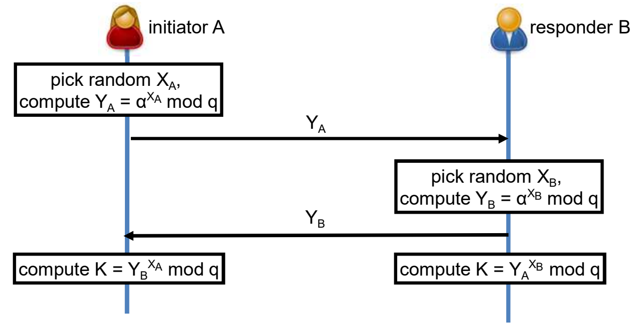
Diffie-Hellman Key Exchange

- Security depends on the hardness of finding discrete logarithms: given α, y, and q, find an x that satisfies

$$y = \alpha^{x} \bmod q.$$

  - widely believed to be a computationally hard problem
- If q is prime, then α is a primitive root if $\alpha, \alpha^2, \alpha^3, \ldots, \alpha^{(q-1)}$ are all different modulo q.

Diffie-Hellman Protocol

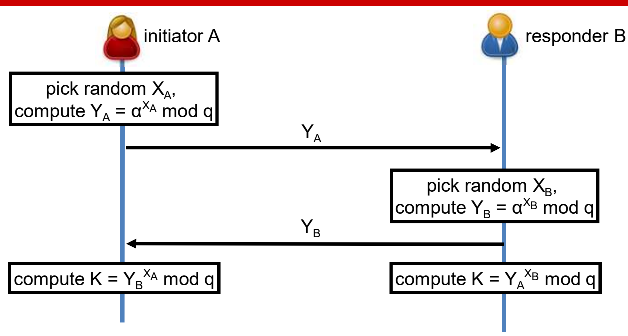


- Public: let q be a large prime number, and α be a primitive root of q

$$K = Y_B^{X_A} = (\alpha^{X_B})^{X_A} = \alpha^{X_A X_B} = (\alpha^{X_A})^{X_B} = Y_A^{X_B} = K \bmod q$$

- Secure against eavesdropping since computing X_A or X_B is hard
- Elliptic Curve D-H (ECDH): same principle, more efficient

