CSC 774 Advanced Network Security

Topic 2. Network Security Primitives

Outline

- Absolute basics
  - Encryption/Decryption; Digital signatures; D-H key exchange; Hash functions; Pseudo random functions; traditional key distribution techniques
  - Review of CSC 574
- Primitives based on hash functions
  - One-way hash chain, Merkle hash tree, client puzzles, Bloom filters
- Secret sharing
- Rabin’s information dispersal algorithms
- Rabin’s fingerprinting algorithm
- Secret handshake
- ID-based cryptography

CSC 774 Advanced Network Security

Topic 2.1 Absolute Basics
Encryption/Decryption

- Plaintext: a message in its original form
- Ciphertext: a message in the transformed, unrecognized form
- Encryption: the process that transforms a plaintext into a ciphertext
- Decryption: the process that transforms a ciphertext to the corresponding plaintext
- Key: the value used to control encryption/decryption.

Cryptanalysis

- Ciphertext only:
  - Analyze only with the ciphertext
  - Example: Exhaustive search until “recognizable plaintext”
  - Smarter ways available
- Known plaintext:
  - Secret may be revealed (by spy, time), thus <ciphertext, plaintext> pair is obtained
  - Great for mono-alphabetic ciphers

Cryptanalysis (Cont’d)

- Chosen plaintext:
  - Choose text, get encrypted
  - Useful if limited set of messages
- Chosen ciphertext:
  - Choose ciphertext
  - Get feedback from decryption, etc.
Secret Key Cryptography

plaintext → encryption → ciphertext → decryption → plaintext

• Same key is used for encryption and decryption
• Also known as
  – Symmetric cryptography
  – Conventional cryptography

Public Key Cryptography

plaintext → encryption → ciphertext → decryption → plaintext

Public key → Private key

• Invented/published in 1975
• A public/private key pair is used
  – Public key can be publicly known
  – Private key is kept secret by the owner of the key
• Much slower than secret key cryptography
• Also known as
  – Asymmetric cryptography

Public Key Cryptography (Cont’d)

message → Sign → Digital signature → Verify → Yes/No

Private key → Public key

• Another mode: digital signature
  – Only the party with the private key can create a digital signature.
  – The digital signature is verifiable by anyone who knows the public key.
  – The signer cannot deny that he/she has done so.
Public Key Cryptography (Cont’d)

• Example algorithms
  – RSA
  – DSA
  – Diffie-Hellman

Digital Signature Algorithm (DSA)

• Generate public parameters
  – p (512 to 1024 bit prime)
  – q (160 bit prime): \( q|p-1 \)
  – \( g = h^{(p-1)/q} \mod p \), where \( 1 < h < (p-1) \) such that \( g > 1 \).
  – \( g \) is of order \( q \mod p \).
• User’s private key \( x \)
  – Random integer with \( 0 < x < q \)
• User’s public key \( y \)
  – \( y = g^x \mod p \)
• User’s per message secret number
  – \( k = \) random integer with \( 0 < k < q \).

DSA (Cont’d)

• Signing
  – \( r = (g^k \mod p) \mod q \)
  – \( s = [k^{-1}(H(M)+xr)] \mod q \)
  – Signature = \( (r, s) \)
• Verifying
  – \( M', r', s' = \) received versions of \( M, r, s \).
  – \( w = (s')^{-1} \mod q \)
  – \( u_1 = [H(M')w] \mod q \)
  – \( u_2 = (r')w \mod q \)
  – \( v = (g^{ur}s^w) \mod p \mod q \)
  – \( if \ v = r' \ then \ the \ signature \ is \ verified \)
Hash Algorithms

- Message of arbitrary length
- Hash \( H \)
- A fixed-length short message

- Also known as
  - Message digests
  - One-way transformations
  - One-way functions
  - Hash functions

- Length of \( H(m) \) much shorter than length of \( m \)
- Usually fixed lengths: 128 or 160 bits

Hash Algorithms (Cont’d)

- Desirable properties of hash functions
  - Performance: Easy to compute \( H(m) \)
  - One-way property: Given \( H(m) \) but not \( m \), it is computationally infeasible to find \( m \)
  - Weak collision free: Given \( H(m) \), it is computationally infeasible to find \( m' \) such that \( H(m') = H(m) \)
  - Strong collision free: Computationally infeasible to find \( m_1, m_2 \) such that \( H(m_1) = H(m_2) \)

- Example algorithms
  - MD5
  - SHA-1
  - SHA-256

Applications of Hash Functions

- Primary application
  - Generate/verify digital signature

  \[ m \] \[ H(m) \] \[ \text{Sign} \] \[ \text{Signature} \] \[ \text{Sig}(H(m)) \] \[ \text{Yes/No} \]
Applications of Hash Functions (Cont’d)

- **Password hashing**
  - Doesn’t need to know password to verify it
  - Store $H(password + salt)$ and salt, and compare it with the user-entered password
  - Salt makes dictionary attack more difficult

- **Message integrity**
  - Agree on a secret key $k$
  - Compute $H(m | k)$ and send with $m$
  - Doesn’t require encryption algorithm, so the technology is exportable

Applications of Hash Functions (Cont’d)

- **Authentication**
  - Give $H(m)$ as an authentication token
  - Later release $m$

Pseudo Random Generator

- **Definition**
  - A *cryptographically secure pseudorandom bit generator* is an efficient algorithm that will expand a random $n$-bit seed to a longer sequence that is computationally indistinguishable from a truly random sequence.

  - **Theorem [Levin]**
    - A *one-way function* can be used to construct a cryptographically secure pseudo-random bit generator.
Pseudo Random Functions

• Definition
  – A cryptographically secure pseudorandom function is an efficient algorithm that
  • given an n-bit seed s, and
  • an n-bit argument x,
  • returns an n-bit string \( f(s)(x) \)
  • such that it is infeasible to distinguish \( f(s)(x) \) for random seed s from a truly random function.

• Theorem [Goldreich, Goldwasser, Micali]
  – Cryptographically secure pseudorandom functions can be constructed from cryptographically secure pseudorandom bit generators.

Key Agreement

• Establish a key between two or among multiple parties
  – Classical algorithm
    • Diffie-Hellman

Key Exchange

• Key exchange
  – Between two parties
  – A special case of key agreement
  – Use public key cryptography
    • Examples: RSA, DH
  – Use symmetric key cryptography
    • Usually requires a pre-shared key
Key Distribution

- Involves a (trusted) third party to help establish keys.
- Based on
  - Symmetric key cryptography, or
  - Public key cryptography

Center-Based Key Management

- Key Distribution Center (KDC)
  - Communication parties depend on KDC to establish a pair-wise key.
  - The KDC generates the cryptographic key
    - Pull based
      - Alice communicates with the KDC before she communicates with Bob
    - Push based
      - Alice communicates with Bob, and it’s Bob’s responsibility to contact the KDC to get the pair-wise key

An Example of KDC: Kerberos

1. Request TGT
2. TGT + session key
3. Request SGT
4. Ticket + session key
5. Request service
6. Server authenticator
When Public Key Cryptography is Used

- Need to authenticate public keys
- Public key certificate
  - Bind an identity and a public key together
  - Verify the authenticity of a party’s public key

Attacks

- Replay attacks
- Man-in-the-middle attacks
- Resource clogging attacks
- Denial of service attacks
- Meet-in-the-middle attacks
- Dictionary attacks
- Others specific to protocols