peer-to-peer and agent-based computing

Security in Distributed Systems

UNIVERSITY OF ABERDEEN

Plan of lecture

1. Introduction and Design Issues
   a) Threats to security systems
   b) Security mechanisms
   c) Design issues: focus of data control, layering of security mechanisms and simplicity
2. Cryptography
   a) Basics of cryptography
   b) Types of encryption: symmetric, asymmetric and Hash functions
3. Secure Channels
   a) Using symmetric keys
   b) Using public/private keys
   c) Signing/Digital signatures
4. Secure Mobile Code: sandboxing

Security threats

4 possible threats to security in computer systems:

1. Interception: unauthorised party has gained access to a service or data
   – E.g. eavesdropping, illegal copying of data (or files)
2. Interruption: when a service or data becomes unavailable, unusable, destroyed etc.
   – E.g. when a file is lost or corrupted, denial of service by malicious attacks
3. Modification: unauthorised change in data or service
   – Intercepting and changing transmitted data
   – Changing the behaviour of a service
   – Altering database entries
4. Fabrication: generating data or activity that would not normally exist.
   – E.g., adding a password into a password file or database or falsifying a service

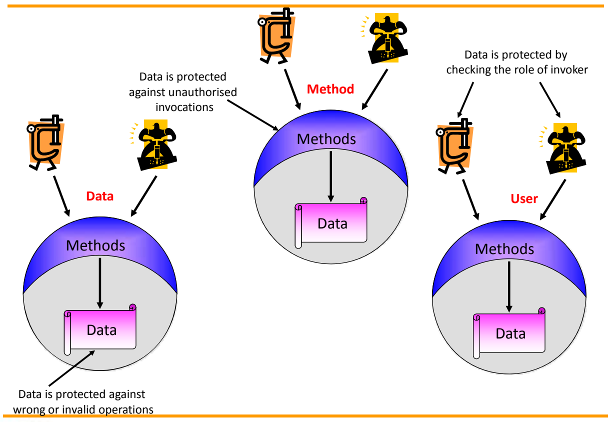
Security mechanisms

- Need a security policy
- Security mechanisms by which a policy can be enforced:
  - Encryption: transforms data (encrypts) into unintelligible data i.e. implements confidentiality, integrity
  - Authentication: verify the claimed identity of the user e.g. passwords, public/private-keys
  - Authorisation: is user authorised to access the resource? e.g. unix
  - Auditing: track which clients accessed what

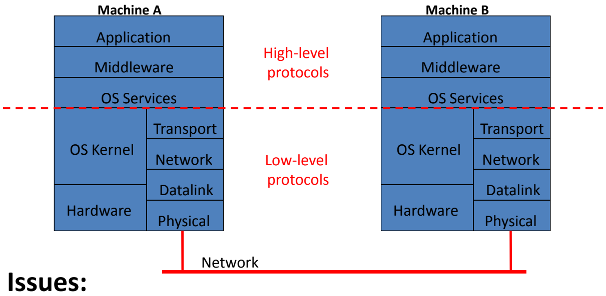
Design issues

- Focus of Control
- Layering of Security mechanisms
- Simplicity

Focus of data control



Layering of security mechanisms



Issues:

- At which level are the security mechanisms placed?
  - communication or services (Middleware)
- Depends on the trust of the client

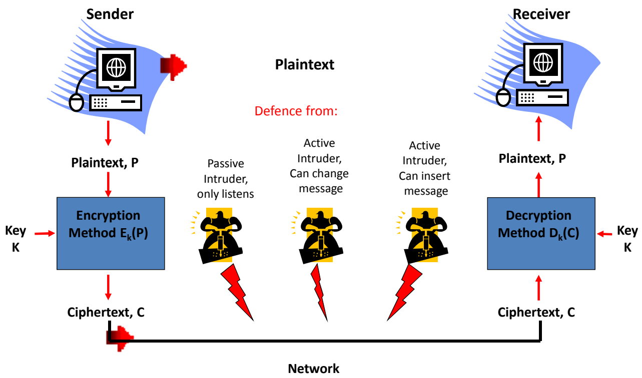
Simplicity

- The good news:
  - The fewer security mechanisms, the better!
  - The mechanisms
    - Need to be easily understood
    - And trusted to work!
  - Simplicity contributes to trust that end users put into the application
- The bad news:
  - Often applications are too complex and security only makes matters worse!

