EXTRA: Encryption

Dylan Schwilk

March 28, 2022

Encryption

Requirements for encryption systems

- Encryption/decryption
- Authentication
- Check integrity
- Disallow repudiation

Three basic cryptography schemes

- Secret key (symmetric)
- Public key (asymmetric)
- Hash functions

Symmetric vs asymmetric encryption

symmetric shared key

asymmetric requires two keys, one "public" and one "private"

[

Notes]Notes B_NOTE

- Symmetrical encryption is a type of encryption where one key can be used to encrypt messages to the opposite party, and also to decrypt the messages received from the other participant. This means that anyone who holds the key can encrypt and decrypt messages to anyone else holding the key.
- "Public" and "private" are just words. Really, the relationship is still symmetrical: one key can only decrypt what the other has encrypted

Basics of encryption (symmetric)

Encryption Plaintext + Key -> Ciphertext

Decryption Cyphertext + Key -> Plaintext

Asymmetric encryption

Requires a key pair that are mathematically related.

- Plaintext + Public Key -> Cyphertext
- Cyphertext + Private Key -> Plaintext

Hash functions

Useful for checking integrity

- plaintext -> Hash Function -> plaintext of fixed length

[

Lecture notes]Lecture notes B_NOTE

- One way function that is deterministic. Infeasible to generate message that has a particular hash or find to messages that produce same hash. Small change in message shgould produce large change in hash so that there is no correlation between message similarity and hash similarity.
- You've all seen hashes such as the identity of an individual commit in git.
- Some hash functions MD5 (used in checksums) , SHA-1 and SH-256 (git commits).

SSH

SSH: Secure SHell

A network protocol that allows two computers to communicate securely. Used for:

- Remote command line access to another machine (interactive or automatic!)
- Remote GUI access to a remote machine under some conditions (eg X11 forwarding)
- Port forwarding. Mapping client ports to server's ports.
- Tunneling (same idea as a VPN)

SSH steps

Connection Uses asymmetrical keys

Session encryption uses shared key behind the scenes which is calculated by both machines

Authentication either password (bad) or asymmetric keys (good)

Hashing used to check data integrity.

[

Notes]Notes B_note

- You only need to worry about keys for the authentication step.

Password based login

```
username@machine:~$ ssh username@185.52.53.222 usernamep@185'.52.53.222s password:

The authenticity of host '185.52.53.222 (185.52.53.222)' can't be established. ECDSA ↩
    key fingerprint is SHA256:9lyrpzo5Yo1EQAS2QeHy9xKceHFH8F8W6kp7EX2O3Ps.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added ' 185.52.53.222' (ECDSA) to the list of known hosts.

username@host:~$
```

[

Notes]Notes B_note

- You can also specify the port to connect to (default is 22)

Key based login

```
[schwilk@algiers ~]$ ssh pearse
Welcome to Ubuntu 21.04 (GNU/Linux 5.11.0-49-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

Last login: Tue Mar 22 09:09:57 2022 from 10.135.27.100
schwilk@pearse:~$
```

[

Notes]Notes B_note

- You can specify a private key to use (rather than default for the current user on the client machine).

SSH key algorithms

DSA Don't use.

RSA OK but use a large key (4096 bytes)

ECDSA Not recommended

ED25519 Current best practice, pretty well supported.

[

notes]notes B_note

- RSA is well supported but is vulnerable to some brute force cracking so long keys are needed – it is estimated the 4096 length keys should be uncrackable for a couple of decades.
- ECDSA uses shorter keys but the Snowden revelations showed that the US NSA chose the curves and it is strongly suspected that they have included some backdoor into the algorithm.
- Both RSA and ECDSA are susceptible to poor random number generators. How Playstations were hacked in 2010.

Creating an SSH key pair

Do this on the client machine.

```
ssh-keygen -t ed25519 -C "username@hostname"
```

[

Notes]Notes B_note

- Create a key for each client machine you use!
- You can choose to protect the private key with a password. I do for interactive keys but then use ssh-agent to store my password for a client login session.
- This example uses the default names for the key files created. See the -f option.

Copy public key to the server

Public key goes in `/~/.ssh/authorized_keys` on the SERVER

github use the web interface

other servers you might need to login locally, or copy over a password authenticated ssh session if allowed.

Important SSH files

In addition to private and public keys all in `/~/.ssh`

- `authorized_keys`
- `config`
- `known_hosts`

Example `/~/.ssh/config`

```
Host *
    Protocol 2
    ServerAliveInterval 100
# Added manually for git-annex
Host git-annex-havana.schwilk.org-schwilk
  Hostname havana.schwilk.org
  IdentityFile ~/.ssh/git-annex/key.git-annex-local
  IdentitiesOnly yes
  StrictHostKeyChecking yes
# quanah:
Host quanah
  Hostname quanah.hpcc.ttu.edu
  User dschwilk
```

ssh-agent

Useful to avoid entering password for private key many times per session.

PGP / GPG

- PGP is "pretty good privacy".
- General asymmetrical key encryption for storing stuff on your PC, sending emails, signing emails, etc.
- Plugins for most email clients make it easy.
- `pass` is the password manager for linux. I prefer it to more gui methods.

GnuPG

- Is the open source program that encrypts and decrypts. Executable is `gpg`
- OpenPGP is another implementation.

Basic use:

I can encrypt to Natasja using her public key. Only she can decrypt. She can "sign" a document using her private key, anyone with public key can be assured only she could have signed it.

Example: encrypting

```
schwilk~/temp$ echo "Plaintext" > plain.txt
schwilk~/temp$ gpg -e plain.txt
You did not specify a user ID. (you may use "-r")

Current recipients:

Enter the user ID. End with an empty line: Dylan

Current recipients:
rsa4096/4D4C7419862599EF 2013-06-19 "Dylan W Schwilk <dylan@schwilk.org>"

Enter the user ID. End with an empty line:
```

Example: encrypting

```
schwilk~/temp$ cat plain.txt.gpg

 MLt  %54a }s ,0     tx  1   B     ]+2 s
                                              [Ga^    p40@f'´
                                                      "S$3 y  <  yX : lDlç~ 1"↩
                                                            qYg'
       sO :
wTrx_   U2qb* k7& < V_i   -4>  jN
                                  {  q
&   Vp  .    @GTW    `<9>   cSqSOx }   x8M  .   R!   KDke_ e,(J
                                    WjhzJ-    YrD7IA1" ` J  | f  (  ln   .46r  %  dGn |j$↩
                                        F   > CZYÊb    O
            G}T<      Oi7G"Ţ   c8t. WUf   ;/ "
#J    EZ .5]    Sma   \|8  I&   ;  rsZschwilk
```

Example: decrypting

```
gpg -d plain.txt.gpg
gpg: encrypted with 4096-bit RSA key, ID 4D4C7419862599EF, created 2013-06-19
     "Dylan W Schwilk <dylan@schwilk.org>"
Plaintext
```

Web of trust

How do you now that recipients public key is really the correct one?

Key signing

and web of trust