Security & Privacy

Arijit Mondal
Dept. of Computer Science & Engineering
Indian Institute of Technology Patna
arijit@iitp.ac.in

Introduction

- Foremost design concerns for cyber physical systems
- Privacy - state of being kept way from observation
- Security - state of being protected from harm
- Cyber physical systems are being increasingly networked

Properties and threat model

- Secrecy / Confidentiality
  - Can secret data be leaked to attacker
- Integrity
  - Can system be modified by attacker
- Authenticity
  - Who is the system communicating / interacting with
- Availability
  - Is the system always be able to perform its function
  - Absence of denial-of-service

Basic of cryptography: One time pads

- Assume a message M and a key K are n bit words
- One easiest option to create cipher text as follows $C = M \oplus K$
- To decode C

$$C \oplus K = (M \oplus K) \oplus K = M \oplus (K \oplus K) = M$$

- This uses the fact that exclusive OR is associative and commutative, that $B \oplus B = 0$ for any B and that $B \oplus 0 = B$ for any B
- Without knowing K, C has no information
- A key should be used only once
  - $C_1 = M_1 \oplus K$, $C_2 = M_2 \oplus K$ then $C_1 \oplus C_2 = M_1 \oplus M_2$
  - It reveals some information about M_1 and M_2
  - If one is known the other can be determined easily

Symmetric key cryptography

- Block cipher
  - k bit key, n bit plain text M, n bit cipher text C
    - $E(K, M) = C$
  - Decryption is the inverse function $D_K = E_K^{-1}$
- DES is one of the popular block cipher
  - 56 bit key, 64 bit message
- AES
  - 128 bit message
  - Key length can be 128, 192, 256

Public key cryptography

- Each participant has two keys - public (K) and private (k)
- Suppose A wants to send message to B
- The message needs to be encrypted with public key of B — K_B
- The message can be decrypted with private key of B — k_B
- Public and private key match via clever algorithm
- Relies on one way function, easy to compute, hard to reverse without knowing private key

RSA

- Key generation
  - Select two large prime numbers p, q and compute $n = pq$
  - $\phi(n)$ - number of integers less than n that are relatively prime to n - $(p-1)(q-1)$
  - Select a random number e, $1 < e < \phi(n)$ such that $gcd(e, \phi(n)) = 1$
  - Compute d, $1 < d < \phi(n)$, such that $ed \equiv 1 \pmod{\phi(n)}$
  - Key - public key: (n, e), private key: d
- Encryption
  - $C = M^e \pmod{n}$
- Decryption
  - $C^d \pmod{n} = M$

Secure hash function

- Provide check for integrity / authenticity
- Properties
  - n bit message maps into k bit hash value, n can be arbitrary large
  - Efficient to compute $H(M)$
  - Pre-image resistance - computationally infeasible to find a message M such that $h = H(M)$ for a given h
  - Second pre-image resistance - Given M_1, it should be computationally infeasible to find another message M_2 such that $H(M_1) = H(M_2)$
  - Collision resistance - It should be computationally infeasible to find two different messages M_1, M_2 where $H(M_1) = H(M_2)$

Digital signature

- Provide check for integrity / authenticity
- Based on public key cryptography
- Author of a digital document authenticate himself to a third party and to provide integrity of the document
- Steps
  - Key generation
  - Signing
  - Verification
- RSA encryption with private key and send M and $S = M^d \pmod{n}$

Key exchange protocol

- Communication is done over public channel, attacker can view the messages
- Steps
  - Two numbers p (large prime) and n $(1 < n < p-1)$ are selected and agreed by both the parties
  - A selects a number a from $\{0, 1, \ldots, p-2\}$, similarly, B selects b (a, b are private)
  - A computes $n^a \pmod{p}$ and sends to B. B computes $n^b \pmod{p}$ and sends to A.
  - They can decide the key as $K = n^{ab} \pmod{p}$

$$A^b \pmod{p} = n^{ab} \pmod{p} = B^a \pmod{p}$$

Software security

```
char sensor_data[16];
int secret_key;
void read_sensor_data() {
  int i = 0;
  // more_data returns 1 if there is more data, and 0 otherwise
  while(more_data()) {
    sensor_data[i] = get_next_byte();
    i++;
  }
  return;
}
```