SSH Unveiled?

Andrea Cucca

October 12, 2012

Table of contents

1. Life without SSH
   - History of SSH
2. Cryptography
   - Classical Cryptography
   - Modern Cryptography
   - RSA Algorithm
3. Life with SSH
   - SSH Architecture
   - Basic SSH use
   - Advanced use
4. Conclusions

SSH

Life without SSH

A short Story

- In 1995 Tatu Ylönen, a researcher at Helsinki University of Technology, had his account hacked by a password-sniffing attack.
- He hence decided to develop a secure remote connection protocol with the same features of rlogin, telnet and rsh. He called it Secure SHell.
- The first version of the protocol (now called SSH-1) was released in july of the same year as free software and by the end of the year there were at least 20.000 users over the world.
- He founded the SSH Communication Security company in December 1995, starting a commercial developement of the software.

A short Story

- In 1999 Björn Grönvall went back to the first free release in order to develop a open source version of the protocol: OSSH.
- This was the base for a fork, done by OpenBSD developers, that implemented their OpenSSH version in OpenBSD 2.6 and ported the code to many other platforms.
- Since 1998 there is a SSH-2 version that replaces SSH-1 version that had some vulnerabilities.
- In the following we concentrate on OpenSSH, now at version 6.1.

Cryptography

Cryptography

The problem of the Cryptography

- Exchange confidential information
- Guarantee and authenticate sender and information
- Encryption $\Rightarrow$ Start from a plain text and produce unintelligible sequence
- Decryption $\Rightarrow$ go back to plaintext

History of Cryptography

- We can define three different cryptographies:
  - Transposition cipher
    - welcome to 411 → cmolewe ot 141
    - only really useful for gaming
  - Coded words
    - Play a lullaby → Attack at sunset
    - Implies a list of conversion to be owned by sender and recipient

History of Cryptography

- We can define three different cryptographies:
  - Transposition cipher
    - welcome to 411 → cmolewe ot 141
    - only really useful for gaming
  - Coded words
    - Play a lullaby → Attack at sunset
    - Implies a list of conversion to be owned by sender and recipient
  - Substitution cipher
    - welcome to 411 → xfmdpnf up 522
    - the most efficient of the simple cryptographies

History of Cryptography

- We can define three different cryptographies:
    - Transposition cipher
        - welcome to 411 → cmolewe ot 141
        - only really useful for gaming
    - Coded words
        - Play a lullaby → Attack at sunset
        - Implies a list of conversion to be owned by sender and recipient
    - Substitution cipher
        - welcome to 411 → xfmdpnf up 522
        - the most efficient of the simple cryptographies
- Caesar's cipher (substituting every letter with the third following character)
- Carved stone ciphertext dating 1900 BC found in Egypt
- Indian Kama Sutra recommend use of cryptography to lovers who need secrecy

History of Cryptography

- We can define three different cryptographies:
  - Transposition cipher
    - welcome to 411 → cmolewe ot 141
    - only really useful for gaming
  - Coded words
    - Play a lullaby → Attack at sunset
    - Implies a list of conversion to be owned by sender and recipient
  - Substitution cipher
    - welcome to 411 → xfmdpnf up 522
    - the most efficient of the simple cryptographies
- Caesar's cipher (substituting every letter with the third following character)
- Carved stone ciphertext dating 1900 BC found in Egypt
- Indian Kama Sutra recommend use of cryptography to lovers who need secrecy

History of Cryptography

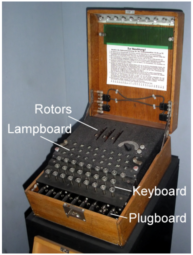
- In 1467 Leon Battista Alberti discovered polyalphabetic cipher and built the first automatic device (a composed wheel with 2 alphabets)

A B C D E F G I L M N O P Q R S T V X Z 1 2 3 4
g k l n p r t v z & x y s o m q h i f d b a c e

- *Tabula recta* (Johannes Trithemius, 1462 - 1516)
- Vigenère cipher (Blaise de Vigenère, 1523 - 1596)

History of Cryptography

- Since the discovery of polyalphabetic cipher cryptography didn't change a lot. The whole research was focused on the construction of more sophisticated cryptographic devices.
- In 1917 Gilbert Vernam invented the first teleprinter cipher (a previously prepared key, kept on paper tape, is combined character by character with the plaintext message to produce the cyphertext)
- During World War II mechanical and electromechanical cipher machines were in wide use.
- After WWII cryptographical techinques were shifted towards more mathematical methods (Claude Shannon).
- Based on mathematics and computational technology with computational hardness assumption.



- In any case cryptography security depends on key security

Modern Cryptography

Note that:

- Unbreakable cipher do exist! (Mathematically proved by Shannon)
- Such a system is secure even against unlimited computing power.
- Not breakable even by quantum computing approaches.
- One-time pad
- Not easy and quite expensive to generate and setup.

We need other methods!

Modern Cryptography

Note that:

- Unbreakable cipher do exist! (Mathematically proved by Shannon)
- Such a system is secure even against unlimited computing power.
- Not breakable even by quantum computing approaches.
- One-time pad
- Not easy and quite expensive to generate and setup.

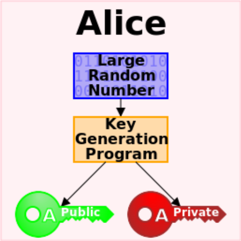
We need other methods!

