Security Attacks, Services and Mechanisms

- **Security Attack**: Any action that compromises the security of information.

- **Security Mechanism**: A mechanism that is designed to detect, prevent, or recover from a security attack.

- **Security Service**: A service that enhances the security of data processing systems and information transfers. A security service makes use of one or more security mechanisms.
Security Attacks (Threats)

(a) Normal flow

(b) Interruption
Attack on availability

(c) Interception
Attack on confidentiality

(d) Modification
Attack on integrity

(e) Fabrication
Attack on authenticity

Threats in Data Communication

- **Eavesdropping**: intercepting and reading messages intended for another process.
- **Traffic analysis**: by monitoring frequency and length of messages, even encrypted, nature of communication may be guessed; Difficult to detect, but can be prevented.
- **Masquerading**: sending or receiving messages using another process’s identity, i.e. pretending to be a different entity
- **Replaying**: using previously sent messages to gain another process’s privileges.
- **Message tampering**: intercepting and altering messages intended for another process.
- **Denial of service**: preventing or inhibiting the normal use or management of communication facilities.
Passive and Active Security Threats

- **Passive Threats**
  - eavesdropping on transmissions
  - monitoring frequency and length of messages to get info on senders;

- **Active Threats**
  - modification of the data stream or the creation of a false stream
  - easy to detect but hard to prevent, focus on detection and recovery

Security Services

- **Data confidentiality or privacy**: protect data content/access,
- **Data integrity**: protect data accuracy, i.e. make sure data has not been altered,
- **Authenticity**: protect data origin, i.e. make sure who created or sent the data,
- **Non-repudiation**: prevents sender from denying of creating and sending data,
- **Authorization or access control**: prevent misuse of resources,
- **Availability**: ensure timely service:
  - denial of service attacks
  - virus that deletes files.
Security Service (continued)

• Data Confidentiality or Privacy:
  — evidence that the two corresponding processes are the only ones that can read the transmission; the mechanisms involved in handling this service are encryption mechanisms.

• Data integrity:
  — verification that the data has not been corrupted by an outside source; the mechanisms involved in handling this service are different hash functions.

• Authenticity:
  — guarantee that the message has come from the process who claims to have sent it; the mechanisms used to handle this service are authentication exchange mechanisms.

Security Service (continued)

• Non-repudiation:
  — verification that a message had been sent by another process in the event the other process tries to deny it at a later time. The mechanisms used to handle this service are digital signature mechanisms.

• Authorization or Access control:
  — insurance that only the authorized processes can gain access to protected resources.
Security Mechanisms (Tools)

- Cryptographic Algorithms
- Hash (Digest) Algorithms
- Authentication Mechanisms
- Digital Signatures
- Traffic Padding
- Key Management Protocols

Cryptography

- Cryptography is a mechanism against eavesdropping and it provides confidentiality, also referred as privacy.
- Fundamental tenet of cryptography:
  
  *If lots of smart people have failed to solve a problem, then it probably won’t be solved (soon).*

- Cryptographic Algorithms:
  - Symmetric or secret key cryptography
  - Public key cryptography
Symmetric or Secret Key Cryptography

- Uses the same key for both encryption and decrypting
- Both sender and receiver need to have the same key in order to communicate successfully; widely used and very popular.

**Advantages:**
- fast relative to public key cryptography,
- considered secure providing the key is relatively long,
- the cryptotext is compact.

**Disadvantages:**
- a large number of keys is needed to communicate securely with a large group of people,
- the administration of the keys can become extremely complicated,
- key distribution and renewal is a huge problem; the key is subject to interception by hackers,
- non-repudiation is not possible.
Attacking Encryption

- Requirement for security is a strong encryption algorithm, thus unable to decrypt without key:
  - even when algorithm is known (and that is often the case),
  - even if many plaintexts & ciphertexts available.
- Cryptanalysis:
  - relay on nature of algorithm plus some knowledge of general characteristics of plaintext,
  - attempt to deduce plaintext or key.
- Brute force:
  - try every possible key until plaintext is recovered,
  - rapidly becomes infeasible as key size increases,
  - 56-bit key is not secure.

Symmetric Cryptography Algorithms

- DES - Data Encryption Algorithm; published in 1977
  - encrypts 64-bit block of plaintext into 64-bit block of ciphertext using a 56-bit key,
  - the key of 56 bits for a long time had been pretty much universally acknowledged to be too small to be secure,
  - broken, in less than three days, in 1998 by Electronic Frontier Foundation by special purpose US$250,000 machine,
- 3DES or EDE (Encrypt-Decrypt-Encrypt)
  - DES algorithm used three times
  - involves repeating basic DES three times, using two or three unique DES keys, for a key size of 112 or 168 bits
Symmetric Cryptography Algorithms (cont.)