Public-Key Encryption

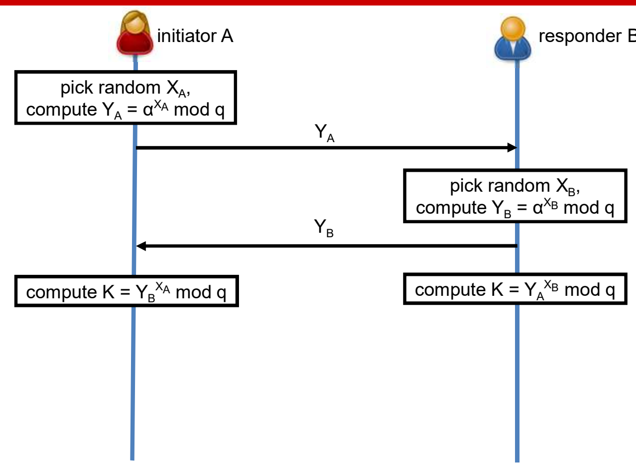


- Public: let q be a large prime number, and α be a primitive root of q

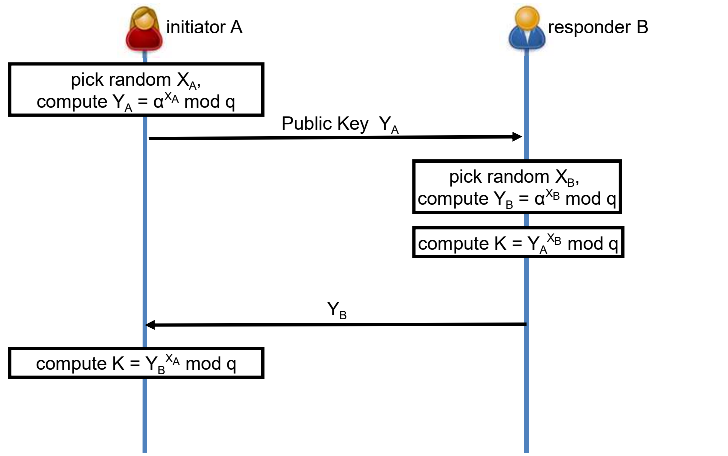
$$K = Y_B^{X_A} = (\alpha^{X_B})^{X_A} = \alpha^{X_A X_B} = (\alpha^{X_A})^{X_B} = Y_A^{X_B} = K \bmod q$$

- Secure against eavesdropping since computing X_A or X_B is hard
- Elliptic Curve D-H (ECDH): same principle, more efficient

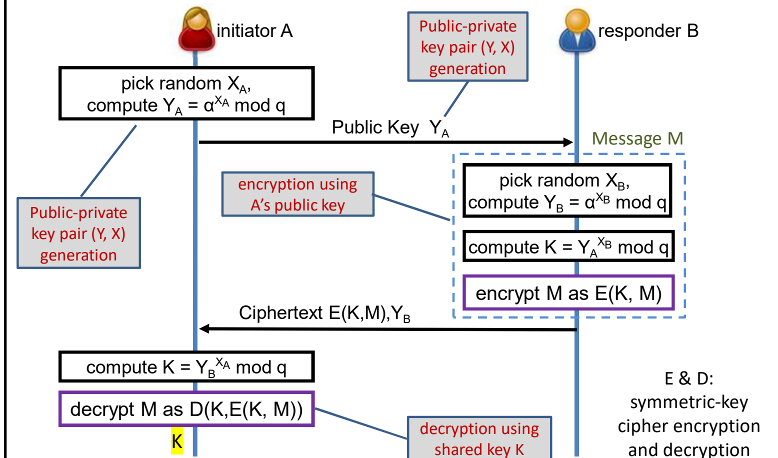
Public-Key Distribution Protocol



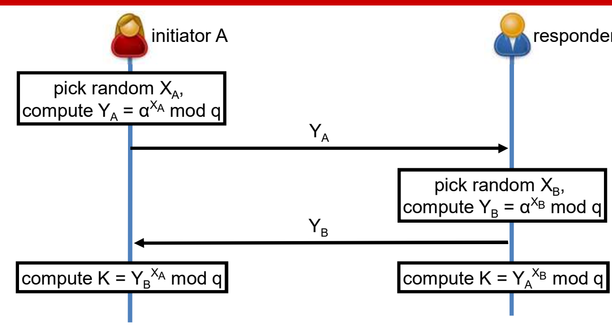
Public-Key Distribution Protocol



Public-Key Distribution of Symmetric Key



Diffie-Hellman Protocol

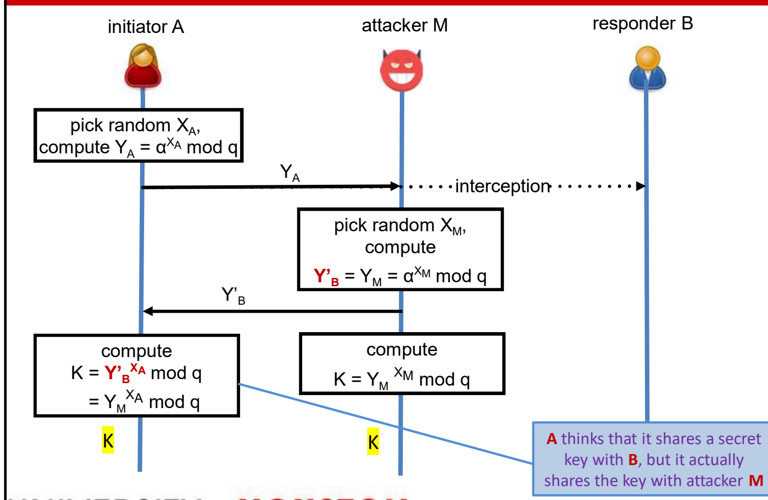


- Public: let q be a large prime number, and α be a primitive root of q

$$K = Y_B^{X_A} = (\alpha^{X_B})^{X_A} = \alpha^{X_A X_B} = (\alpha^{X_A})^{X_B} = Y_A^{X_B} = K \bmod q$$