Public-Key Cryptography

- First appeared in 1976 in a seminal paper of Whitfield Diffie and Matrin Hellmann (but also Ellis, Cocks, Williamson)
- Asymmetric key model, one public and one private key, linked together by some mathematical relation
- The secrecy is guaranteed even without sharing a key!
- The model is based on computational complexity, using mathematical problems that make easy to generate the key couple but difficult to reconstruct it without knowing both parts
- RSA (integer factorization problem), DSA (discrete logarithm problem), EDSA (elliptic curve DSA)
- The most revolutionary new concept in the cryptography since polyalphabetic substitution emerged in the Renaissance

Asymmetric-Key mechanism

- Because of its construction, a single key is useless without the other
- One key encrypts (the public key), the other (the private one) decrypts the message
- If Alice and Bob want to receive encrypted messages they just need to distribute their public key
- The procedure also authenticates their messages



RSA Algorithm - A mail analogy

- Bob wants to send a secret message to Alice

RSA Algorithm - A mail analogy

- Bob wants to send a secret message to Alice
- Alice sends her 🔓 to Bob

RSA Algorithm - A mail analogy

- Bob wants to send a secret message to Alice
- Alice sends her to Bob
- Bob locks a with Alice’s pad and sends it back

RSA Algorithm - A mail analogy

- Bob wants to send a secret message to Alice
- Alice sends her to Bob
- Bob locks a with Alice's pad and sends it back
- Alice uses her private to open the box

Note that:

- There has been no exchange of keys between Alice and Bob

RSA Algorithm - A mail analogy

- Bob wants to send a secret message to Alice
- Alice sends her to Bob
- Bob locks a with Alice's pad and sends it back
- Alice uses her private to open the box

Note that:

- There has been no exchange of keys between Alice and Bob
- Alice can reuse her pad at will as far as she keeps her key in a safe place

RSA Algorithm - A mail analogy

- Bob wants to send a secret message to Alice
- Alice sends her to Bob
- Bob locks a with Alice's pad and sends it back
- Alice uses her private to open the box

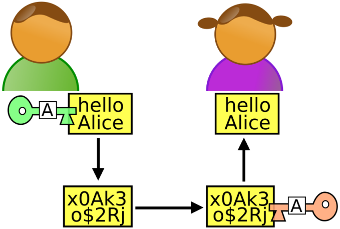
Note that:

- There has been no exchange of keys between Alice and Bob
- Alice can reuse her pad at will as far as she keeps her key in a safe place

RSA Algorithm - The theory

- RSA (Rivest, Shamir, Adleman) is the first (1977) asymmetric key algorithm published
- Based on the difficulty of integer factorisation for very large numbers
- Given two random integer numbers p and q and their product (*modulus*) $n = p \cdot q$
- Compute $\varphi(n) = (p-1)(q-1)$, where φ is Euler's Totient function that has the properties: $\varphi(p \cdot q) = \varphi(p) \cdot \varphi(q)$ and $\varphi(n) = n - 1$
- Then find 2 numbers e and d such as $e \cdot d \equiv 1\, mod[\varphi(n)]$
- **n** and **e** are public (they are the public key) while **d** (combined with n) is the secret private (decryption) key

RSA Algorithm - How-To



- Alice runs her algorithm and gives Bob her n and e.
- Bob wants to send a message **M** to Alice
- Using a standard method, he converts **M** in a integer m and computes $c = m^e mod[n]$.
- c is the cyphered, secret message that Bob and Alice will share
- Alice decrypts c by the formula $m = c^d mod[n]$ and recovers **M**

RSA Algorithm - A real example with numbers

- Let's choose 2 prime numbers

$(p, q) \quad ; \quad n = p \cdot q$

$$p = 61 \text{ and } q = 53$$
$$61 \cdot 53 = 3233$$

- Let's compute the totient function

RSA Algorithm - A real example with numbers

- Let's choose 2 prime numbers

$(p, q) \quad ; \quad n = p \cdot q$

$p = 61$ and $q = 53$

$61 \cdot 53 = 3233$

- Let's compute the totient function

$\varphi(n) = (p-1)(q-1)$

$\varphi(n) = (61-1)(53-1) =$
3120

- Find e and d (use random generation)

RSA Algorithm - A real example with numbers

- Let's choose 2 prime numbers

$$(p, q) \quad ; \quad n = p \cdot q$$

$$p = 61 \text{ and } q = 53$$
$$61 \cdot 53 = 3233$$

- Let's compute the totient function

$$\varphi(n) = (p-1)(q-1)$$

$$\varphi(n) = (61-1)(53-1) = 3120$$

- Find e and d (use random generation)

$$e \cdot d \equiv 1\, mod[\varphi(n)] \; ; \; 1 < e < \varphi(n)$$

$$e = 17 \; ; \; 17 \cdot d \equiv 1\, mod[3120]$$

- Invert the relation to find d

RSA Algorithm - A real example with numbers

- Let's choose 2 prime numbers

$$(p, q) \quad ; \quad n = p \cdot q$$

$$p = 61 \text{ and } q = 53$$
$$61 \cdot 53 = 3233$$

- Let's compute the totient function

$$\varphi(n) = (p-1)(q-1)$$

$$\varphi(n) = (61-1)(53-1) = 3120$$

- Find e and d (use random generation)

$$e \cdot d \equiv 1\, mod[\varphi(n)] \; ; \quad 1 < e < \varphi(n)$$

$$e = 17 \; ; \; 17 \cdot d \equiv 1\, mod[3120]$$

- Invert the relation to find d

$$d = e^{-1}\, mod \varphi[n]$$

$$d = 2753$$

RSA Algorithm - A real example with numbers

- Let's choose 2 prime numbers

$(p, q) \quad ; \quad n = p \cdot q$

$p = 61$ and $q = 53$
$61 \cdot 53 = 3233$

- Let's compute the totient function

$\varphi(n) = (p-1)(q-1)$

$\varphi(n) = (61-1)(53-1) =$
3120

- Find e and d (use random generation)

$e \cdot d \equiv 1 \, mod[\varphi(n)]$;
$1 < e < \varphi(n)$

$e = 17 \; ; \; 17 \cdot d \equiv 1 \, mod[3120]$

- Invert the relation to find d
$d = e^{-1} \, mod \varphi[n]$

$d = 2753$

RSA Algorithm - A real example with numbers

- The public key is given by the 2 numbers n = 3233 and e = 17
- The encryption function is $m^{e} mod[n] = m^{17} mod[3233]$
- The private key is given by n = 3233 and d = 2753
- The decryption function $c^{d} mod[n] = c^{2753} mod[3233]$