- Advanced Encryption Standard (AES); issued in 1997
  - security strength equal to or better than 3DES
  - significantly improved efficiency
  - symmetric block cipher with block length 128 bits
  - key lengths 128, 192, and 256 bits
- RC4
  - a proprietary encryption schema marketed by RSADSI
  - a stream encryption schema:
    - first, both sender and receiver produces (efficiently) an unbounded length pseudorandom stream from a varying length key (1 to 256 bytes),
    - then, the stream is XORed (by sender) with the data for encryption.
    - receiver XORes encrypted data for decrypting.

Encrypting Larger Messages

- The block algorithms encrypt a fixed size block, e.g. 64 bits.
- Obvious solution is break a message into blocks and encrypt a block at a time.
- This is called Electronic Code Book (ECB)
- ECB has two serious flaws:
  - since repeated plaintext blocks yield repeated ciphertext blocks, this will give an eavesdropper some information
  - encryption does not guarantee integrity, since blocks can be rearranged or modify without automatic detection,
- As result ECB is rarely used to encrypt messages.
- Other modes “chain” to avoid this, such as Cipher Block Chaining (CBC).
Electronic Code Block Encryption

Message

m1 m2 m3 m4 m5 m6 m7 m8

Break into blocks

Encrypted with secret key

c1 c2 c3 c4 c5 c6 c7 c8

Electronic Code Block Decryption

Message

m1 m2 m3 m4 m5 m6 m7 m8

Decrypted with secret key

Reassembled blocks

c1 c2 c3 c4 c5 c6 c7 c8
Randomized ECB Encryption

- To decrypt this, you would decrypt all $c_i$s, and for each $c_i$, after decrypting it, you would xor it with the random number $r_i$.
- But inefficient; twice as much data to be transmitted, since $r_i$s have to be also transmitted.

Cipher Block Chaining Encryption

- CBC generates its own “random numbers”; $c_i$ is used as $r_{i+1}$
- CBS generates the initial random number IV (initialization vector) to avoid having two plaintext messages that start the same to have the same ciphertext at the beginning; IV is also transmitted.
Cipher Block Chaining Decryption

- Decryption is simple because xor is its own inverse
- Since the cost of xor is trivial comparing to the cost of an encryption CBC has the same performance as ECB, except the cost of generating and transmitting IV.

Figure 21.4

Location of Encryption Devices

Presentation L 21

Presentation L 22
Location of Encryption Devices (cont.)

- **Link encryption:**
  - each communication link equipped at both ends,
  - all traffic secure,
  - high level of security,
  - but, requires lots of encryption devices,
  - and message must be decrypted at each switch to read address,
  - thus, security vulnerable at switches; particularly on public ones.

- **End-to-End encryption:**
  - encryption done at ends of system,
  - data in encrypted form crosses network unaltered,
  - destination shares key with source to decrypt,
  - but, host can only encrypt user data, otherwise switching nodes could not read header and route packet
  - hence traffic pattern not secure.

- **Solution is to use both link and end to end**

---

Message Authentication

- Encrypting messages does not provide **integrity protection**, meaning that after receiving an encrypted message, we are not automatically able to say if the message has been tampered with.

- Message authentication should protect against active attacks with:
  - falsification of data,
  - falsification of source.

- Message authentication allows a receiver to verify that a message is authentic implying:
  - it has not been altered,
  - it is from the claimed/authentic source,
  - its timeliness, e.g. it has not been artificially delayed and replayed.
Authentication Using Symmetric Encryption

- It is possible to perform authentication simply by the use of symmetric encryption:
  - assuming sender & receiver only know key then,
  - only sender could have encrypted message for other party and
  - encrypted message should include:
    - error detection code to assure that no alterations have been made,
    - sequence number to assure that sequencing is proper,
    - time stamp to assure that the message has not been delayed beyond that normally expected for network transit.

