C2110 UNIX and programming

2nd lesson

Petr Kulhánek, Jakub Štěpán

kulhanek@chemi.muni.cz

National Centre for Biomolecular Research, Faculty of Science
Masaryk University, Kotlářská 2, CZ-61137 Brno

INVESTMENTS IN EDUCATION DEVELOPMENT

CZ.1.07/2.2.00/15.0233

Contents

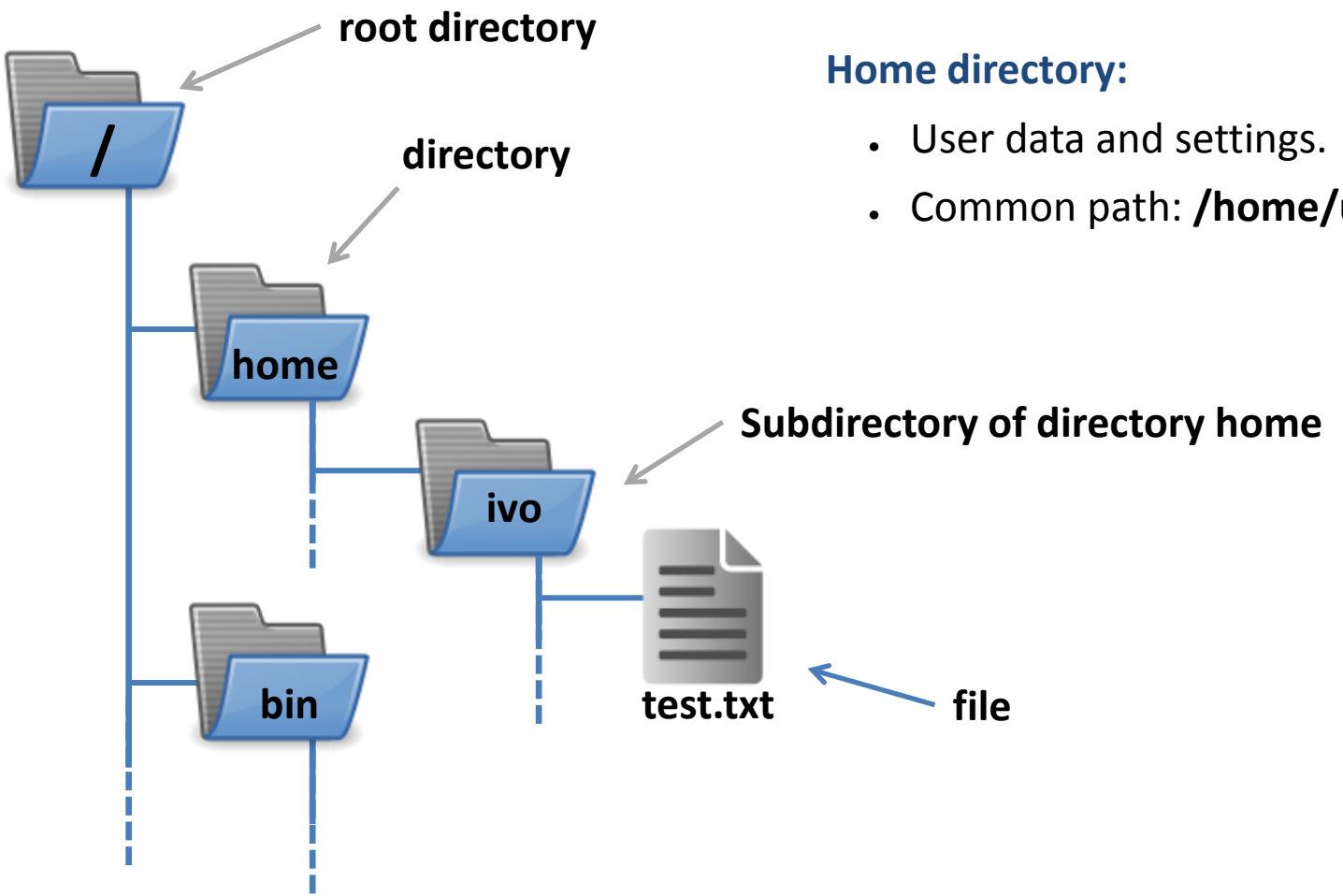
- **Unix in cube**
  - **File system, paths**
  - **Submitting commands**
  - **Basic commands**
    - File system browsing
    - Copying, moving, deleting
- **Remote login**
  - **ssh**
  - **Encryption**
  - **Recursive login**

Unix in cube

- File system, paths
- Submitting commands
- Basic commands
  - File system browsing
  - Copying, moving, deleting

File system structure

UNIX uses **hierarchical** directory **file system** consisting of directories and files. All directories and files are located in the **only root rirectory (/)**.