RSA Algorithm - A real example with numbers

- The public key is given by the 2 numbers n = 3233 and e = 17
- The encryption function is $m^e mod[n] = m^{17} mod[3233]$
- The private key is given by n = 3233 and d = 2753
- The decryption function $c^d mod[n] = c^{2753} mod[3233]$
- Let's put m = 65

RSA Algorithm - A real example with numbers

- The public key is given by the 2 numbers n = 3233 and e = 17
- The encryption function is $m^e mod[n] = m^{17} mod[3233]$
- The private key is given by n = 3233 and d = 2753
- The decryption function $c^d mod[n] = c^{2753} mod[3233]$
- Let's put m = 65
- Bob encrypts this text by using $c = m^e mod[n] = 65^{17} mod[3233] = 2790$

RSA Algorithm - A real example with numbers

- The public key is given by the 2 numbers n = 3233 and e = 17
- The encryption function is $m^e mod[n] = m^{17} mod[3233]$
- The private key is given by n = 3233 and d = 2753
- The decryption function $c^d mod[n] = c^{2753} mod[3233]$
- Let's put m = 65
- Bob encrypts this text by using $c = m^e mod[n] = 65^{17} mod[3233] = 2790$
- Alice decrypts c by using $m = c^d mod[n] = 2790^{2753} mod[3233] = 65$

RSA Algorithm - A real example with numbers

- The public key is given by the 2 numbers n = 3233 and e = 17
- The encryption function is $m^e mod[n] = m^{17} mod[3233]$
- The private key is given by n = 3233 and d = 2753
- The decryption function $c^d mod[n] = c^{2753} mod[3233]$
- Let's put m = 65
- Bob encrypts this text by using $c = m^e mod[n] = 65^{17} mod[3233] = 2790$
- Alice decrypts c by using $m = c^d mod[n] = 2790^{2753} mod[3233] = 65$

SSH

Life with SSH

What SSH does?

A protocol, not only a software.

It is composed of programs like: `ssh, scp, sftp, sshfs`

It takes care of:

1. Authentication:
   - Reliably determines someone's identity.
     When you login on a remote computer, SSH asks for digital proof of your identity.
     If you pass the test, you may log in; otherwise SSH rejects the connection.
2. Encryption
   - Scrambles data so it is unintelligible except to the intended recipients.
     This protects your data as it passes over the network.
3. Integrity
   - Guarantees the data traveling over the network arrives unaltered.
     If a third party captures and modifies your data in transit, SSH detects this fact.

How SSH works?

SSH is an client-server based architecture, built on three layers

1. Transport layer:
    - Handles initial key exchange and server authentication
    - Sets up encryption, compression and integrity verification. No data will pass in clear.
    - It uses port 22
2. User authentication layer
    - Handles client authentication

      ⇒ By password authentication

      ⇒ By public key authentication
3. Connection layer
    - Handles data transmission and services transferred via the SSH channel

How SSH works?

On the server side:

- A couple of keys is generated during the installation.
- These keys are used for authenticate server identity
- They are stored in /etc/ssh folder
- `/etc/ssh/sshd_config`
  contains the server-side basic configuration and options (i.e. port $\neq 22$, root connections, etc.)

SSH files

On the client side:

- `/etc/ssh/ssh_config`
  contains the overall client-side configuration and options (i.e. restriction on the connection to a given server)
- `$HOME/.ssh directory`
  Usually it contains:
  `known_hosts`
  `id_rsa` (your private key with RSA encoding)
  `id_rsa.pub` (your public key with RSA encoding)
  `authorized_keys`
  `config`

Configuration data is parsed as follows:

- command line options
- user-specific file
- system-wide file

SSH settings - Passwords

Password-mode connection:

-
  ```
  ssh verdi
  The authenticity of host ’verdi (129.104.22.80)’
  can’t be established.
  RSA key fingerprint is
  79:c0:a3:1a:0f:12:b8:a1:c0:93:41:d2:6a:6b:ae:9d.
  Are you sure you want to continue connecting
  (yes/no)?
  ```

-
  ```
  Warning:  Permanently added the RSA host key for
  IP address ’129.104.22.80’ to the list of known
  hosts.
  cucca@verdi’s password:
  ```

SSH settings - Passwords

Password-mode connection:

-
  ```
  ssh verdi
  The authenticity of host 'verdi (129.104.22.80)'
  can't be established.
  RSA key fingerprint is
  79:c0:a3:1a:0f:12:b8:a1:c0:93:41:d2:6a:6b:ae:9d.
  Are you sure you want to continue connecting
  (yes/no)?
  ```

-
  ```
  Warning:  Permanently added the RSA host key for
  IP address '129.104.22.80' to the list of known
  hosts.
  cucca@verdi's password:
  ```
- A new line has been added to $HOME/.ssh/known_hosts containing an ID that will be checked at every connection.

SSH settings - Passwords

Password-mode connection:

-
  ```
  ssh verdi
  The authenticity of host 'verdi (129.104.22.80)'
  can't be established.
  RSA key fingerprint is
  79:c0:a3:1a:0f:12:b8:a1:c0:93:41:d2:6a:6b:ae:9d.
  Are you sure you want to continue connecting
  (yes/no)?
  ```

-
  ```
  Warning:  Permanently added the RSA host key for
  IP address '129.104.22.80' to the list of known
  hosts.
  cucca@verdi's password:
  ```
- A new line has been added to $HOME/.ssh/known_hosts containing an ID that will be checked at every connection.