- Secure against eavesdropping since computing X_A or X_B is hard
- Elliptic Curve D-H (ECDH): same principle, more efficient

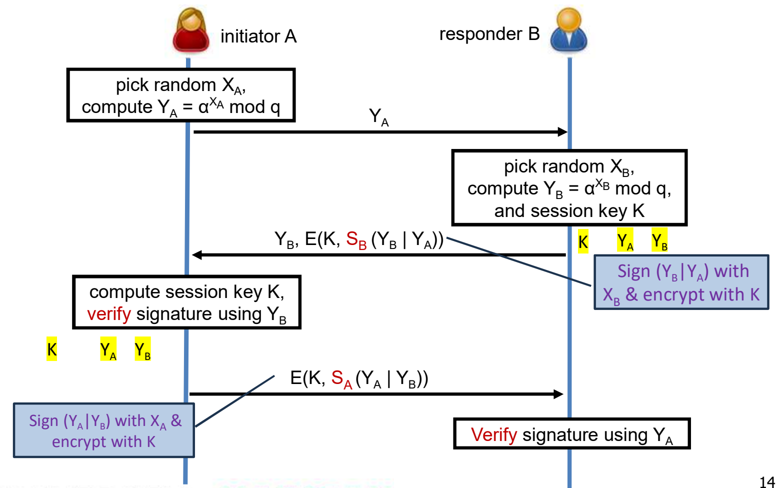
D-H: Man-in-the-Middle Attack



Station-to-Station Protocol

- The Station-to-Station (STS) protocol is a cryptographic key agreement scheme.
- The protocol is based on classic Diffie-Hellman and provides mutual key and entity authentication.
- Unlike the classic Diffie–Hellman, this protocol assumes that the parties have signature keys, which are used to sign messages, thereby providing security against man-in-the-middle attacks.
- Assume that A knows B's public key PU_B and B knows A's public key PU_A
- Digital signature: S(PR, M) is message M signed using private key PR

Basic Station-to-Station Protocol

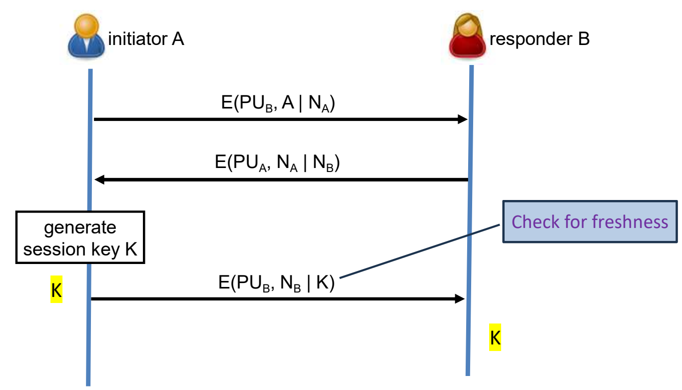


Basic STS

1. A generates a random number *x* and sends g^x to B.
2. B generates a random number *y* and computes g^y.
3. B computes the shared secret key $K = (g^x)^y$.
4. B concatenates the exponentials (g^y, g^x), signs it using B's private key, and then encrypts the signature with *K*. B sends the ciphertext along with his own exponential g^y to A.
5. A computes the shared secret key $K = (g^y)^x$.
6. A decrypts and verifies B's signature using B's public key.
7. A concatenates (g^x, g^y), signs them using A's private key, and then encrypts the signature with *K*. A sends the ciphertext to B.
8. B decrypts and verifies A's signature using her public key.

