Week 06M: SSH Setup and Use in Applications

Thuy Vu

Reminders

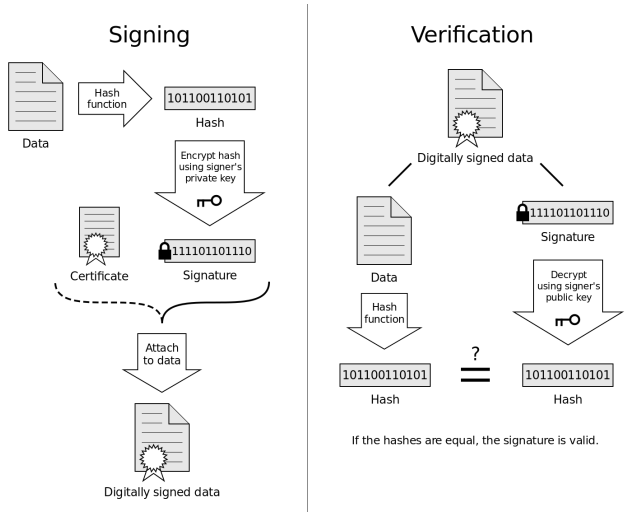
- Assignment 5
  - `web.cs.ucla.edu/classes/winter17/cs35L/assign/assign5.html`
  - Time due 23:55 this Friday, February 17 (in 3 days!)
- Assignment 10 Sign-up
  - `https://goo.gl/794MQM`

Communication Over the Internet

1. Communication Over the Internet: Confidentiality (message secrecy); Data Integrity (message consistency); Authentication (identity confirmation); Authorization (access rights to resources)
2. SSH (Secure Shell) remotely access with authentication and encryption
   - `telnet` has no security measures
   - `ssh name@host` → verify `host`, public key in `~/.ssh/known_hosts`
   - future host validation: use this public key
3. Encryption Types – two key encryption methods
   - Symmetric – one key for both encrypt and decrypt
   - Asymmetric – public key to encrypt; private key to decrypt; in-pair
4. Client Authentication – two authentication methods
   - password-based – usename ↔ password
   - key-based – by public and private keys
     1. client to generate key-pair
     2. public key to copy to `~/.ssh/authorized_keys`
     3. server authenticates client if client shows to obtain private key
     4. passphrase is to protect private key, but "annoying" → `ssh-agent`
     5. `ssh-add` prompts for passphrase once and adds to `ssh-agent`'s list
5. X Window System – Windowing system for GUIs on UNIX
   - X Session Forwarding – program run on a computer yet display on another

Communication Over the Internet

1. Digital Signature – a electronic stamp/seal appended to a document to ensure data integrity during transmission



2. from sender
   1. generate a *message digest* using hashing algorithms
   2. encrypt that *message digest* using sender's private key $\rightarrow$ *digital signature*
   3. attach the *digital signature* to the message and send to receiver
3. from receiver
   1. decrypt the *digital signature* using sender's public key
   2. generate a *message digest* using hashing algorithms
   3. compare two *message digests*
4. detached signature
   1. detached signature to be stored transmitted separately from the message
   2. commonly used to validate software distributed in compressed tar files
   3. you can't sign such a file internally without altering its contents

The Lab – SSH Setup

1. install openssh
   - `dpkg --get-selections | grep openssh`
   - `sudo apt-get install openssh-server`
   - `sudo apt-get install openssh-client`
2. generate public and private keys (`ssh-keygen`)
3. find IP (`ifconfig`) and try communicate via IP (`ping 10.97.85.xyz`)

client – "I need an account"

1. ...waiting
2. copy the public key (`~/.ssh/id_rsa.pub`)

   over the server (`~/.ssh/authorized_keys`)

   and add private key
   - `ssh-copy-id -i <username>@<server_ip>`
   - `ssh-add`
3. SSH to server
   - `ssh <username>@<server_ip>`
   - `ssh -X <username>@<server_ip>`
4. do something: e.g. `xterm`; `gedit`; `firefox`;
5. `chmod o-rwx "$HOME"`

server – "wait a second"

1. `sudo useradd -d /home/<homedir> -m <username>`
2. `sudo passwd <username>`
3. create .ssh directory for new user
   - `cd /home/<homedir>`
   - `sudo mkdir .ssh`
4. change the ownership and permission
   - `sudo chown -R <username> .ssh`
   - `sudo chmod 700 .ssh`
5. tell the client "done!"
6. disable password-based authentication
   - `emacs /etc/ssh/sshd_config`
   - `PasswordAuthentication` → `no`

The Homework – SSH Use in Applications

1. answer 2 questions in the file `hw.txt`
2. generate key-pair with the GNU Privacy Guard `$gpg --gen-key`
3. export public key in ASCII format

4. make a tarball of the above files + `log.txt` using `$tar -cf hw.tar <files>` then zip it with gzip to produce hw.tar.gz, using `$gzip hw.tar`
5. use the private key you created to make a detached clear signature hw.tar.gz.sig for hw.tar.gz

6. use given commands to verify signature and file formatting