SSH settings -Keys

Key-mode connection:

-
  ```
  ssh-keygen
  Generating public/private rsa key pair.
  Enter file in which to save the key (/home/cucca/.ssh/id_rsa):
  Enter passphrase (empty for no passphrase):
  Enter same passphrase again:
  ```

SSH settings -Keys

Key-mode connection:

-
  ```
  ssh-keygen
  Generating public/private rsa key pair.
  Enter file in which to save the key (/home/cucca/.ssh/id_rsa):
  Enter passphrase (empty for no passphrase):
  Enter same passphrase again:
  Your identification has been saved in /home/cucca/.ssh/id_rsa.
  Your public key has been saved in /home/cucca/.ssh/id_rsa.pub.
  The key fingerprint is:
  dc:a0:85:c4:d8:22:a3:8c:24:b7:80:cf:80:80:f9:90 cucca@verdi.polytechnique.fr
  The key's randomart image is:
  +--
                                       RSA2048

  ----+
  |*o +.  |
  |E.+ o.o.  |
  |=X + ..  o |
  |o.= + o |
  | .  S . |
  | |
  | |
  | |
  | |
  +-----------------+
  ```

SSH settings -Keys

Key-mode connection:

-
  ```
  ssh-keygen
  Generating public/private rsa key pair.
  Enter file in which to save the key (/home/cucca/.ssh/id_rsa):
  Enter passphrase (empty for no passphrase):
  Enter same passphrase again:
  Your identification has been saved in /home/cucca/.ssh/id_rsa.
  Your public key has been saved in /home/cucca/.ssh/id_rsa.pub.
  The key fingerprint is:
  dc:a0:85:c4:d8:22:a3:8c:24:b7:80:cf:80:80:f9:90 cucca@verdi.polytechnique.fr
  The key's randomart image is:
  +--
                                     RSA2048

  ----+
  |*o +.  |
  |E.+ o.o.  |
  |=X + ..  o |
  |o.= + o |
  | .  S . |
  | |
  | |
  | |
  | |
  +-----------------+
  ```

SSH settings - Keys

Key-mode connection:

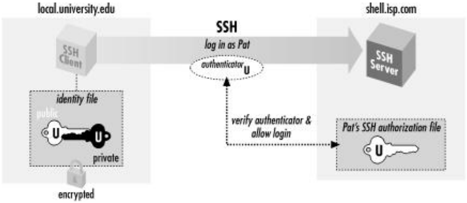
- `ssh-copy-id -i $HOME/.ssh/id_rsa.pub login@server`
- In the remote server a line has been added to $HOME/.ssh/authorized_keys
- Example:
  `ssh-copy-id -i /home/cucca/.ssh/id_rsa.pub cucca@nero`

SSH settings - Keys

Key-mode connection:

- `ssh-copy-id -i $HOME/.ssh/id_rsa.pub login@server`
- In the remote server a line has been added to $HOME/.ssh/authorized_keys
- Example:

```
ssh-copy-id -i /home/cucca/.ssh/id_rsa.pub
cucca@nero
```

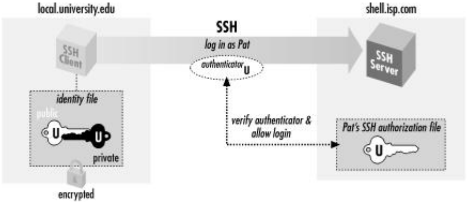


- Safer than password, even when using the most secure password in the world (i.e. `bonjour001`)

SSH settings - Keys

Key-mode connection:

-
  ```
  ssh-copy-id -i $HOME/.ssh/id_rsa.pub
  login@server
  ```
- In the remote server a line has been added to $HOME/.ssh/authorized_keys
- Example:
  ```
  ssh-copy-id -i /home/cucca/.ssh/id_rsa.pub
  cucca@nero
  ```



- Safer than password, even when using the most secure password in the world (i.e. `bonjour001`)

SSH settings - Keys

- *Caveat*: you are not safe against ID files loss.
  Use passphrases!
- Passphrase + keys improves the security
- ssh-agent keep (encrypted) trace of passphrases for every key couple you have in your account.
-
  ```
  ssh-agent
  SSH_AUTH_SOCK=/tmp/ssh-PCmQylG18617/agent.18617; export SSH_AUTH_SOCK;
  SSH_AGENT_PID=18618; export SSH_AGENT_PID;
  echo Agent pid 18618;
  ```
-
  ```
  ssh-add
  Enter passphrase for /home/cucca/.ssh/id_dsa:
  Identity added:  /home/cucca/.ssh/id_dsa (/home/cucca/.ssh/id_dsa)
  ```
- Now you can login without password and without passphrase

SSH settings - Keys

- *Caveat*: you are not safe against ID files loss.
  Use passphrases!
- Passphrase + keys improves the security
- `ssh-agent` keep (encrypted) trace of passphrases for every key couple you have in your account.

-
  ```
  ssh-agent
  SSH_AUTH_SOCK=/tmp/ssh-PCmQylG18617/agent.18617; export SSH_AUTH_SOCK;
  SSH_AGENT_PID=18618; export SSH_AGENT_PID;
  echo Agent pid 18618;
  ```

-
  ```
  ssh-add
  Enter passphrase for /home/cucca/.ssh/id_dsa:
  Identity added:  /home/cucca/.ssh/id_dsa (/home/cucca/.ssh/id_dsa)
  ```
- Now you can login without password and without passphrase
- Use `ssh-add -D` to delete a key
- Use `ssh-add -l` to list the stored keys

SSH settings - Keys

- *Caveat*: you are not safe against ID files loss.
  Use passphrases!
- Passphrase + keys improves the security
- `ssh-agent` keep (encrypted) trace of passphrases for every key couple you have in your account.
-
  ```
  ssh-agent
  SSH_AUTH_SOCK=/tmp/ssh-PCmQylG18617/agent.18617; export SSH_AUTH_SOCK;
  SSH_AGENT_PID=18618; export SSH_AGENT_PID;
  echo Agent pid 18618;
  ```
-
  ```
  ssh-add
  Enter passphrase for /home/cucca/.ssh/id_dsa:
  Identity added:  /home/cucca/.ssh/id_dsa (/home/cucca/.ssh/id_dsa)
  ```
- Now you can login without password and without passphrase
- Use `ssh-add -D` to delete a key
- Use `ssh-add -l` to list the stored keys
- If you want to modify a passhphrase use:
  ssh-keygen -p -f $HOME/.ssh/id_dsa
  Enter old passphrase:
  Key has comment '/home/cucca/.ssh/id_dsa'
  Enter new passphrase (empty for no passphrase):
  Enter same passphrase again:
  Your identification has been saved with the new passphrase.

