CSC 7700: Scientific Computing

Module A: Basic Skills

Lecture 4: Advanced **S**ecure **Sh**ell

Dr Frank Löffler

October 19th 2010

Overview

History

Authentication

Public-Key Cryptography

Uses of SSH

SSH implementation examples

SSH usage examples

Overview

Overview

- **S**ecure **Sh**ell
- Network protocol
- Reserved network port: 22
- Secure channel
- Public-key cryptography
- Authentication
- One of major uses: access to remote shell accounts
- Can tunnel shell-unrelated network traffic

History

Early history

- 1995: Tatu Ylönen, a researcher at Helsinki University of Technology, Finland, designed the first version of the SSH protocol
- Reason: password-sniffing attack at his university network
- Goal: replace the earlier rlogin, TELNET and rsh protocols
- Implementation was released as freeware in July 1995
- End of 1995: about 20.000 users in 50 countries

SSH Forks

- December 1995: Ylönen founded SSH Communications Security
- Releases evolved into increasingly proprietary software
- As of 2000: about 2 million users of SSH
- 1999: Björn Grönvall forked old, free version of SSH
- Shortly thereafter, OpenBSD forked Grönvall's code, creating OpenSSH
- As of 2005: OpenSSH was the single most popular SSH implementation

Recent History

- 2006: Version 2 of SSH protocol: SSH-2
- Incompatible with version 1
- Both security and feature improvements
- OpenSSH implements both versions
- By default, version 1 is usually disabled

Internet standards

- RFC 4250, The Secure Shell (SSH) Protocol Assigned Numbers
- RFC 4251, The Secure Shell (SSH) Protocol Architecture
- RFC 4252, The Secure Shell (SSH) Authentication Protocol
- RFC 4253, The Secure Shell (SSH) Transport Layer Protocol
- RFC 4254, The Secure Shell (SSH) Connection Protocol
- RFC 4255, Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints
- RFC 4256, Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)
- RFC 4335, The Secure Shell (SSH) Session Channel Break Extension
- RFC 4344, The Secure Shell (SSH) Transport Layer Encryption Modes
- RFC 4345, Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol
- RFC 4419, Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol (March 2006)
- RFC 4432, RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol (March 2006)
- RFC 4462, Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol (May 2006)
- RFC 4716, The Secure Shell (SSH) Public Key File Format (November 2006)
- RFC 5656, Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer (December 2009)

Authentication

Authentication

- Establishing or confirming something (or someone) as authentic
- Not to be confused with Authorization
  - Authentication must precede authorization
- Short notations: A1/AuthN (authentication), A2/AuthZ (authorization)
- Impossible to prove the identity of a computer user with absolute certainty

Mechanism examples:

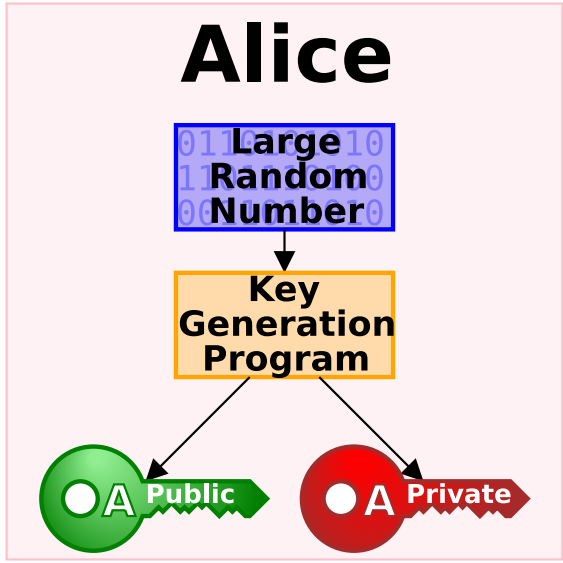
- Shared secret (e.g. passwords, key cards)
- Public+secret key pairs
- Certificates