Key Distribution Using Public-Key Encryption

Assumes that communication parties know each other's public keys.



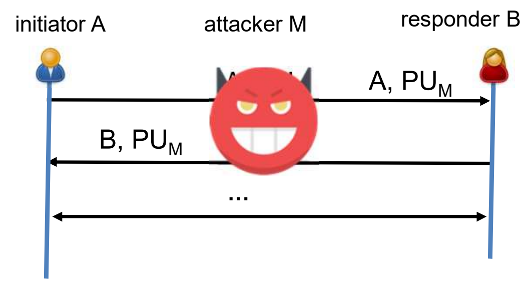
2. Distribution of Public Keys

- Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:
  a) Public Announcement
  b) Publicly Available Directory
  c) Public-key Authority
  d) Public-key Certificates

Distributing Public Keys

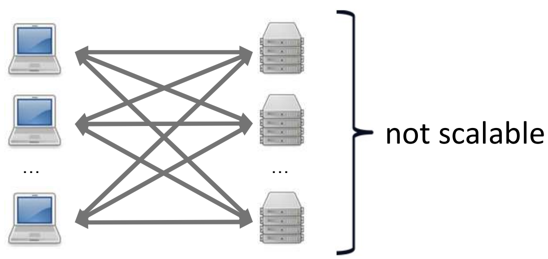
How to distribute public keys?

- Naïve public-key distribution



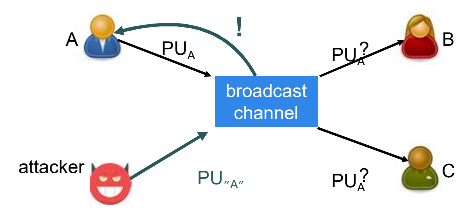
Distributing Public Keys

- Manual public-key distribution (*e.g.*, physical transfer) for all pairs.



(a) Public Announcement

- Announce the public key on some broadcast channel
  - announce on some public forum, such as an e-mail list or social media
  - public keys used for PGP (Pretty Good Privacy) are often distributed in this way
- *Weakness*: anyone can forge such an announcement
  - impersonated entity can detect the attack and notify others
  - but until then, the forger can impersonate the victim



Public-Key Publication

Figure 14.10 Uncontrolled Public-Key Distribution

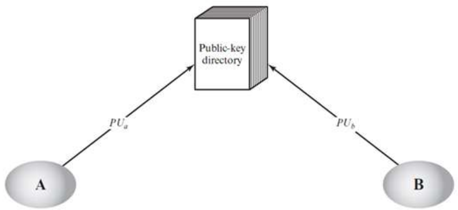
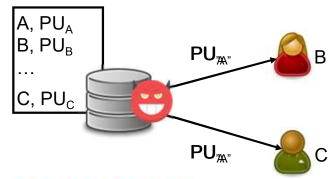


Figure 14.11 Public-Key Publication

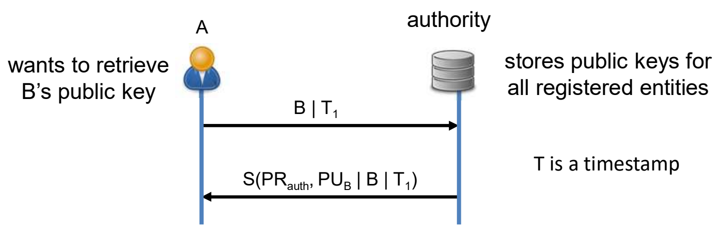
(b) Publicly Available Directory

- A trusted entity (e.g., one of the organizations governing the Internet) maintains a publicly available dynamic directory of public keys
  - authorized participants can register their (name, public key) pairs in the directory
- Weakness: a trusted directory is a single point of failure
  - directory itself may be compromised
  - attacker may impersonate the directory