SSH settings - Keys

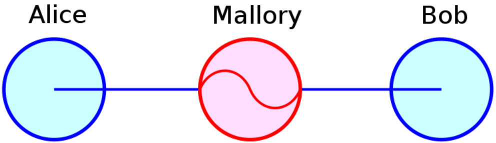
- *Caveat*: you are not safe against ID files loss.
  Use passphrases!
- Passphrase + keys improves the security
- `ssh-agent` keep (encrypted) trace of passphrases for every key couple you have in your account.
-
  ```
  ssh-agent
  SSH_AUTH_SOCK=/tmp/ssh-PCmQylG18617/agent.18617; export SSH_AUTH_SOCK;
  SSH_AGENT_PID=18618; export SSH_AGENT_PID;
  echo Agent pid 18618;
  ```
-
  ```
  ssh-add
  Enter passphrase for /home/cucca/.ssh/id_dsa:
  Identity added:  /home/cucca/.ssh/id_dsa (/home/cucca/.ssh/id_dsa)
  ```
- Now you can login without password and without passphrase
- Use `ssh-add -D` to delete a key
- Use `ssh-add -l` to list the stored keys
- If you want to modify a passhphrase use:
  ```
  ssh-keygen -p -f $HOME/.ssh/id_dsa
  Enter old passphrase:
  Key has comment '/home/cucca/.ssh/id_dsa'
  Enter new passphrase (empty for no passphrase):
  Enter same passphrase again:
  Your identification has been saved with the new passphrase.
  ```

Man-in-the-Middle

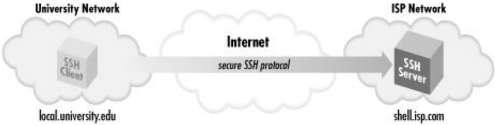
You may obtain:

-
  ```
  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  @ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
  @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
  Someone could be eavesdropping on you right now (man-in-the-middle attack)!
  It is also possible that a host key has just been changed.
  The fingerprint for the RSA key sent by the remote host is
  79:c0:a3:1a:0f:12:b8:a1:c0:93:41:d2:6a:6b:ae:9d.
  Please contact your system administrator.
  Add correct host key in /home/cucca/.ssh/known_hosts to get rid of this message.
  Offending RSA key in /home/cucca/.ssh/known_hosts:44
  remove with: ssh-keygen -f "/home/cucca/.ssh/known_hosts" -R verdi
  RSA host key for verdi has changed and you have requested strict checking.

  Host key verification failed.
  ```



Basic ssh use - shell



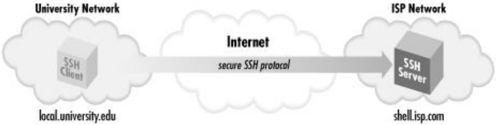
- Login to a shell on a remote host (replacing telnet and rlogin)

  `ssh login@server` (or → `ssh server`)

- Executing a single command on a remote host

  ```
  ssh login@server ls
  ssh login@server less remote-file
  ```

Basic ssh use - shell



- Login to a shell on a remote host (replacing telnet and rlogin)

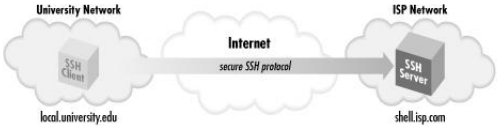
  `ssh login@server` (or → `ssh server`)

- Executing a single command on a remote host

  ```
  ssh login@server ls
  ssh login@server less remote-file
  ```

⇒ Try to add the −v (or −vv) flag to the command line.

Basic ssh use - shell



- Login to a shell on a remote host (replacing telnet and rlogin)

  `ssh login@server` (or → `ssh server`)

- Executing a single command on a remote host

  ```
  ssh login@server ls
  ssh login@server less remote-file
  ```

⇒ Try to add the `-v` (or `-vv`) flag to the command line.

Basic ssh use - copy

- Secure file transfer (with `scp`)
  - `scp /path/to/localfile login@server:/path/to/remotefile` (push method)
  - `scp login@server:/path/to/remotefile /path/to/localfile` (pull method)

Basic ssh use - copy

- Secure file transfer (with scp)
  - `scp /path/to/localfile login@server:/path/to/remotefile` (push method)
  - `scp login@server:/path/to/remotefile /path/to/localfile` (pull method)
- Examples:
  - `scp /home/cucca/testfile cucca@nero:/home/cucca/testfolder/`
  - `scp cucca@nero:/home/cucca/testfolder /home/cucca/testfile`

Basic ssh use - copy

- Secure file transfer (with scp)
  - `scp /path/to/localfile login@server:/path/to/remotefile` (push method)
  - `scp login@server:/path/to/remotefile /path/to/localfile` (pull method)
- Examples:
  - `scp /home/cucca/testfile cucca@nero:/home/cucca/testfolder/`
  - `scp cucca@nero:/home/cucca/testfolder /home/cucca/testfile`
- Copy several files:
  - `scp /home/cucca/*.KSS cucca@nero:/home/cucca/KSSfolder/`

Basic ssh use - copy

- Secure file transfer (with scp)
  - `scp /path/to/localfile login@server:/path/to/remotefile` (push method)
  - `scp login@server:/path/to/remotefile /path/to/localfile` (pull method)
- Examples:
  - `scp /home/cucca/testfile cucca@nero:/home/cucca/testfolder/`
  - `scp cucca@nero:/home/cucca/testfolder /home/cucca/testfile`
- Copy several files:
  - `scp /home/cucca/*.KSS cucca@nero:/home/cucca/KSSfolder/`
  - `for h in host1 host2 host3 host4 ; { scp file user@$h:/destination_path/ ; }`