Public-Key Cryptography

Public-Key Cryptography

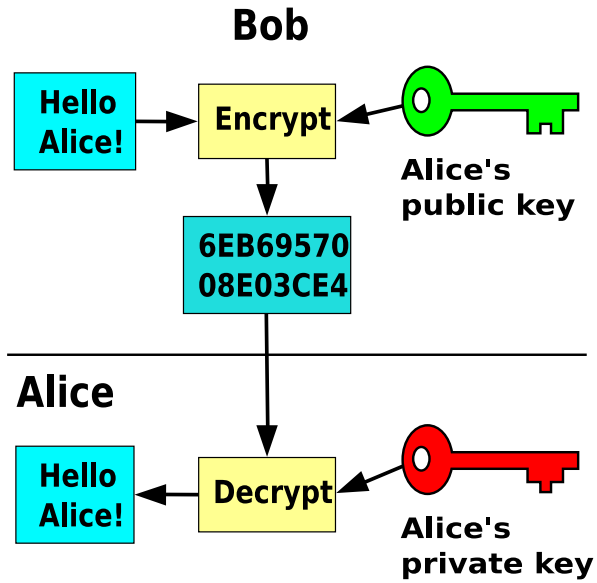
- Use of asymmetric key algorithms
- No need for secure initial secret key exchange (unlike for symmetric key algorithms)
- Creation of mathematically related key pair:
  - secret, private key
  - published, public key
- Can be used for protection of
  - confidentially and integrity by public key encryption
  - authenticity by digital signatures

Asymmetric key algorithm



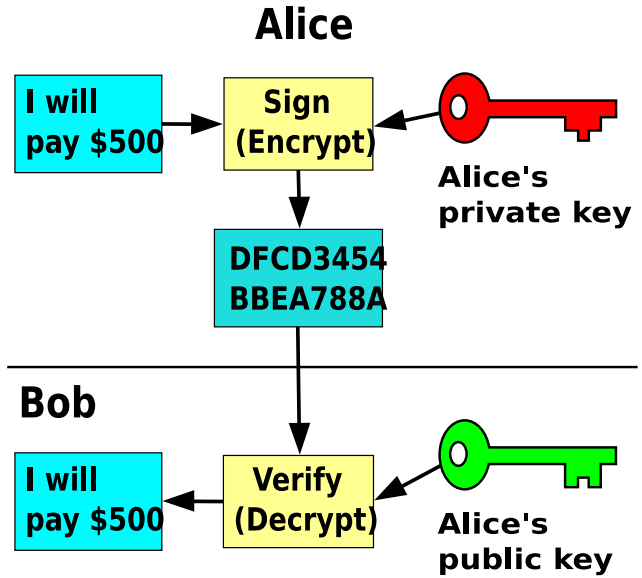
An unpredictable (typically large and random) number is used to begin generation of an acceptable pair of keys suitable for use by an asymmetric key algorithm.

Asymmetric key algorithm



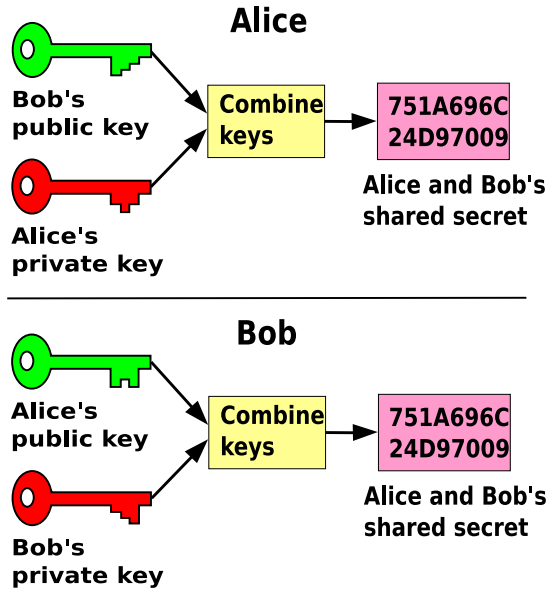
Anyone can encrypt messages using the public key, but only the holder of the paired private key can decrypt. Security depends on the secrecy of that private key.

Asymmetric key algorithm



The private key can be used to sign a message; but anyone can check the signature using the public key. Validity depends on private key security.

Asymmetric key algorithm



In the Diffie–Hellman key exchange scheme (which is used in SSH), each party generates a public/private key pair and distributes the public key. After obtaining an authentic copy of each other's public keys, Alice and Bob can compute a shared secret offline. The shared secret can be used as the key for a symmetric cipher.

Asymmetric key algorithm

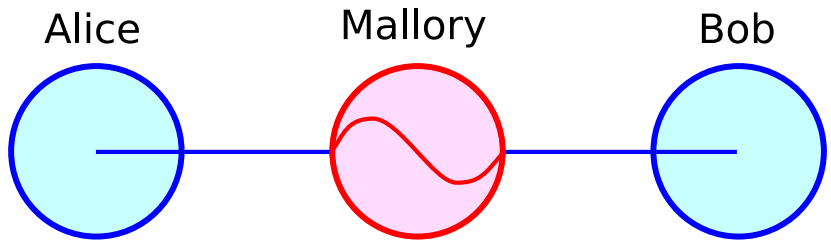
Central problem:

- Confidence that a public key is correct, belongs to the person or entity claimed, and has not been tampered with or replaced by a malicious third party

