Grid Computing

Security - Cryptography

Paul A. Farrell
Fall 2006

The Grid: Core Technologies
Maozhen Li, Mark Baker
John Wiley & Sons; 2005, ISBN 0-470-09417-6

Cryptography and Network Security, Third Edition
William Stallings

Material from Dr John Watt, Grid Security Engineer
National e-Science Centre, University of Glasgow
Lecture slides by Amy Apon, Lawrie Brown,
University of Aksansas

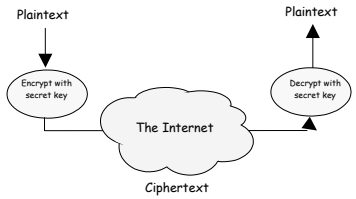
International Summer School on Grid Computing 2006
Ischia (Naples) July 9-21, 2006

Cryptography Goals

- Message confidentiality: only authorised recipient able to read contents of a mess
- Message integrity: detection of alteration during transmission
- Sender authentication: verify identity of sender, and that sender sent
- Sender non-repudiation: the sender cannot deny sending

Symmetric Cryptosystems

- A SYMMETRIC key system
  - Same key is required for encryption and decryption
  - Can also protect data integrity by attaching a checksum before encryption and verifying after decryption
  - But the key CANNOT be transmitted without compromising both sides' messages.

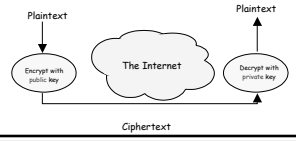


Example: Data Encryption Standard (DES)

- Two components
  - Algorithm using iterations of simple transformation
    - Uses transposition and substitution alternately
    - Algorithm publicly known
  - Key
    - Same key used to encrypt and decrypt
    - Anyone who has can decrypt
    - Must be kept secret

Asymmetric Cryptosystems

- Have two keys one for encryption and one for decryption.
- Make the encryption key PUBLIC for anyone to use, but keep decryption key PRIVATE
- Now anyone who wishes to send you a message uses your PUBLIC key to encrypt it, safe in the knowledge that the only person who can decrypt it is the holder of the PRIVATE key (i.e. you!)



Public-Key Characteristics

- Public-Key algorithms rely on two keys with the characteristics that it is:
  - computationally infeasible to find decryption key knowing only algorithm & encryption key
  - computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known
  - either of the two related keys can be used for encryption, with the other used for decryption (in some schemes)

Sending a message with double encryption

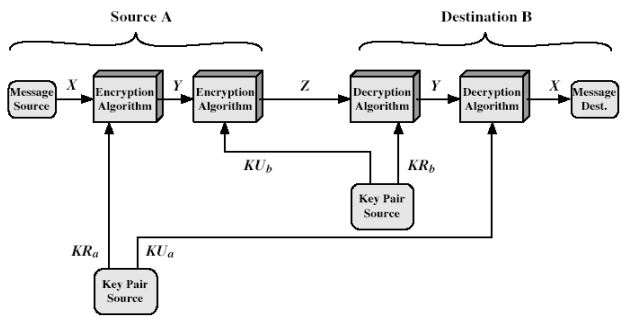


Figure 9.4 Public-Key Cryptosystem: Secrecy and Authentication

Public-Key Applications

- can classify uses into 3 categories:
  - **encryption/decryption** (provide secrecy)
  - **key exchange** (of secret session keys)
    - Session keys can be used in a session between a client and a server to encrypt network messages.
    - They expire at the end of the session – the short life span makes them difficult to break
  - **digital signatures** (provide authentication)

Example: RSA

- Named after Rivest, Shamir, Adleman
- Provides authentication as well
- No distinction between function of private and public keys
- Keys generated mathematically using prime numbers
  - based on exponentiation in a finite (Galois) field over integers modulo a prime
  - exponentiation takes $O((\log n)^3)$ operations (easy)
- Security security due to cost of factoring large numbers (256 to 1024 bit keys)
  - factorization takes $O(e^{\log n \log \log n})$ operations (hard)
- Private key must be kept securely
- In principle, could be broken by brute force but would take significant computation

SSL: An example of key exchange using public/private keys

- SSL (Secure Socket Layer) and TLS (Transport Layer Security) use public/private keys to exchange a secret key used during a session
- The SSL handshake consists of several steps, as follows:

Step 1: The client contacts the server and sends SSL version number, a random number X, and some additional information

Step 2: The server sends the client the SSL version number, random number Y, and its public key (packaged into a certificate)

Step 3: The client verifies that the server is who is says it is by examining the certificate (more on this in a bit)

Step 4: The client creates a "premaster secret" using X,

SSL Handshake

Step 5: If the server has requested authentication, the client sends its own certificate and the premaster secret to the server

Step 6: The server authenticates the client by examining the client's certificate, uses its private key to decrypt the premaster secret, then uses it to generate the master secret. The client also generates the master secret.