Home directory:

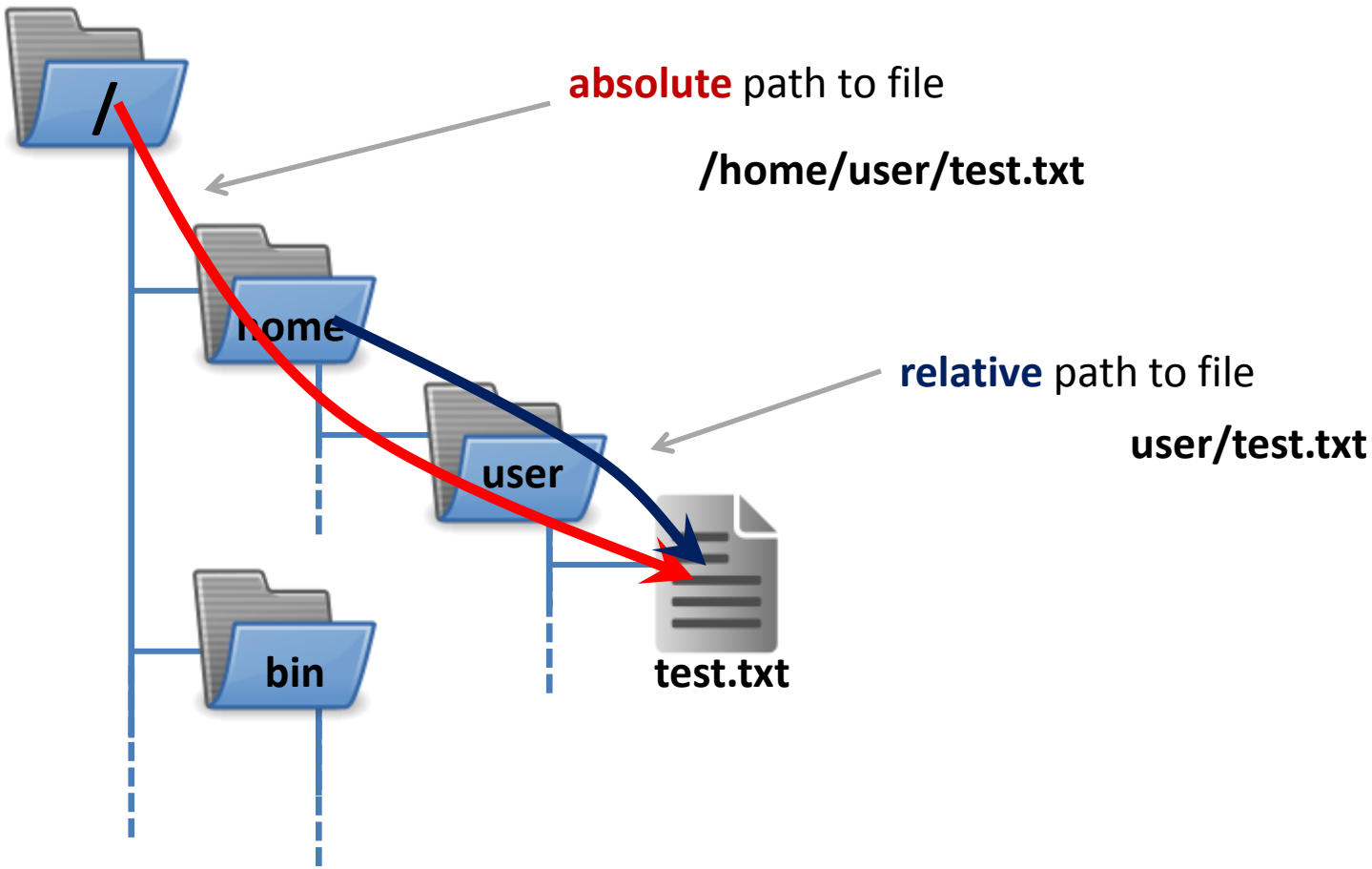
- User data and settings.
- Common path: **/home/user_name**

Comparison with MS Windows fs

Property	Linux (ext2/ext3/ext4)	MS Windows (FAT32,NTFS)
Disk partitions	Hidden Disk partitions are connected as directories.	C:, D:, etc. Optionally connectable as directory (ntfs).
Names	Case sensitive.	Case insensitive.
Name separator	Slash	Back slash
Access permissions	Yes POSIX	Yes (only NTFS) ACL
Devices (hardware)	As special files.	No.

