5. Security Services

Distributed Systems
Prof. Dr. Alexander Schill
http://www.rn.inf.tu-dresden.de
Outline

- Motivation and requirements
- Encryption
- Authentication
- Authorization
- Firewalls
- Anonymity
Motivation

- Online shop payment example
  - Perform credit/debit transaction to purchase products online
  - Check availability of funds before purchase

- Security issues
  - Ensure that the server is the server of your bank
  - Ensure that the client is the person owning the account
  - Protect messages against changes and replay
  - Ensure that only you are authorized to access your account
Requirements: Attack Scenarios

- **Inhibit**
  - Service Provision - Denial of Service through flooding with requests
  - Data Accessibility - Destruction of saved data to prevent its use
  - Communication - Prevention of information exchange by interrupting communications

- **Obtain**
  - Confidential information – Eavesdropping and copying of data
  - Access to resources - access to services by giving false identity

- **Modify / Falsify**
  - Data and messages
    - sum of bank transaction modified in favour of the attacker
    - unauthorized introduction of new data and messages
  - Services
    - gain unauthorised access to system and record activities
    - copy and resend service invocations (replay attacks)
Requirements: Protection Goals

- **Confidentiality**
  - Only authorized persons should have access to messages and data

- **Integrity**
  - Unauthorized changes of messages and data have to be detected and to be eliminated

- **Authenticity**
  - User identity can be validated

- **Access Control**
  - No unauthorized access to data and services is allowed

- **Accountability**
  - Interactions can be mapped to particular instances or persons who are accountable

- **Anonymity**
  - Executing procedures without disclosing user identities (as opposed to authenticity!)

- **Availability**
  - Permanent availability of services, information and communication connections
Protection of data to guarantee confidentiality (by encryption) and integrity (by encrypted checksums)

Encryption (cipher) at sender
Decryption (decipher) at receiver

Encryption algorithms parameterized by cryptographic keys held by sender and receiver

Encryption techniques
- Symmetric
- Asymmetric

Problem: key distribution over insecure communication channel
Encryption: Symmetric Cryptography

- Same key used for ciphering and deciphering
- Keys created by key generator
- Keys delivered to communication partners over a secure communication channel (usually based on additional encryption by predefined confidential keys)
Advantage:
  • Efficient method – appropriate for encryption of large amounts of data

Disadvantages:
  • Key must be securely distributed
  • Digital Signatures not possible (key is not unique)

Encryption Algorithms (cipher)
  • Stream cipher
    o Bitwise and/or characterwise encryption and decryption of data stream
    o Delay free and continuous
    o E.g. RC4 (Rivest’s Cipher 4)
  
  • Block cipher
    o Operates on data blocks of fixed size (e.g. 64 or 128 bit), more efficient (throughput)
    o E.g. RC5 (Rivest’s Cipher 5),
      DES (Data Encryption Standard),
      AES (Advanced Encryption Standard)
Encryption: Asymmetric Cryptography

- Different keys for ciphering and deciphering
- Unique key pair:
  - Public key for ciphering – distributable to everyone
  - Private key for deciphering – secret key
  - Key pair created by key owner (e.g. based on calculus of large prime numbers); public key derived by one-way function - inverse function not computable in reasonable time
  - So private key can not be derived from the public key
Advantage: Simple distribution of public key
- Email, Directory service, Web-site
- Secure communication channel not necessarily required

Disadvantage: Lower efficiency and necessary key certification
- Key generation algorithm requires expensive computation
- Possible attack:
  - Attacker could distribute a public key under false name
  - Trusting participants could possibly use this key and send confidential information (such as passwords) to the attacker
  - Solution: certification of public keys by digital signatures of trust centers

Example algorithms
- Rivest, Shamir and Adleman (RSA)
- Elliptic Curve Cryptography (ECC)
Encryption: Digital Signatures

- Check sum (message digest) generated by means of cryptographic secure hash procedure
- Inversion of asymmetric cryptographic procedure, i.e. private key for encryption and public key for decryption
- Private Key only known by sender
- Check sum from sender and calculated sum from receiver match?
  → message was sent by authentic sender
Solution to attacks involving distribution of false public keys:

- Trust Centers or Certificate Authorities certify public keys as belonging to users

- Method of Certification - Digital signatures by Trust Centers
  - Tuple <public key, username>
  - Time stamp and validity period

- Trust Centers themselves must be certified by higher-level Trust Centers, with some well-known top-level Trust Centers (e.g. hard-coded in web browsers)

- Revocation of certificates before expiration of validity period requires additional blacklists to be checked dynamically
Symmetric session keys distributed by asymmetric cryptography -> secure and simple
Session keys are then used for symmetric encryption and decryption of the actual user data -> high efficiency
Widespread in practice (e.g. Secure Socket Layer, Transport Layer Security, Pretty Good Privacy)
Encryption: Statistical Attacks