Basic ssh use - copy

- Secure file transfer (with scp)
  - `scp /path/to/localfile login@server:/path/to/remotefile` (push method)
  - `scp login@server:/path/to/remotefile /path/to/localfile` (pull method)
- Examples:
  - `scp /home/cucca/testfile cucca@nero:/home/cucca/testfolder/`
  - `scp cucca@nero:/home/cucca/testfolder /home/cucca/testfile`
- Copy several files:
  - `scp /home/cucca/*.KSS cucca@nero:/home/cucca/KSSfolder/`
  - `for h in host1 host2 host3 host4 ; { scp file user@$h:/destination_path/ ; }`

Basic ssh use - copy

- Copy directories
  - `scp -rp /home/cucca/testfolder cucca@nero:/home/cucca/testfolder/`
- Copy from distant to distant
  - `scp login@server1:myfile login@server2:myfile`

Basic ssh use - copy

- Copy directories
  - `scp -rp /home/cucca/testfolder cucca@nero:/home/cucca/testfolder/`
- Copy from distant to distant
  - `scp login@server1:myfile login@server2:myfile`
- Use non standard ports
  - `ssh -p 2222 cucca@server`
  - `scp -P 2222 /home/cucca/testfolder cucca@server:testfolder/`

Basic ssh use - copy

- Copy directories

  -
    ```
    scp -rp /home/cucca/testfolder
    cucca@nero:/home/cucca/testfolder/
    ```
- Copy from distant to distant

  -
    ```
    scp login@server1:myfile
    login@server2:myfile
    ```
- Use non standard ports

  -
    ```
    ssh -p 2222 cucca@server
    ```

  -
    ```
    scp -P 2222 /home/cucca/testfolder
    cucca@server:testfolder/
    ```

There are more options. Please type `man scp` / `man ssh`

Basic ssh use - copy

- Copy directories
  - `scp -rp /home/cucca/testfolder cucca@nero:/home/cucca/testfolder/`
- Copy from distant to distant
  - `scp login@server1:myfile login@server2:myfile`
- Use non standard ports
  - `ssh -p 2222 cucca@server`
  - `scp -P 2222 /home/cucca/testfolder cucca@server:testfolder/`

There are more options. Please type `man scp` / `man ssh`

X forwarding

```
ssh cucca@nero xmgrace
```

```
Can’t open display
Failed initializing GUI, exiting
```

X forwarding

```
ssh cucca@nero xmgrace

Can't open display
Failed initializing GUI, exiting
```

Graphical programs need X server to be executed.
X is not a standard program, and needs several variables to be set (*DISPLAY*, *HOST:n.v*).

X forwarding

```
ssh cucca@nero xmgrace
```

```
Can't open display
Failed initializing GUI, exiting
```

Graphical programs need X server to be executed.
X is not a standard program, and needs several variables to be set (*DISPLAY*, *HOST:n.v*).

- ssh sets `$DISPLAY` pointing on the server machine, a .Xauthority file, and a “fake” authentication cookie
- `ssh -X cucca@nero xmgrace`
- Must be enabled in client configuration file (Default = Yes).

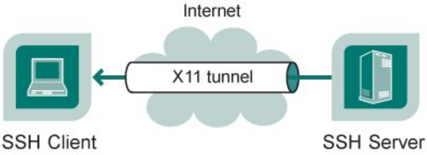
X forwarding

```
ssh cucca@nero xmgrace
```

```
Can't open display
Failed initializing GUI, exiting
```

Graphical programs need X server to be executed.
X is not a standard program, and needs several variables to be set (*DISPLAY*, *HOST:n.v*).

- ssh sets `$DISPLAY` pointing on the server machine, a .Xauthority file, and a "fake" authentication cookie
- `ssh -X cucca@nero xmgrace`
- Must be enabled in client configuration file (Default = Yes).



- `-Y` flag can be used instead of `-X` but:

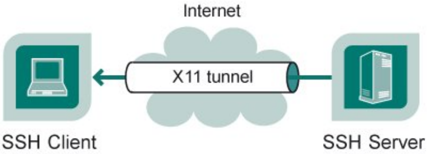
X forwarding

```
ssh cucca@nero xmgrace
```

```
Can't open display
Failed initializing GUI, exiting
```

Graphical programs need X server to be executed.
X is not a standard program, and needs several variables to be set (*DISPLAY, HOST:n.v*).

- ssh sets `$DISPLAY` pointing on the server machine, a .Xauthority file, and a "fake" authentication cookie
- `ssh -X cucca@nero xmgrace`
- Must be enabled in client configuration file (Default = Yes).



- `-Y` flag can be used instead of `-X` but:
- X is already able to do connection with remote clients but they are insecure (i.e. uncrypted) and therefore subject to restriction in remote access.
- `-Y` flags enables trusted X11 forwarding (so lowering the security level)

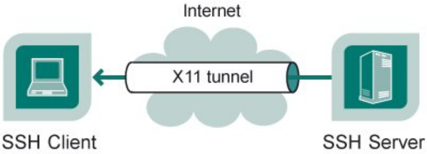
X forwarding

```
ssh cucca@nero xmgrace

Can't open display
Failed initializing GUI, exiting
```

Graphical programs need X server to be executed.
X is not a standard program, and needs several variables to be set (*DISPLAY, HOST:n.v*).

- ssh sets `$DISPLAY` pointing on the server machine, a .Xauthority file, and a "fake" authentication cookie
- `ssh -X cucca@nero xmgrace`
- Must be enabled in client configuration file (Default = Yes).



- `-Y` flag can be used instead of `-X` but:
- X is already able to do connection with remote clients but they are insecure (i.e. uncrypted) and therefore subject to restriction in remote access.
- `-Y` flags enables trusted X11 forwarding (so lowering the security level)

⇒ ssh do all the setting work for you at the cost of a `-X` flag

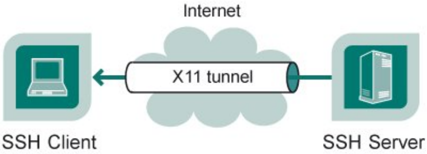
X forwarding

```
ssh cucca@nero xmgrace

Can't open display
Failed initializing GUI, exiting
```

Graphical programs need X server to be executed.
X is not a standard program, and needs several variables to be set (*DISPLAY, HOST:n.v*).