Authentication Without Encryption

- Message is not encrypted, and it can be read by anybody, but authentication tag generated and appended to each message.
- Useful when don’t want encryption because:
  - messages broadcast to multiple destinations:
    - have one destination responsible for authentication,
  - one side heavily loaded:
    - encryption adds to workload,
    - can authenticate random messages,
  - authentication of a computer program in plaintext is an attractive service; a program can be executed without having to decrypt it every time, but with check for integrity.
- Hashing (digest) algorithms are used for generating authentication tags, without need to encrypt message content.
Message Hash (Digest) Algorithms

- Message digest algorithms are one-way functions and they are designed to solve the problem of authenticity and data integrity.
- Calculation of the digest is infinitely easier compared to the process of retrieving the message from the digest.
- Message digest algorithms generally possess the following characteristics:
  - they take in any length input and generate a fixed length output called a hash or a message digest,
  - it is not computationally feasible to calculate the message based on the digest,
  - it is not computationally feasible to find two messages which generate the same digest.
- Message digest algorithms: MD2, MD4, MD5 all with 128 bit digests, and SHA-1 with 160 bit digest.

Message Authentication Using Hashing

- With conventional encryption of a digest

Figure 21.7a
Message Authentication Using Hashing (cont.)

- Using secret value

Public Key Encryption

- Uses one key for encryption and another for decrypting.
- Each user generates (has) two keys - one public & one private,
  - public key and private key are mathematically linked.
- User places its public key in public domain, i.e. it is revealed to all,
  - private key remains a secret, i.e. know only to its owner.
- Encryption is performed with the public key of receiver,
- Decrypting can be only performed with the private key,
  - infeasible to determine the decryption (private) key given the encryption (public) key, the algorithm and ciphertext.
Public Key Encryption: Confidentiality

Figure 21.9a

Public Key Encryption: Characteristics

• Advantages:
  – considered very secure,
  – no form of secret is required, thus reducing key administration to minimum,
  – the number of key managed by each user is much less compared to secret key cryptography,
  – supports non-repudiation.

• Disadvantages:
  – much slower compared to secret key cryptography,
  – the ciphered text is larger than the plaintext.
Public Key Encryption Algorithms

- **RSA algorithm** (Rivest, Shamir & Adleman):
  - it supports variable length key, thus a longer key for enhanced security, or a short key for efficiency,
  - the most commonly used key is 512 or 1024 bits,
  - the block size in RSA, i.e. the chunk of data to be encrypted, is also variable and the plaintext block must be smaller than key length,
  - resulting ciphertext will be of the length of the key.

- **Diffie-Hellman algorithm**
- **Digital Signature Standard algorithm (DSS)**

**RSA: Key Generation**

- Select two prime numbers \(p\) and \(q\); \(p = 17\) and \(q = 11\) selected;
- Calculate \(n = p \times q\); \(n = 17 \times 11 = 187\) calculated;
- Calculate \(\phi(n) = (p - 1)(q - 1); \phi(n) = 16 \times 10 = 160\) calculated;
- Select \(e\) such that \(e\) is relatively prime to \(\phi(n)\) and \(e < \phi(n)\); \(e = 7\) selected;
- Determine \(d\) such that \((d \times e) \mod \phi(n) = 1\) and \(d < \phi(n)\); \(d = 23\), determined; it is correct because \(23 \times 7 = 161 = 10 \times 16 + 1\);
- The resulting keys are:
  - public key \(PU = \{e, n\} = \{7, 187\}\,
  - private key \(PR = \{d, n\} = \{23, 187\}\).
RSA: Encryption and Decryption

- Given public key \{e, n\} and a plaintext \(M\), the encryption:
  \[ C = M^e \mod n \]

- Given private key \{d, n\} and a ciphertext \(C\), the decryption for:
  \[ M = C^d \mod n = (M^e)^d \mod n = M^{ed} \mod n \]

![Diagram of encryption and decryption](image)

**Figure 21.11**

RSA Security

- Brute force by trying all possible keys:
  - larger size of \(e\) and \(d\), the more secure the algorithm,
  - but larger keys do slow calculations in key generation and in encryption/decryption.

- Most discussion about RSA focused on the task of factoring \(n\) to recover \(p\) & \(q\):
  - a very hard mathematical problem,
  - key size of 1024-bits, i.e. 300 decimal digits, currently secure for most applications.

- 129 decimal digits challenge broken in April of 1994, by a group working over the Internet and using over 1600 computers claimed the prize after only eight months of work.
Public Key Encryption: Authentication

- Non-repudiation, but it is really slow as always with PK.
- Solutions:
  - digital signature,
  - digital envelope.