Cryptography

- Basics of cryptography
- Types of encryption
- Symmetric encryption
- Asymmetric encryption
- Hash functions

Basics of cryptography



Types of encryption

Text converted to ciphertext via algorithm & key
- Algorithm is publicly known
- Key is held private

Three main categories
1. Secret Key (symmetric cryptosystem)
   - single key is used to encrypt and decrypt information
2. Public/Private Key (asymmetric cryptosystem)
   - two keys are used: one for encryption (public key) and one for decryption (private key)
3. One-way Function (hash functions)
   - information is encrypted to produce a "digest" of the original information that can be used later to prove its authenticity

Symmetric encryption

- Sender and receiver have same secret key that will encrypt and decrypt plain text
  - Strength of encryption technique depends on key length
- Symmetrical algorithms:
  - Data Encryption Standard (DES) – 56-bit key
  - Triple DES, DESX, GDES, RDES – 168-bit key
  - RC2, RC4, RC5 – variable length, up to 2048 bits
  - IDEA – (basis of PGP) 128-bit key
  - Blowfish – variable length, up to 448 bits

Example: Data Encryption Standard (DES)

- Widely-used
  - Private (secret) key, so difficult to break it was restricted for export by US Gov.
  - 72,000,000,000,000,000 (72 quadrillion) or more possible encryption keys
- Key chosen at random
  - Both sender and receiver must know and use the same private key
- Can run in several modes and involves 16 rounds or operations
- Many companies use "triple DES", applying three keys in succession
- In 1997, Rivest-Shamir-Adleman, owners of another encryption approach, offered a $10,000 reward for breaking a DES message
- Cooperative effort on the Internet of over 14,000 computer users trying out various keys finally deciphered the message, discovering the key after running through only 18 quadrillion of the 72 quadrillion possible keys!

Asymmetric encryption

- Better known as Public/Private Key
  - User X has pair of keys, one public & one private
  - To encrypt a message to X, Y uses X's public key
  - X will decrypt encrypted message using X's private key that "matches" X's public key
  - Uses modular arithmetic & elementary number theory
    - Based on the fact that it is extremely difficult to find the prime factors of large numbers

Used in
- Pretty Good Privacy (PGP)
- The Secure Sockets Layer (SSL)
- S/MIME, Secure Electronic Transactions (SET)
- Secure Shell (SSH)
- Included in Web browsers (Microsoft Internet Explorer)

Hash functions

- One-Way functions
  - Non-reversible "quick" encryption
  - Produces a fixed length value called a hash or message digest
  - Used to authenticate contents of a message
- Common message digest functions
  - MD4 and MD5: produce 128 bit hashes
  - SHA: produces 160 bit hashes

Secure channels

- Protecting communication between two users
  - E.g. peer-to-peer or client/server
- Two types:
  - Symmetric
    - Shared Secret Keys
    - Session Keys
  - Asymmetric, i.e. Public/Private Key

Uses of secure channels

- Secure Socket Layer (SSL):
  - Improves safety of Internet communications
  - Standard for encrypted client/server communic.
  - Protocol that runs on top of TCP/IP
  - Uses several security techniques e.g. public keys, symmetric keys, and certificates.
  - Web sites commonly use SSL to guard private information such as credit card numbers
- Transport Layer Security (TLS):
  - Protocol which ensures privacy between users
  - Successor to SSL

Symmetric: shared secret keys

- Generated once and secretly passed to the individuals
- This can be done in a number of ways:
  - Other methods e.g. by using public-keys
  - Telephone each other
  - Post it to each other
- Example system that uses this
  - Kerberos (http://web.mit.edu/kerberos/)

Symmetric: session keys

- Dynamically created at run time
- Can be done in two ways:
  1. Using public-keys
     - Cumbersome
     - Poor performance
  2. Dynamically create using Diffie-Hellman key exchange (next slide)

Symmetric Diffie-Hellman key exchange

- Also called "exponential key agreement"
  - Developed in 1976,
  - Published as ground-breaking paper "New Directions in Cryptography"
- Allows 2 users to exchange a secret key over an insecure medium without any prior secrets:

1. Both pick 2 large numbers, n and g (public), subject to certain mathematical properties
2. Tim chooses secret large random number, x
3. Gareth chooses secret large random number, y
4. Tim computes $(g^x) \bmod n$ (*public:* virtually impossible to compute x from $g^x \bmod n$)
5. Gareth computes $(g^y) \bmod n$
6. *They exchange public keys* $(g^x) \bmod n$ *and* $(g^y) \bmod n$
7. Gareth computes $((g^x) \bmod n)^y \bmod n = g^{xy} \bmod n$
8. Then, Tim computes $((g^y) \bmod n)^x \bmod n = g^{xy} \bmod n$