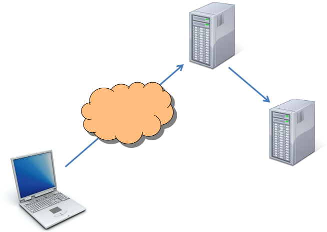
- ssh sets `$DISPLAY` pointing on the server machine, a .Xauthority file, and a "fake" authentication cookie
- `ssh -X cucca@nero xmgrace`
- Must be enabled in client configuration file (Default = Yes).



- `-Y` flag can be used instead of `-X` but:
- X is already able to do connection with remote clients but they are insecure (i.e. uncrypted) and therefore subject to restriction in remote access.
- `-Y` flags enables trusted X11 forwarding (so lowering the security level)

⇒ ssh do all the setting work for you at the cost of a `-X` flag

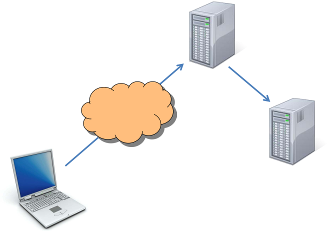
SSH tunneling



Connection to a remote machine protected by a firewall.

```
ssh -t theory ssh nero
```

SSH tunneling

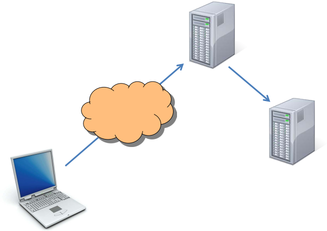


Connection to a remote machine protected by a firewall.

```
ssh -t theory ssh nero
```

- You can add flags like `-C`, `-X`

SSH tunneling

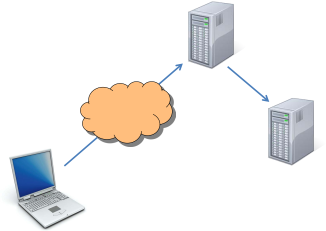


Connection to a remote machine protected by a firewall.

```
ssh -t theory ssh nero
```

- You can add flags like `-C, -X`
- On the opposite schema, you might want to connect to a remote machine passing through a proxy server.
- Edit the $HOME/.ssh/config file adding the following lines
-
```
Host *.idris.fr
ProxyCommand connect -H cache.polytechnique.fr:8080 %h %p
```
- Need to install a program like "connect"

SSH tunneling



Connection to a remote machine protected by a firewall.

```
ssh -t theory ssh nero
```

- You can add flags like `-C, -X`
- On the opposite schema, you might want to connect to a remote machine passing through a proxy server.
- Edit the $HOME/.ssh/config file adding the following lines

-
  ```
  Host *.idris.fr
  ProxyCommand connect -H cache.polytechnique.fr:8080 %h %p
  ```
- Need to install a program like "connect"

Port forwarding

- Port forwarding means redirect TCP traffic of non encrypted applications listenenig to insecure ports to other ports securised by ssh
- Syntax is: `ssh -L local_port:HOSTNAME:remote_port login@remote_machine`
- Example: ssh can redirect the local web traffic to a given port on a remote machine
- Consult scientific reviews accessible only from Polytechnique

Port forwarding

- Port forwarding means redirect TCP traffic of non encrypted applications listenenig to insecure ports to other ports securised by ssh
- Syntax is: `ssh -L local_port:HOSTNAME:remote_port login@remote_machine`
- Example: ssh can redirect the local web traffic to a given port on a remote machine
- Consult scientific reviews accessible only from Polytechnique
- `ssh -L9081:cache.polytechnique.fr:8080 cucca@theory.polytechnique.fr`
- ssh forwards 9081 local traffic to 8080 port of cache.polytechnique.fr
- Then change your browser settings and put proxy = localhost:9081 (foxyproxy highly recommended)

Port forwarding

- Port forwarding means redirect TCP traffic of non encrypted applications listenenig to insecure ports to other ports securised by ssh
- Syntax is: `ssh -L local_port:HOSTNAME:remote_port login@remote_machine`
- Example: ssh can redirect the local web traffic to a given port on a remote machine
- Consult scientific reviews accessible only from Polytechnique
- `ssh -L9081:cache.polytechnique.fr:8080 cucca@theory.polytechnique.fr`
- ssh forwards 9081 local traffic to 8080 port of cache.polytechnique.fr
- Then change your browser settings and put proxy = localhost:9081 (foxyproxy highly recommended)
- Use a socksifier program like tsocks (to bypass standard proxy configuration with customised ones)
- Add to `/etc/tsocks.conf` file your remote server (theory) and a port (9080)
- `ssh -D9080 cucca@theory.polytechnique.fr`
- This start a SOCKS proxy on localhost, port 9080
- `tsocks firefox`

Port forwarding

- Port forwarding means redirect TCP traffic of non encrypted applications listenenig to insecure ports to other ports securised by ssh
- Syntax is: `ssh -L local_port:HOSTNAME:remote_port login@remote_machine`
- Example: ssh can redirect the local web traffic to a given port on a remote machine
- Consult scientific reviews accessible only from Polytechnique
- `ssh -L9081:cache.polytechnique.fr:8080 cucca@theory.polytechnique.fr`
- ssh forwards 9081 local traffic to 8080 port of cache.polytechnique.fr
- Then change your browser settings and put proxy = localhost:9081 (foxyproxy highly recommended)
- Use a socksifier program like tsocks (to bypass standard proxy configuration with customised ones)
- Add to `/etc/tsocks.conf` file your remote server (theory) and a port (9080)
- `ssh -D9080 cucca@theory.polytechnique.fr`
- This start a SOCKS proxy on localhost, port 9080
- `tsocks firefox`
- Remote port forwarding:
  `ssh -R remote_port:HOSTNAME:local_port login@remote_machine`
- As example think about svn that listen to 3690.
  `svn checkout svn+ssh://etsf.polytechnique.fr/var/svn/repos`