Public Key Encryption: Digital Signature

- Message authentication with public key encryption:
  - integrity,
  - non-repudiation.
Creating Digital Envelope

1. Message → Symmetric encrypting function → Encrypted message
2. Random symmetric key → Public key encrypting function → Encrypted symmetric key
3. Receiver’s public key
4. Encrypted symmetric key
5. Encrypted message

Verifying Digital Envelope

1. Encrypted message
2. Encrypted symmetric key
3. Encrypted symmetric key → Public key decrypting function → Decrypted symmetric key
4. Decrypted symmetric key
5. Receiver’s private key
Certification Authority

- The private key is secure because owner X need never reveal it.
- However, others must be sure that the public key with X's name written all over it is in fact X's public key, since someone else could have broadcast a public key and said it was X's.
- The solution to this problem is the public-key certificate.
- In essence, a certificate consists of a public key plus a User ID of the key owner, with the whole block signed the private key by a trusted third party.

Certification Authority (cont.)

- Typically, the third party is a certificate authority (CA) that is trusted by the user community and its public key is distributed, thus available to everybody.
- A user can present public key to the authority in a secure manner and obtain a certificate signed by the CA's private key.
- The user can then publish the certificate.
- Anyone, needing an user's public key, can obtain the certificate and verify that it is valid by way of the attached trusted signature.
- But is CA to be trusted? CA distributes its certificate.
- Who certifies CA's certificate? Hierarchy of CAs.
Public Key Certificate

![Diagram of Public Key Certificate]

**Figure 21.12**

---

**Authentication Handshake**

\[
[XYZ]_{ABC} = \text{encryption of } XYZ \text{ by } ABC\text{'s private key}
\]

Protocol 1. Bob authenticates Alice based on her public key signature

\[
\{XYZ\}_{ABC} = \text{encryption of } XYZ \text{ by } ABC\text{'s public key}
\]

Protocol 2. Bob authenticates Alice if she can decrypt a message encrypted with her public key

---

Presentation L 43

Presentation L 44
Authentication Handshake (cont.)

\[ [XYZ]_{ABC} = \text{encryption of } XYZ \text{ by } ABC\text{'s private key} \]
\[ \{XYZ\}_{ABC} = \text{encryption of } XYZ \text{ by } ABC\text{'s public key} \]

\[ \text{I'm Alice, } \{R_2\}_{Bob} \]
\[ R_2, \{R_1\}_{Alice} \]
\[ R_1 \]

Protocol 3. Mutual authentication based on public key

Secure Socket Layer - SSL

- SSL is designed for a user-level process, and runs on top of TCP
- Session keys are derived during the initial handshake.

- Alice chooses: secret S
- Alice computes: \( K = f(S, R_{Alice}, R_{Bob}) \)

\[ \text{I want to talk, ciphers I support, } R_{Alice} \]
\[ \text{my certificate, cipher I choose, } R_{Bob} \]
\[ \{S\}_{Bob}, \text{(keyed hash of handshake messages)} \]
\[ \text{(keyed hash of handshake messages)} \]
\[ \text{data protected with keys derived from } K \]

- Bob computes: \( K = f(S, R_{Alice}, R_{Bob}) \)

- This handshake procedure is a characteristic for a client (without certificate) securely accessing a server (with certificate).
Secure Socket Layer – SSL (cont.)

Message 1: Alice says she would like to talk (but doesn’t identify herself), and gives a list of cipher algorithms, along with a random number $R_{Alice}$, that will contribute to form of various keys,

Message 2: Bob sends Alice his certificate, responds with one of ciphers listed, and a random number $R_{Bob}$, that will also contribute to form various keys,

➢ After receiving message 2, Alice chooses a random number $S$ (known as the pre-master secret) and computes the master secret $K$ as the function of $S$, $R_{Alice}$ and $R_{Bob}$.

➢ Alice now derives session keys by hashing $K$, $R_{Alice}$ and $R_{Bob}$; the keys are for encryption/decryption, integrity and IV in each direction; total of six keys.

Secure Socket Layer – SSL (cont.)

Message 3: Alice sends $S$, encrypted with Bob’s public key, and keyed hash of $K$ and the handshake messages, both to prove she knows the key and to ensure that tampering of the handshake messages would be detected.

➢ After receiving message 3, Bob computes the master secret $K$ as the function of $S$, $R_{Alice}$ and $R_{Bob}$.

➢ Then Bob derives session keys by hashing $K$, $R_{Alice}$ and $R_{Bob}$; the same six keys as Alice.

Message 4: Bob proves he knows the session keys, and ensures that earlier messages arrived intact, by sending a keyed hash of $K$ and the handshake messages.
Autumn Semester 2012

CSE 3461/5461
Introduction to Computer Networking and Internet Technologies

The End