- **Problem**
  - Recognition of frequently occurring characters in encrypted messages
  - May be mapped to real world characters e.g. vowels (i,o,e,u,a)

- **Solution**
  - Change keys often to reduce amount of observable data
  - Use of long keys
    - e.g. 64 bits - too easy to break
    - 256 bits or more – very secure (for symmetric cryptography)
Ensures confidentiality and integrity (digital signatures)

Symmetric cryptography
- Secret key: same key for ciphering and deciphering
- Secure key distribution required (key distribution centre)
- Higher efficiency of cryptographic algorithms
- Digital signatures not supported (shared key)

Asymmetric cryptography
- Unique key pair: different keys for ciphering and deciphering
- Simple distribution of public key
- Lower efficiency of cryptographic algorithms
- Support of digital signatures (private key)

Combinations used in practice
- Asymmetric cryptography for distribution of secret keys
- More efficient symmetric cryptography for data encryption

Statistical attacks to be considered
Identification and Authentication

- **Identification:**
  - Presentation of an explicit identifier
  - Naming conventions important during binding (based on directory service)

- **Authentication:**
  - Verification of identity via presentation of a secret identifier
  - Secret as proof of identity
    - Use of private keys (e.g. derived from password)
    - Kerberos: Authentication of the client and of the server via decryption of the (session) key
  - Key distribution center: acts as authentication service
  - Additionally: timestamp for prevention of message replay

- Symmetric and asymmetric cryptography
Key Distribution Center (KDC)
- Centralized approach for key distribution
- Reliefs hosts from maintaining keys of any potential communication partner
- Trusted entity -> shares secret key with every host, hosts do not have to share keys

Basic usage
1. Client requests conversation key from KDC for communication with server
2. KDC generates conversation key and sends it to Client and server encrypted with the according secret keys $K_C$ and $K_S$

Issues
- Lack of synchronisation between client and server
- Replay attacks possible, e.g. by re-sending response messages after breaking CK
Authentication: Symmetric Cryptography
Needham-Schroeder authentication protocol

1. Client requests conversation key from KDC to communicate with Server, including random number $Z_1$
2. KDC response with same $Z_1$, conversation key and ticket (CK encrypted with $K_S$)
3. Client sends RPC request together with ticket and challenge $Z_2$ to server
4. Server answers with RPC response including the response to $Z_2$ and new challenge $Z_3$
5. Client responds to server challenge with $Z_3-1$, followed by server response in step 6.
Authentication: Symmetric Cryptography
Needham-Schroeder authentication protocol

- **Nonce**
  - random number \((Z_1, Z_2, Z_3)\) used only once, uniquely relates two messages to each other

- **Replay Attack**
  - Attacker wants to enforce the usage of an older, already compromised session key by re-sending a recorded old response message.
  - Inclusion of message \(Z_1\) in message 1 and 2 ensures that client can detect attack; similarly, further requests and responses are related by \(Z_2, Z_3\) (with subtraction), and so on.
Authentication: Symmetric Cryptography
Needham-Schroeder authentication protocol

- Key Distribution Center (KDC) generates conversation key CK

1. Key Request: $\text{key}_{\text{request}}(Z_1, C, S)$

2. Response: $K_C(Z_1, S, CK, K_S(C, CK))$

3. Request: $CK(Z_2, <\text{RPC}>, K_S(C, CK))$

4. Response: $CK(Z_2-1, Z_3, <\text{RPC-result}>)$

5. Request: $CK(Z_3-1, <\text{RPC}>)$

6. Response: $CK(<\text{RPC-result}>)$

- Modify attack
  - Attacker modifies communication partner in message 1
  - KDC sends conversation key and ticket for attacker
  - Client would think to communicate with server

- Integration of server identity in message 1 and 2 (encrypted) enables detection
Authentication: Symmetric Cryptography
Needham-Schroeder authentication protocol

**Key Distribution Center (KDC)** generates conversation key CK

1. `key_request (Z₁,C,S, Kₛ[Z₄])`

2. response `(Kₐ[Z₁,S, CKₜₖₜₖₛ[Z₄])]`

3. request `(CKₜₖₖₛ[Z₂,<RPC>], Kₛ[C, CKₜₖₖₛ[Z₄]])`

4. response `(CKₜₖₖₛ[Z₂₋₁, Z₃,<RPC-result>])`

5. request `(CKₜₖₖₛ[Z₃₋₁,<RPC>])`

6. response `(CKₜₖₖₛ[<RPC-result>])`

- **Weak point:** Replay of message 3 to establish secure channel with server
  - Attacker has stolen or broken an old conversation key (CKₚₜₑₖₛ)
  - Attacker replays an intercepted message 3; Server would believe that client is communication partner