Both now have the **shared secret key** $g^{xy} \bmod n$

Asymmetric public/private key pairs

- Scenario:
  - Tim sold Gareth a data projector for £750 in a chat room
  - Email was their only communication channel
  - Gareth sends Tim a message confirming that he will buy the projector for £750
- Two issues:
  1. Tim needs to be assured that Gareth cannot deny ever having sent the message (if he has second thoughts)
  2. Gareth needs to be assured that Tim will not change the sum of £750 specified in his message to something higher…

Asymmetric public/private key pairs

Using RSA keys:

1. Gareth encrypts the message using his private key
2. Gareth also encrypts the message (for privacy) using Tim's public key
3. Tim can first decrypt the message using his private key then he can use Gareth's public key to decrypt the original message from Gareth

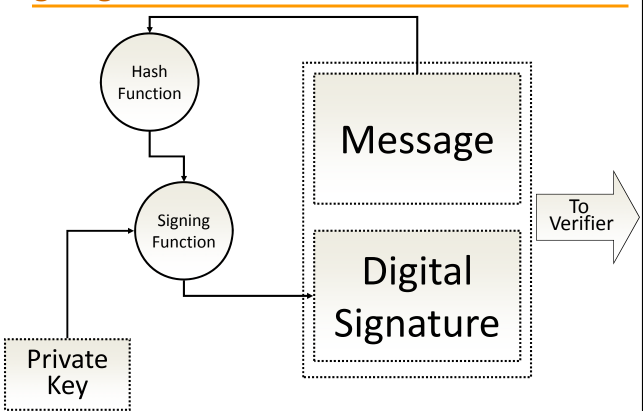
What can be inferred:

- If Tim accepts that Gareth's public key is in fact his then this must mean that the message came from Gareth
- Gareth knows that Tim received the message containing the original message because only Tim can open the message as he is the only person who has access to his private key

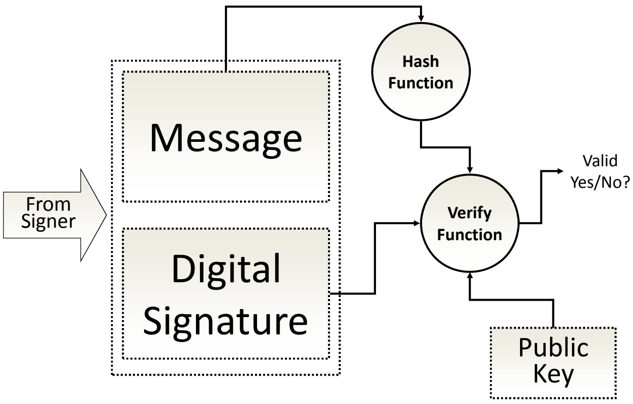
Digital signatures

- Asymmetric cryptosystems allow users to digitally sign messages
  - Allows a user to establish their authenticity
- A hash function is used to create & verify a digital signature
  - Converts the document into a hash
  - Concise and efficient for calculation

Signing



Verifying



Digital signature verification

- Verification indicates that:
  - The digital signature was created by the signer (i.e., the only person with access to private key)
  - The message was not altered since it was signed ("collisions" are mathematically improbable)
- There are different mathematical formulae and procedures, but all share overall operational pattern
- NB: signing does not encrypt a message!!
  - It is merely a method of verifying identity
  - However, encrypting a message with a private key also verifies a message
    - Much less efficient if this is its only purpose, though...

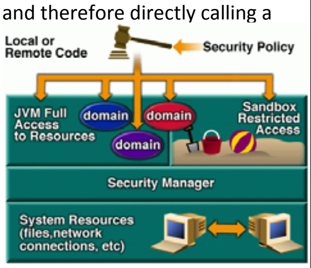
Secure mobile code

- How do you trust remote code to run locally?
  - Traditionally, you prevented malicious code from running on your system
- Use the "sandbox" security model
  - Downloadable program executed in a way that each of its instructions can be fully controlled

Sandboxing with Java

A Java sandbox prohibits:
- Reading or writing to the local disk
- Making network connections to any host, except the host that hosted the applet
- Creating a new process
- Loading a new dynamic library and therefore directly calling a native method

- Typically for applets
  - Also for P2P?
  - "Signed applets" trusted
    - Treated like local code



Further reading and resources

- Chaper 8 of textbook
- Article "New Directions in Cryptography", Whitfield Diffie and Martin E. Hellman.
  http://www.cs.berkeley.edu/~christos/classics/diffiehellman.pdf
- Kerberos Web page
  http://web.mit.edu/kerberos/