Step 7: Both the client and the server use the master secret to generate the session secret key

Steps 8 (9): The client (server) sends a message to the server (client) telling it that it will use the secret key. It sends a second message encrypted with the secret key.

SSL Handshake

Step 10: The handshake is complete and the SSL session has begun.

Read http://developer.netscape.com/docs/manuals/security/sslin/index.html for a description about the SSL handshake.

Digital Signatures

- Integrity guaranteed in public-key systems by using digital signatures
  - a sequence of bits conforming to one of a number of standards
- To verify identity of originator
  - Send message in plain text AND encrypted with PRIVATE key
  - Recipient decrypts with PUBLIC key of sender and compares

Digital Signatures - Details

Use a combination of a message digest (hash) and public key encryption to be able to guarantee that a message was sent by who claimed to send it

Step 1: I create a message digest of the message

Step 2: encrypt the message digest with my private key (that only I know). This is my digital signature

Step 3: Append the message with my digital signature and send the message in the open network

Step 4: Anyone with my public key can decrypt the signature, apply the hash function to get the hash, then compare the hash with the decrypted signature to see if they are the same

See http://www.youdzone.com/signature.html

How secure is public key encryption?

- like private key schemes brute force **exhaustive search** attack is always theoretically possible
- but keys used are too large (>512bits)
- security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- more generally the **hard** problem is known, its just made too hard to do in practise
- requires the use of **very large numbers**
- hence is **slow** compared to private key schemes

Distribution of Public Keys

- Can be considered as using one of:
  - Public announcement
  - Publicly available directory
  - Public-key authority
  - Public-key certificates

Public Announcement – a **bad** distribution technique!

- users distribute public keys to recipients or broadcast to community at large
  - eg. append PGP keys to email messages or post to news groups or email list
- major weakness is forgery
  - **Anyone can create a key claiming to be someone else and broadcast it**
  - Until forgery is discovered can masquerade as claimed user

Recall the Digital Signature Application

- What if my enemy Doug wants to fool you into thinking that I sent a message?
- Doug might send you a public key that he claims is mine (and keep the matching private key to himself).
- If you believe that the public key Doug sent is mine, then Doug could sign a message with the private key and pretend to be me.
- How can you be sure that the public key you receive is mine?

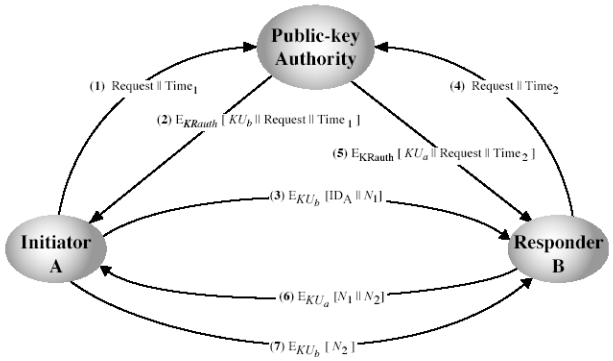
Public Key Distribution Using a Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
  - contains {name,public-key} entries
  - participants register securely with directory
  - participants can replace key at any time
  - directory is periodically published
  - directory can be accessed electronically
- still vulnerable to tampering or forgery

Public Key Distribution Using a Public-Key Authority

- improve security by tightening control over distribution of keys from directory
- has properties of directory
- and requires users to know public key for the directory
- then users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed

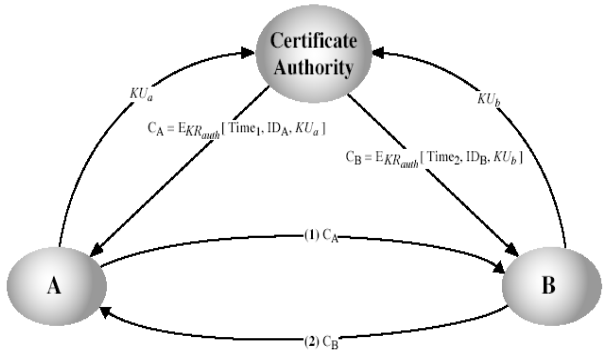
Public-Key Authority



Public Key Distribution Using Public-Key Certificates

- certificates allow key exchange without real-time access to public-key authority
- a certificate binds **identity** to **public key**
  - usually with other info such as period of validity, rights of use etc
- with all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
- can be verified by anyone who knows the public-key authority's public-key

Public-Key Certificates



Public Key Certificates

- IF you trust the Certificate Authority
- AND you are confident that the KU_{auth} key that you have is really the public key of the Certificate Authority
- THEN, you can decrypt the certificate with confidence to obtain the public key of the sender

Read http://docs.sun.com/source/816-6154-10/contents.htm section starting with Certificates and Authentication

Public Key Distribution Using Public-Key Certificates

- The problem is really an authentication problem – do you believe that the sender of the certificate is who it says it is?

Next, a short diversion on **authentication** (section 14.1 and 14.2 from [Nutt]) and then we will talk about X.509, a standard for public-key certificates.