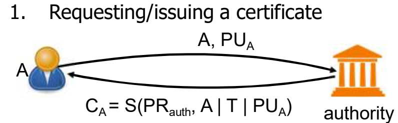
(c) Public-Key Authority

- Each participant knows the public key PU_{auth} of an authority that maintains a publicly available directory of public keys
- Weaknesses
  - authority is a single point of failure
  - authority must be online



(d) Public-Key Certificates

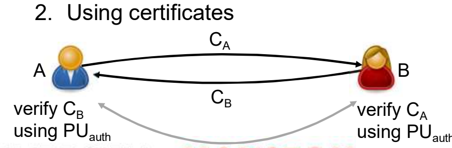
- Certificates
  - enable participants to exchange keys without contacting the authority
  - in a way that is as secure as if the keys were obtained from the authority
- Two phases:
  1. Requesting/issuing a certificate



through a secure authenticated channel
(for example, in person)

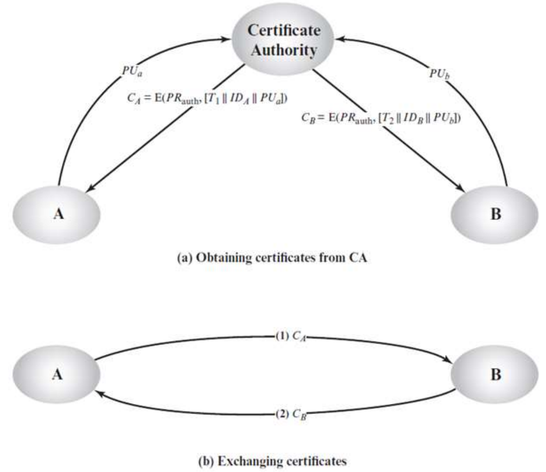
T is a timestamp

  2. Using certificates



proves the authenticity of public keys without using the authority

Another View



(a) Obtaining certificates from CA

(b) Exchanging certificates

Public-Key Certificates

- Participant A applied to the Certificate Authority (Auth) with A's ID and Public Key (PU_A).
- The authority issues a certificate:
  $$C_A = E(PR_{auth}, [T||A||PU_A])$$
- Participant A passes the certificate to Participant B, who reads and verifies the certificate:
  $$D(PU_{auth}, C_A)$$
  $$= D(PU_{auth}, E(PR_{auth}, [T||A||PU_A]))$$
  $$= T||A||PU_A$$
- Since B used the authority's public key, the certificate must come from the certification authority.

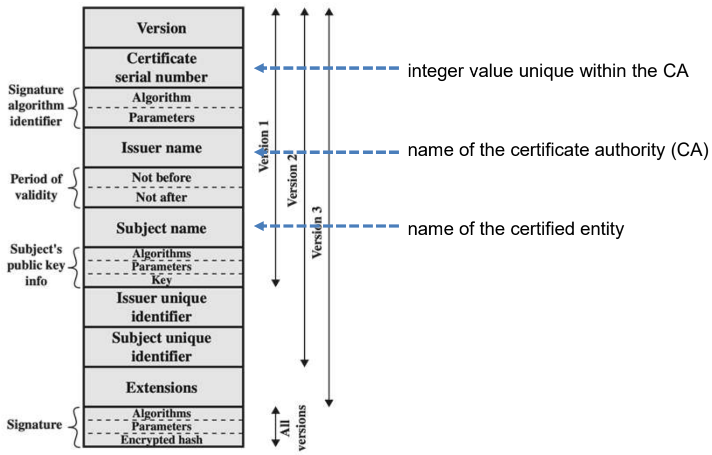
Requirements

- Certificate = (owner's name, public key, timestamp) signed by the authority
- Requirements
  1. any participant can read a certificate to determine the name and public key of the certificate's owner
  2. only the certificate authority can create certificates
  3. any participant can verify that a certificate originated from the authority
  4. any participant can verify that a certificate is recent
- Problem: compromised private key
  - if an attacker has learned the private key PR_A of an entity A, then the attacker can use A's certificate C_A to impersonate A