PATH File and directory identifier

Path to directory or file can be defined as **absolut** or **relative**. File and directory names are separated by **slash /**.



Path types

Absolute path has to be specified to root or home directory. Thus it starts either by slash **/** or tilde **~**.

/home/kulhanek/Documents/home_work.txt

Use of tilda:

~	home directory of current user
~user_name	home directory of user with name user_name

Relative path is specified to current / work directory. (Absolute path of current directory can be obtained by command **pwd**.)

../alois/Documents

Special directory names:

. (dot)	current / work directory
.. (two dots)	parent (higher) directory

Path examples

Absolute paths:

/home/kulhanek/Documents

/home/kulhanek/Documents/domaci_ukol.txt

~/Documents → **/home/kulhanek**/Documents

~alois/Documents → **/home/alois**/Documents

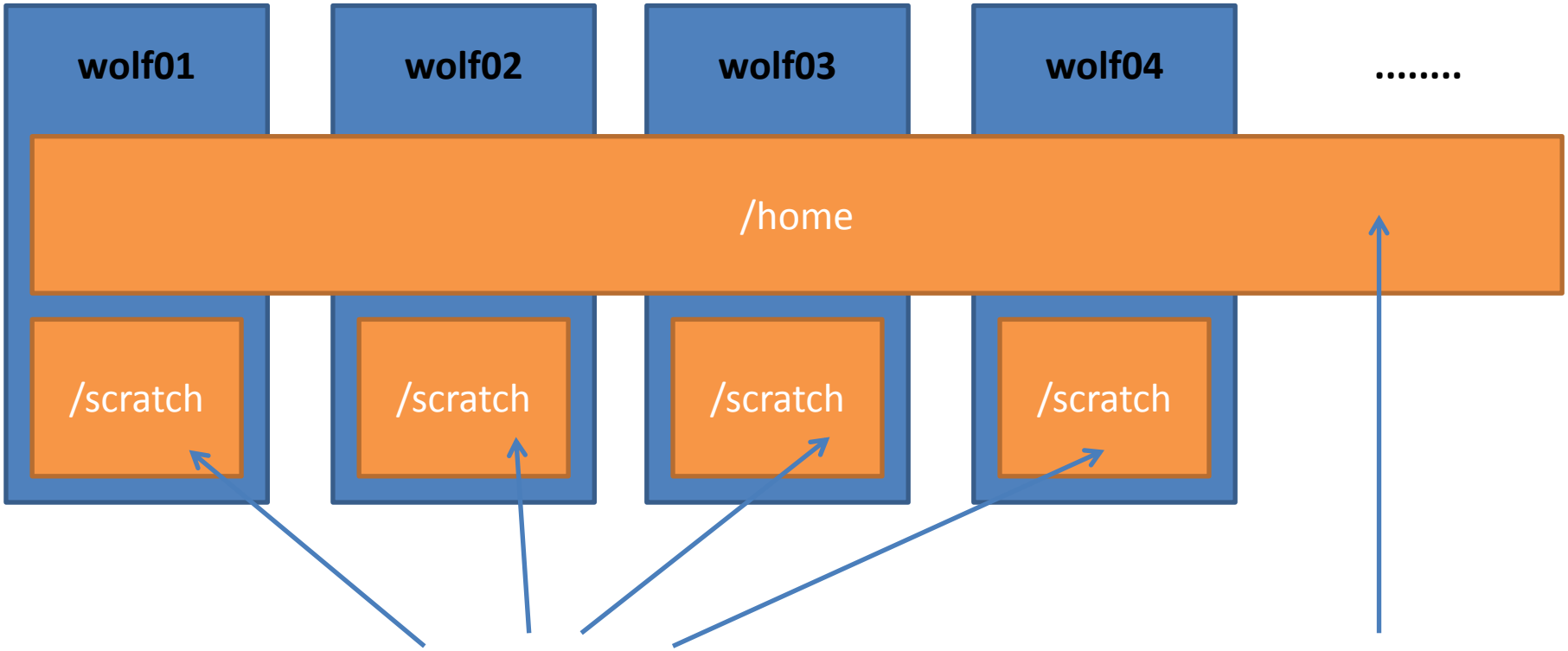
Relative paths:

Documents/domaci_ukol.txt

../alois/Documents

./muj_script

WOLF cluster file system



Different contents on all nodes.
Data on volume /scratch has **no backup** and can be deleted anytime.
Per-user **capacity is not restricted**.

