Distributed Systems (COMP9243)

Lecture 11 (A): Security

Slide 1
A Introduction
A Cryptography
B Secure protocols and communication
B Authentication
B Authorisation

Slide 2

Security in Distributed Systems

Confidentiality: information disclosed/services provided only to authorised parties

Integrity: alterations can only be made in an authorised way

Availability: system is ready to be used by authorised parties

Slide 3

The CAST

Slide 4

Alice: Sends a message to Bob
Saying to meet her somewhere.

Bob: Oh huh.

But Eve sees it too, and goes to the place.

With you so far.

Bob: Is delayed. And
Alice and Eve meet.

Yeah?

I've discovered a way to get computer scientists to listen to any boring story.
The Good Guys:
- Alice, Bob
- Want to communicate securely

The Bad Guys:
- Eve
- The eavesdropper — tries to thwart Alice and Bob’s plans

The Alice and Bob After Dinner Speech:
- google it for more about Alice and Bob

**Authorised Actions**

Security is about making sure that only authorised actions are performed in the system.

**Example Actions:**
- Reading data
- Modifying data (writing, creating, deleting)
- Using a service
- Managing a service

All of these could be abused if performed in unauthorised ways.

**Security Policy**

Security is a question of tradeoffs

**Security Policy:**
- A statement of security requirements
- Describes which actions entities in a system are allowed to take and which ones are prohibited
  - Entities: users, services, data, machines, etc.
  - Operations: read, write, send, start, stop, etc.

**Example:**
- Everyone (staff and students) has an account
- Access to course accounts must be approved
- Only course accounts can modify grades

Anything missing?

**Breaking Security**

**Vulnerability:**
A vulnerability is a weakness in the system that could potentially be exercised (accidentally triggered or intentionally exploited) to cause a breach or violation of the system’s security policy.

**Threat:**
A threat is a possible breach of security policy (the potential for an attack). A concrete threat consists of a threat-source and an exercisable vulnerability.

**Attack:**
When a vulnerability is exercised we call this an attack.
Classes of Security Threats

**Interception:** unauthorised party has gained access to a service or data

** Interruption:** service or data become unavailable, unusable, destroyed, etc.

**Modification:** unauthorised changing of data or tampering with a service (so that it no longer adheres to its specifications)

**Fabrication:** additional data or activity are generated that would normally not exist

Attacking a Distributed System

**Attacking the Communication Channel:**
- Eavesdropping
- Masquerading
- Message tampering
- Denial of service

**Attacking the Interfaces:**
- Unauthorised access
- Denial of Service

**Attacking the Systems:**
- Applications
- OS
- Hardware

Protecting a Distributed System

**Controls:**

**Authentication:** verify the claimed identity of an entity

**Authorisation:** determine what actions an authenticated entity is authorised to perform

**Auditing:** trace which entities access what

**Message Confidentiality:** secret communication

**Message Integrity:** tamperproof messages

Security Mechanisms

**Good Mechanisms:**

**Encryption:** transform data into something an attacker cannot understand
- A means to implement confidentiality
- Support for integrity checks (check if data has been modified)

**Signatures and Digests** support for integrity, authentication

**Secure Protocols** support for authentication, authorisation

**Secure Communication** support confidentiality and integrity

**Security Architecture** based on sound principles such as:
- small TCB, Principle of Least Privilege, support for authorisation
Less Good Mechanisms:

- **Obscurity**: count on system details being unknown
- **Intimidation**: count on fear to keep you safe

**WHY SECURITY IS HARD**

- **Weakest Link**: Security of a system is only as strong as its weakest link
- **Need to make sure all weak links are removed**
- **One bug is enough**
- **People are often the weakest link**

- **Complexity**: Security involves many separate subsystems
- **Complex to set up and use**
- **People won’t use complex systems**

- **Pervasiveness**: Application level
- **Middleware level**
- **Network level**
- **OS level, Hardware Level**