Port forwarding

- Port forwarding means redirect TCP traffic of non encrypted applications listenenig to insecure ports to other ports securised by ssh
- Syntax is: `ssh -L local_port:HOSTNAME:remote_port login@remote_machine`
- Example: ssh can redirect the local web traffic to a given port on a remote machine
- Consult scientific reviews accessible only from Polytechnique
- `ssh -L9081:cache.polytechnique.fr:8080 cucca@theory.polytechnique.fr`
- ssh forwards 9081 local traffic to 8080 port of cache.polytechnique.fr
- Then change your browser settings and put proxy = localhost:9081 (foxyproxy highly recommended)
- Use a socksifier program like tsocks (to bypass standard proxy configuration with customised ones)
- Add to `/etc/tsocks.conf` file your remote server (theory) and a port (9080)
- `ssh -D9080 cucca@theory.polytechnique.fr`
- This start a SOCKS proxy on localhost, port 9080
- `tsocks firefox`
- Remote port forwarding:
  `ssh -R remote_port:HOSTNAME:local_port login@remote_machine`
- As example think about svn that listen to 3690.
  `svn checkout svn+ssh://etsf.polytechnique.fr/var/svn/repos`

SSHFS

SSHFS - Securely mount a directory on a remote server as a filesystem on a local computer.

- `sshfs hostname:remote_mount_point local_mount_point`
- Mount ⇒ `sshfs nero:  /home/cucca/nero`
- Unmount ⇒ `fusermount -u /home/cucca/nero`
- SSHFS is a FUSE filesystem: can't be shared between multiple users
- It can be forced to do it but file permission will be wrong. Does not support `statfs`

Multiple connections (home → theory → nero)

- `ssh -L6666:nero:22 cucca@theory.polytechnique.fr`
- `sshfs -p 6666 cucca@localhost:  /home/cucca/nero`

SSHFS

SSHFS - Securely mount a directory on a remote server as a filesystem on a local computer.

- `sshfs hostname:remote_mount_point local_mount_point`
- Mount ⇒ `sshfs nero:  /home/cucca/nero`
- Unmount ⇒ `fusermount -u /home/cucca/nero`
- SSHFS is a FUSE filesystem: can't be shared between multiple users
- It can be forced to do it but file permission will be wrong. Does not support `statfs`

Multiple connections (home → theory → nero)

- `ssh -L6666:nero:22 cucca@theory.polytechnique.fr`
- `sshfs -p 6666 cucca@localhost:  /home/cucca/nero`
- A second (perhaps simpler) way is edit .ssh/config

-
  ```
  Host nero.polytechnique.fr
  ProxyCommand /usr/bin/ssh -W %h:%p theory.polytechnique.fr
  ```
- `sshfs nero.polytechnique.fr:  /home/cucca/nero`

SSHFS

SSHFS - Securely mount a directory on a remote server as a filesystem on a local computer.

- `sshfs hostname:remote_mount_point local_mount_point`
- Mount ⇒ `sshfs nero:  /home/cucca/nero`
- Unmount ⇒ `fusermount -u /home/cucca/nero`
- SSHFS is a FUSE filesystem: can't be shared between multiple users
- It can be forced to do it but file permission will be wrong. Does not support `statfs`

Multiple connections (home → theory → nero)

- `ssh -L6666:nero:22 cucca@theory.polytechnique.fr`
- `sshfs -p 6666 cucca@localhost:  /home/cucca/nero`
- A second (perhaps simpler) way is edit .ssh/config
-
  ```
  Host nero.polytechnique.fr
  ProxyCommand /usr/bin/ssh -W %h:%p theory.polytechnique.fr
  ```
- `sshfs nero.polytechnique.fr:  /home/cucca/nero`

rsync + ssh & sftp

rsync

- rsync + ssh can back up, copy and mirror files efficiently and securely
- `rsync -avz -e ssh -r -l -p -g -t -x $HOME/directory server:$HOME/directory`
- -a, –archive archive mode
  -v, –verbose increase verbosity
  -z, –compress compress file data during the transfer
  -e, –rsh=COMMAND specify the remote shell to use
  -r, –recursive recurse into directories
  -l, –links copy symlinks as symlinks
  -p, –perms preserve permissions
  -g, –group preserve group
  -t, –times preserve modification times
  -x, –one-file-system don't cross filesystem boundaries
- Our backup system is based on rsync

sftp

- Sftp works as ftp but using the SSH protocol.
- Browse remote files and directories
- Remove remote files
- Resume interrupted transfers

rsync + ssh & sftp

rsync

- rsync + ssh can back up, copy and mirror files efficiently and securely
- `rsync -avz -e ssh -r -l -p -g -t -x $HOME/directory server:$HOME/directory`
- -a, –archive archive mode
  -v, –verbose increase verbosity
  -z, –compress compress file data during the transfer
  -e, –rsh=COMMAND specify the remote shell to use
  -r, –recursive recurse into directories
  -l, –links copy symlinks as symlinks
  -p, –perms preserve permissions
  -g, –group preserve group
  -t, –times preserve modification times
  -x, –one-file-system don't cross filesystem boundaries
- Our backup system is based on rsync

sftp

- Sftp works as ftp but using the SSH protocol.
- Browse remote files and directories
- Remove remote files
- Resume interrupted transfers

Final remarks

- SSH is a safe, secure, versatile program
- It simplifies your job
- It provides you access to your work environment when you're away from office
- It's trusteful and secret but....

Final remarks

Just keep your password secret!

SSH!
DON'T GIVE AWAY YOUR PASSWORD TO ANYONE WHO MIGHT BE LISTENING!
Always use SSH (Secure SHell) when connecting to Computer Science Department hosts, instead of telnet or rsh. For more information:
http://www.cs.umd.edu/faq/ssh.html

Bibliography:

- man ssh
- http://www.openssh.org
- http://docstore.mik.ua/orelly/networking_2ndEd/ssh/index.htm
- http://en.wikipedia.org/wiki/Secure_Shell (and links therein)

Thanks for your attention

(The End)