Shared contents on all WOLF cluster nodes.
Data **has backup**. Capacity **is restricted** by quota per-user to maximum **1,5GB**.

Command line

Prompt – user type ($ regular user, # super user, other prompts %, >)

```
[kulhanek@wolf ~]$
```

User name

Machine name

Current directory (~ is home directory /home/login)

commands

Command is given by key **Enter**.

History: by arrow keys (up, down) list of recently used commands can be searched. Any command can be re-used or edited and used. Full list of recorded commands can be printed by command **history**.

Auto complete: Tab key makes command line interpreter to try complete started word. Completed can be command names, paths, file names (if one click does not complete word, there is more possibilities to complete, another click shows list of them).

Text copy: Do not use Ctrl+C! Mouse text select automatically adds text to clipboard, press mouse wheel to paste on cursor place.

Commands help, cmd searching

Manual pages (When I do not know what to do?):

man prints manual page of command

$ **man** [section_number] topic

Name of command, function, theme, chapter

Sections:

- *Section 1* user commands
- *Section 2* system calls
- *Section 3* library functions
- *Section 4* special files
- *Section 5* *file formats*
- *Section 6* games
- *Section 7* conventions and miscellany
- *Section 8* administration and privileged commands
- *Section L* math library functions
- *Section N* tcl functions

Section name is needed if there is same name in multiple sections.

$ man 1 printf Manual page of command printf

$ man 3 printf Manual page of C language function printf()

Commands help, cmd searching

Browsing in manual pages text:

- Movement in text by lines (up, down arrows, keys **j** and **k**)
- Movement in text by pages (**PgDn** and **PgUp** or keys **f** and **b**)
- Searching (**/search_pattern** , key **n** for next occurrence, **N** for previous)
- Manual pages close (key **q**)

On-line manual pages in HTML:

http://linux.die.net/man/

Useful commands:

whatis	prints short command description (manual page header)
apropos	search for commands containing pattern in manual page
info	command info pages (similar to manual pages)

Running commands, applications

Commands and system applications

```
$ ls -l
$ cp file.txt file1.txt
```

Command name or application name

command

Command arguments (modifies command behavior, input information for processing)

User scripts and commands

```
$ ./muj_script
$ ~/bin/my_application
```

Command or script name with full path (absolute or relative)

Redirect standard output

```
$ kwrite &> /dev/null
```

Redirection of standard output is done behind command, arguments.

Running application on background

```
$ gimp &
```

Behind command, arguments and redirections by **& (ampersand)** command is run on background.

Basic commands

File system:

• **pwd**	**prints path to current / working directory**
• **cd**	**change current / working directory**
• **ls**	**prints list of contents of current / working directory**
• mkdir	create directory
• cp	copy directory or file
• mv	move directory or file
• rm	remove directory or file

Investicative commands:

• hostname	prints machine name
• whoami	prints name of logged user
• id	prints ID info of user
• w	prints who is logged and his running command
• ps	prints running processes

Create directory

- **Create directory**

  $ mkdir dir_name

- **Create directory substructure**

  $ mkdir **-p** dir_name1/dir_name2/dir_name3

Copy

- **To copy files and directories use command cp**

$ cp file1 file2

Creates copy of file vytvoří kopii file "file1" s názvem "file2"

$ cp file1 file2 file3 directory1/

Copy files “file1 ", “file2", “file3" to directory “directory1"

$ cp **-r** directory1 directory2

Creates copy of directory “directory1" with new name “directory2"; if directory “directory2" already exists, creates copy of directory “directory1" as subdirectory of directory “directory2"

$ cp **-r** file1 directory2 file3 directory1/

Copy files “file1", “file3" and directory “directory2" to directory “directory1"

Move

- **To move files and directories use command mv**

$ mv file1 file2

Rename file "file1" to "file2"

$ mv file1 file2 file3 directory1/

Move files "file1", "file2", "file3" to directory "directory1"

$ mv directory1 directory2

Rename directory "directory1" to "directory2"; if directory "directory2" exists, then move directory "directory1" to directory "directory2"

$ mv file1 directory2 file3 directory1/

Move files "file1", "file3" and directory "directory2" to directory "directory1"