**HOW TO MAKE IT EASIER**

- **Distribution of Mechanisms**: Trusted Computing Base (TCB): those parts of the system that are able to compromise security
- **The smaller the TCB the better.**
- **May have to implement key services yourself**
  - Physically separate security services from other services

- **Simplicity**: Simplicity contributes to trust
- **Very difficult to make a simple secure system**

**FOUNDATIONS**

- **Cryptography**
  - Ciphers
  - Signatures and Digests
  - Secure Communication
  - Security Protocols
- **Authentication**
- **Authorisation**
The Basic Idea:

- Plaintext, P
- Encryption key, E
- Encryption method
- Decryption method
- Ciphertext, C
- Decryption key, D
- Active intruder can alter messages
- Active intruder can insert messages
- Passive intruder only listens to C

Encryption:

- Map cleartext (or plaintext) T to ciphertext (or cryptogram) C
- Mapping is by a well-known function parameterised by a key K
- Infeasible to reconstruct from C without knowledge of key K
- \( E(K_E, T) = \{T\}_{K_E}; \ D(K_D, C) = \{C\}_{K_D}; \ \{T\}_{K_E} K_D = T \)

Cryptographer:

- Uses cryptography to convert plaintext into ciphertext

Cryptanalyst:

- Uses cryptanalysis to attempt to turn ciphertext back into plaintext
- Cryptanalysis: the science of making encrypted data unencrypted

What properties should a good cipher possess?

- Confusion and Diffusion
  - Confusion: every bit of key influences large number of ciphertext bits
  - Diffusion: every bit of plaintext influences large number of ciphertext bits
- Fast to compute, ideally in hardware
- Not critically depend on users selecting “good” keys
- Have been heavily scrutinised by experts
- Based on operations which are provably “hard” to invert
- Easy to use
In practice, keys are of finite length. Consequences?

- Finite key space ⇒ susceptible to exhaustive search
- Longer keys ⇒ more time needed for brute-force attack
  - Time to guess a key is exponential in the number of bits of the key
- Longer keys also make $E$ and $D$ more expensive
- Cipher must be secure against any systematic attack significantly faster than exhaustive search of key space

**Basic Ciphers**

**Substitution Ciphers:**
- Each plaintext character replaced by a ciphertext character
- Caesar cipher: shift alphabet $x$ positions
  - Easy to break using statistical properties of language
- Book cipher: replace words by location of word in book
  - Knowledge of book is the key

**One Time Pads:**
- Random string XORed with plaintext
- Information theoretically secure
- Random string must:
  - Have no pattern or be predictable
  - Not be reused
  - Not be known by cryptanalyst
- Key distribution problem

**Symmetric Ciphers**

- $E = K_E$
- $D = K_D$
- Secret key $K_E = K_D$
- Fast ⇒ suited for large data volumes
- Secure channel needed to establish the shared, secret key
- How many keys needed for $N$ agents?
  - For any two agents, one key is needed

**Tiny Encryption Algorithm (TEA)**

Symmetric encryption algorithm by Wheeler & Needham:
- Encode a 64-bit block ($text$) consisting of two 32-bit integers
- Using a 128-bit key ($k$) represented by four 32-bit integers
- Despite its simplicity, TEA is a secure and reasonably fast encryption algorithm
- Can easily be implemented in hardware
- Approximately three times as fast as DES
- Achieves complete diffusion
void encrypt (unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];
    unsigned long delta = 0x9e3779b9, sum = 0; int n;
    for (n = 0; n < 32; n++) {
        sum += delta;
        y += ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
        z += ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
    }
    text[0] = y; text[1] = z;
}

void decrypt (unsigned long k[], unsigned long text[]) {
    unsigned long y = text[0], z = text[1];
    unsigned long delta = 0x9e3779b9, sum = delta << 5; int n;
    for (n = 0; n < 32; n++) {
        z -= ((y << 4) + k[2]) ^ (y + sum) ^ ((y >> 5) + k[3]);
        y -= ((z << 4) + k[0]) ^ (z + sum) ^ ((z >> 5) + k[1]);
        sum -= delta;
    }
    text[0] = y; text[1] = z;
}