- **Solution:** relation of message 1 and 3 using a nonce created by server
  - Client requests nonce from server before message 1 (not shown in diagram)
  - Server responds with nonce encrypted by own secret key Kₛ[Z₄] for protection (not shown)
  - Encrypted nonce is part of message 1, 2, and 3, so server could detect later replay (mismatch of nonce)
Authentication: Asymmetric Cryptography

- Client request – contains client identifier C and random number challenge Z_C for server

  - Z_C decrypted by server and sent back to client - server identity authenticated
  - Conversation key CK (to be used in further communications) and random number challenge Z_S for client also sent
  - Z_S decrypted by client, then encrypted (but this time with CK) and sent to server – client identity authenticated

- No key distribution center required; can even be further simplified: client sends challenge, server signs challenge with certified private key, client checks authenticity with public server key
Authorization

Awarding and control of access rights:
- well known from file systems, applied for distributed resources

Access Matrix: Capabilities for client or Access Control Lists for server
- Access Control Lists – Row in Matrix
- Capability, Ticket – Column in Matrix
- Subjects (Users: Persons, Roles, Groups)
- Objects (Resources of the distributed system)

<table>
<thead>
<tr>
<th>„Subject“</th>
<th>„Object“</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer X</td>
<td>Order</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Company Y – Business Customer</td>
<td></td>
</tr>
<tr>
<td>Administrator Warehouse Management</td>
<td></td>
</tr>
<tr>
<td>Access control list (ACL)</td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td></td>
</tr>
<tr>
<td>Customer Data</td>
<td></td>
</tr>
</tbody>
</table>
Authorization: Access Control Lists

- Defines access rights of subjects for certain resource ("object") in a decentralized way
- Easy validation of access for certain subjects, easy withdrawal of access rights
- Delegation of rights not possible; user group rights might lead to conflicting interpretation
- Black-List and White-List approach
  - Black List: rules for denying access (everything allowed)
  - White List: rules for permitting access (everything forbidden)
Authorization: Capabilities

- Defines access rights of certain subject for multiple resources
- Must be modification and forgery proof -> Digitally Signed
- More efficient access right evaluation per subject
- Delegation possible (e.g. secretary gets access rights on behalf of boss)

Problem: Higher expense to modify access rights later
- System administration can not easily withdraw capabilities of distributed users
- Objects must be renamed and new associated capabilities allocated
- Existing capabilities no longer valid because object has new name
Firewalls determine which:
- data packets to let through network segments (filtering)
- actions should be logged (Auditing)
- authentication and access requirements exist (Access Control, Authentication)
Firewalls: Filter types

- **Packet Filter**
  - Established on network and transport layer
  - Filter based on packet header (source, destination, protocol type etc.)
  - E.g. Filter rule: Deny requests to an internal web server if they do not originate from address area of internal network

- **Application Filter**
  - Work on application layer
  - Check content of all in and outgoing messages in application specific form
  - E.g. spam filter – scan messages for keywords and examine sender and receiver. Mark or delete spam messages

- **Proxy Filter**
  - Active on the transport layer using TCP and UDP
  - Mediate between clients in the external network if they wish to use internal services
  - Acts as server for the external clients and as client for the internal servers
Anonymity: Goals and methods

- **Goal:** Perform online activities without disclosing identity
  - anonymous message sending, e.g. to hide business relations between companies
  - non-disclosure of product orders or search queries for privacy reasons

- **Mixes**
  - Intermediate instance „Mix“; messages sent by sender to receiver via Mixes:
    - Mediates messages so that they arrive in an anonymous form, e.g. by modifying intermediate source and destination addresses
    - Hide linkage of incoming and outgoing data, for example by encrypting control information
  - Linkage may still be detectable based on message characteristics, e.g. length or structure

- **Pseudonyms**
  - Pseudonym – a fictitious name used when a person plays a particular role
  - Actions attributable to the pseudonym based on internal user profile
  - Limitations:
    - Delivery of goods to anonymous receiver – post-office box
    - User acting under several pseudonyms can be linked via similarity of actions
    - Information about user can be aggregated and combined from the different pseudonyms
Summary

- Nature of distributed systems → vulnerable to many possible attacks

- Various mechanisms described to protect against such attacks
  - Encryption – ensure confidentiality and integrity, aid authentication
  - Authentication – verify identity of users
  - Authorisation – Allow/disallow access by certain users to resources
  - Firewalls – Protect network areas
  - Anonymity – allow user actions without disclosing identity