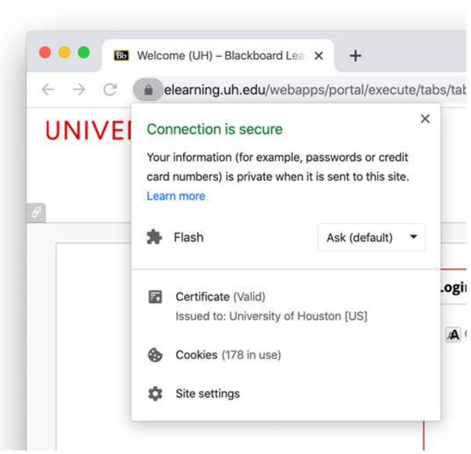
X.509 Certificates

- X.509 standard
  - ITU-T standard for public-key certificates and related functions
  - first published in 1988, updated multiple times
  - does not dictate specific algorithms (e.g., for signature)
- Very widely used
  - SSL/TLS
  - IPSec
  - S/MIME
- The International Telecommunication Union (ITU) Telecommunication Standardization Sector (ITU-T) is a UN-sponsored agency.

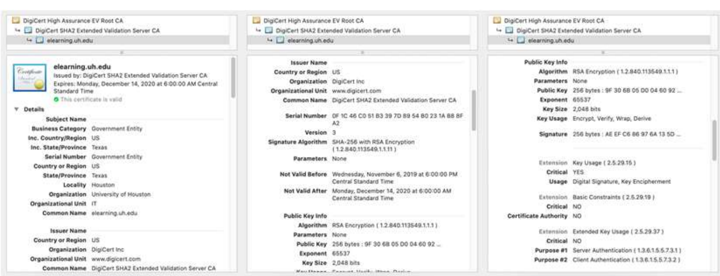
X.509 Format



X.509 Certificate in Practice



X.509 Certificate in Practice



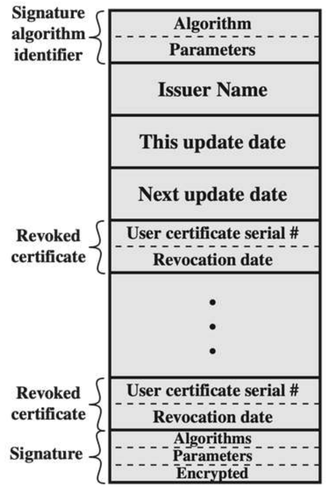
Certificate Authorities in Practice

- Operating systems and web browsers typically come with a list of trusted certificate authorities (e.g., Microsoft Root Certificate Program, Mozilla Root Certificate Program)
  - these CAs are trusted by the developers (e.g., they follow security standards)
  - users can add to or remove CAs from this list
- Common types of CAs
  - commercial: charges a fee for issuing a certificate
  - governmental
  - private non-profit (e.g., CAcert)

CA	Market share (2021 September)
IdenTrust	54%
DigiCert	19%
Sectigo	17%
...	

Certificate Revocation (the Ugly Part)

- Revocation is necessary if
  - private key of the owner is compromised
  - owner is no longer certified
  - authority's certificate is compromised
- Each CA maintains a Certificate Revocation List (CRL)
  - signed and published by the CA
  - when checking the validity of a certificate, one must check if it is on the CRL
- For efficiency, clients cache the list → revoked certificates may be accepted until the cache expires



Conclusion

- Distributing symmetric (i.e., secret) keys
  - decentralized → not scalable
  - centralized (e.g., extended Needham-Schroeder, Kerberos)
  - public-key cryptography (e.g., Diffie-Hellman key exchange)
- Distributing public keys
  - public announcement, public-key authority
  - public-key certificates
    - requesting on a secure channel (e.g., in person)
    - certificate proves the authenticity of a public key
    - revocation lists signed and published by the CA

Next Topic

- Public Key Distribution
- WiFi Security
- WPA2 and IPSec