Usual approaches:

- "Web of trust" like used in e.g. PGP (not used for SSH)
- Certificates and certificate authorities

Man in the middle attack



1. Alice sends a message to Bob, which is intercepted by Mallory:
   Alice *"Hi Bob, it's Alice. Give me your key"* → **Mallory Bob**
2. M. relays message to Bob, who cannot tell its not from Alice:
   Alice Mallory "Hi Bob, it's Alice. Give me your key." → **Bob**
3. Bob responds with his encryption key:
   Alice Mallory ← [Bob's key] **Bob**
4. Mallory replaces Bob's key with own, and relays:
   Alice ← [Mallory's key] **Mallory Bob**
5. Alice encrypts a message with that key:
   Alice "Buy stock A." [Mallory's key] → **Mallory Bob**
6. Mallory decrypts, reads, modifies, re-encrypts and forwards:
   Alice Mallory "Sell everything." [Bob's key] → **Bob**

Uses of SSH

Uses of SSH

- Login to a shell on a remote host (replacing Telnet and rlogin)
- Executing a single command on a remote host (replacing rsh)
- Secure file transfer
- In combination with rsync to back up, copy and mirror files efficiently and securely
- Forwarding or tunneling a network port
- Using as a full-fledged encrypted VPN.
- Securely forwarding X from a remote host
- Browsing the web through an encrypted proxy connection (SOCKS)
- Securely mounting a directory on a remote server as a filesystem on a local computer (SSHFS)

File transfer protocols using SSH

- Secure copy (SCP): rcp replacement
  - Combination of RCP and SSH
  - Only transfer of files
  - Usually command line clients
- SSH File Transfer Protocol (SFTP)
  - secure alternative to FTP
  - resuming, directory listings, remote file removal
  - Gui clients feasable

Above can be used by either dedicated clients, or by e.g. SSHFS.

SSH implementation examples

SSH Server implementation examples

- Some commercial implementations
- OpenSSH
  - By far the most popular implementation
  - Available on almost every platform, from PC over Mac, Amiga, to iPhones
  - Feature-rich
- CopSSH
  - SSH server package for Microsoft Windows
  - uses OpenSSH, OpenSSL, Cygwin and GNU tools
- Dropbear
  - lightweight SSH-2 implementation
  - designed for environments with low memory and processor resources
  - no SSH-1 or SFTP support

SSH client implementation examples

- Some comercial implementations
- OpenSSH
  - Often automatically included in OS distributions
- PuTTY
  - Originally a Microsoft Windows GUI client
  - Now ported to a number of other OSs as well
- Dropbear
- CopSSH
- ...

SSH usage examples

Basic login commands

Simplest command:

```
ssh hostname.domain
```

Forwarding X:

```
ssh -Y user@hostname.domain
```

Executing single command:

```
ssh user@hostname.domain ls
```

Using a “trampoline”:

```
ssh -t user@hostname1.domain user@hostname2.domain
```

Using above to open remote X terminal:

```
ssh -Y user@hostname1.domain xterm
```

Key management

(Some) Authentication options

- Passwords
- Key pairs
- Certificates

Key pairs:

- Public/Secret, personal key pair
- Used instead of password
- Secret key usually passphrase-protected
- Two (three) key types: rsa, dsa (rsa1 for ssh-1)
- Usually stored in `$HOME/.ssh/id_dsa[.pub]`, `$HOME/.ssh/id_rsa[.pub]`

Key management examples

Tool to create and change keys: `ssh-keygen`, Usage examples:
Creating new DSA Ssh key:

```
ssh-keygen -t dsa
```

Changing the passphrase of an existing key:

```
ssh-keygen -t dsa -p
```

Authorizing key as replacement for password by adding public key to remote file

```
cat ~/.ssh/id_dsa.pub | ssh user@machine.domain \
  'mkdir -p .ssh; cat >> .ssh/authorized_keys'
```

SSH Agents

Private SSH keys should have passphrase set (for most uses). But:

- Cumbersome to always enter passphrases
- Even more problematic when "hopping" from host to host

Solution: SSH-Agents

- Will ask for password once, (securely) store it and reuse later
- Configurable lifetime of stored passwords
- Tools: `ssh-agent`, `keychain`

SSH can forward agent information:

```
$ cat .ssh/config
Host *
  ForwardAgent yes
```

Summary

Summary

Secure Shell (SSH)

- Provides secure network channel for variety of uses, e.g:
  - Remote login shell
  - Secure file transfer
  - Network traffic tunneling

Best usage practices:

- Use ssh keys instead of passwords
- Set passphrase for ssh keys
- Use ssh agent to ease work with ssh keys