**Slide 25**

32 rounds: shift and combine the halves of text with the four parts of the key
- Constant delta is used to obscure the key in portions of the plaintext that do not vary
- Confusion (xor operations and shifting of the key) and diffusion (shifting and swapping of the two halves of the text)

**Slide 26**

**Other Symmetric Ciphers**

**Data Encryption Standard (DES):**
- Developed by IBM for US government
- 56 bit key. No longer considered safe.
- Triple DES: 2x56 bit key. encrypt-decrypt-encrypt

**International Data Encryption Algorithm (IDEA):**
- Uses 128-bit key to encrypt 64-bit blocks
- Approximately three times as fast as DES
- Same function for encryption and decryption (like DES)

**Advanced Encryption Standard (AES):**
- Defined in 2001, to replace DES
- Variable block and key length: specification 128, 192, or 256 bit keys and 128, 192 or 256 bit blocks

**Slide 27**

**Asymmetric Ciphers**

- Due to Diffie & Hellman & Merkle (1976)
- Instead of one secret key per pair of agents, one public/private key pair per agent
- $K_E \neq K_D$: $K_D$ infeasible to compute from $K_E$
Each agent can publish public key $K_E = K_P$, keep private key $K_D = K_p$ secret.

Too slow to encrypt large volumes of data.

Examples: RSA and variants of Diffie & Hellman’s original algorithm, such as ElGamal.

**Slide 30**

How they work:
- Trap-door functions: one-way functions with a secret exit
- Easy to compute in one direction, but infeasible to invert unless a secret (secret key) is known
- Key pair is usually derived from a common root (such as large prime numbers) such that it is infeasible to reconstruct the root from the public key

**Slide 31**

**Block ciphers**
- Encrypt fixed-size blocks of data (e.g., 64 bits), one at a time
- Requires some padding in the last block: why is this a weakness?
- Blocks of ciphertext are independent
  - Attacker may spot repeating patterns and infer relationship to plaintext: how?
- Cipher block chaining

**Slide 32**

**Stream cipher**
- Encode a given plaintext bit by bit (e.g., voice)
- Xor a keystream (sequence of ‘random’ bits) with the plaintext
- Keystream: Output of a random number generator encoded with a block cipher algorithm
- How does the receiver reconstruct the plaintext?
  - Generate the same keystream and xor it with the ciphertext
  - requires starting value of RNG and the secret key
- Under which conditions can partial message loss be tolerated?

Note: This is not the same as a One Time Pad
**Secure Hash (Digest)**

Cryptographically ensure message integrity and authenticate originator.

How can we check whether a message has been altered?

- Secure digest or hash
- Fixed-length value condensing information in the message
- Given message $M$ and hash $H(M)$, it must be very hard to find $M'$ with $H(M) = H(M')$
- If hash $H(M)$ is the same after transmission, message is unaltered with very high likelihood

Hash functions:

- 128-bit constant
- Padded message (multiple of 512 bits)

**Digital Signature**

- How to verify who sent the message

Sender:

- Given a message $M$ and sender private key $K_{pri}$, signed message:

\[(M, (H(M))_{K_{pri}})\]

Must be resilient to:

- Collision:
  - find $m_1$ and $m_2$ such that $H(m_1) = H(m_2)$
  - related to birthday attack
- Pre-image:
  - given $h$, find $m$ such that $H(m) = h$
- Second pre-image:
  - given $m_1$ find $m_2$ such that $H(m_1) = H(m_2)$

Does a hash provide:

- confidentiality?
- integrity?
- authenticity?
- non-repudiation?
Slide 37

Receiver:

- Recipient uses matching public key $K_{pub}$ to recover digest
- Compare recovered digest to result of computing $H(M)$
- If same, sent message must be unaltered and sender the owner of $K_{pri}$

**Reading list**

*Slide 38*  