Mazání

- **K mazání slouží příkaz rm**

$ rm file1

Removes file "file1"

$ rm -r directory1

Removes directory "directory1"

Exercise

1. Download study materials from IS to directory **~/Downloads**.
2. Create subdirectory **pokus** in directory **/scratch/your_login** .
3. Create directory **studmat** in your **home** directory.
4. To directory **studmat** copy study materials from directory **~/Downloads** .
5. Open presentation (Lesson 02) in **okular** software, run okular so that it does not print any standard output information on terminal.
6. Copy presentation to directory **/scratch/your_login/pokus** .
7. Rename presentation to new name **pokus.pdf** in directory **/scratch/your_login/pokus** .
8. Open presentation **pokus.pdf** in okular software, run program on background.
9. Remove presentation in directory **~/Downloads** .

Try to use shortcuts:

- **Auto complete (key TAB)**
- **Text copy (select by mouse / insert by wheel)**
- **history**

Remote login

- ssh
- Encryption
- Recursive login

Remote login

There is number of protocols / commands for remote login (rsh, XDMCP, etc.), most used and most **secure** is protocol / command **ssh** (secure shell).

[] - optional

Syntax:

```
$ ssh [user@]hostname [command]
```

User name;
If name is not given, curent user name is used

Machine name

Command to be submitted to remote machine, if omitted command line on remote machine is activated

Examples:

$ ssh wolf.wolf.inet

$ ssh wolf01.wolf.inet w

Logout:

Remote interactive login is closed by command **exit**.

First remote login

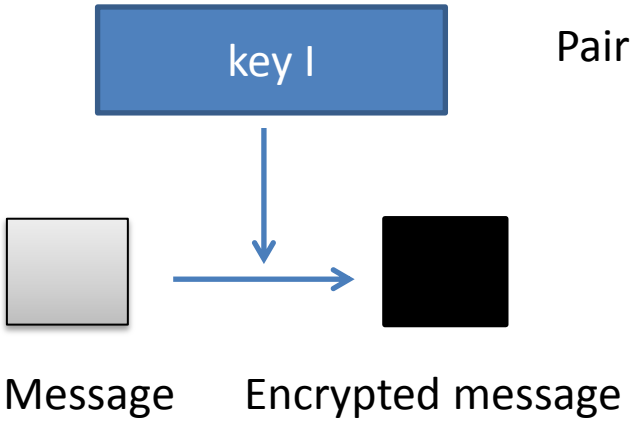
```
[kulhanek@wolf01 ~]$ ssh wolf02

The authenticity of host 'wolf02 (10.251.28.102)' can't be established.
ECDSA key fingerprint is 1f:9d:f3:d3:1d:24:28:12:56:30:99:ef:2d:68:d2:cf.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'wolf02,10.251.28.102' (ECDSA) to the list of known hosts.

[kulhanek@wolf02 ~]$
```

On first remote login user has to confirm authenticity of remote machine.

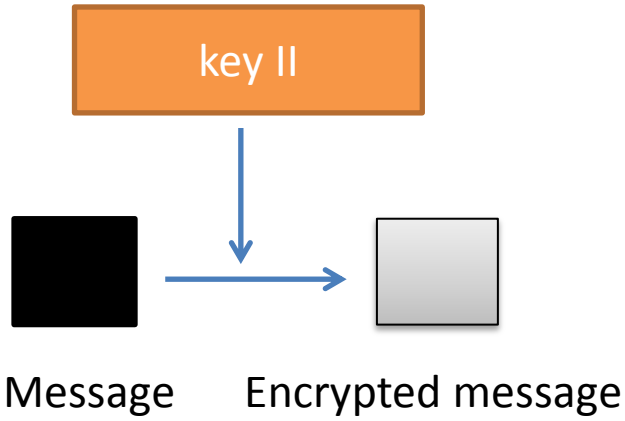
In secure network remote machine fingerprint can be accepted without verification.

In the Internet it is better to verify fingerprint by independent way (receive fingerprint from remote machine admin by mail).

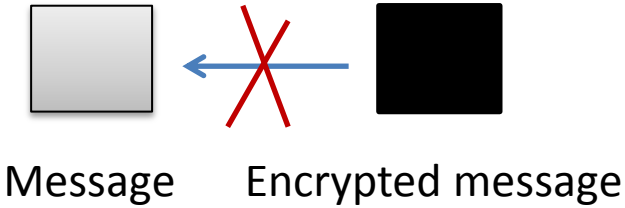
Asymmetric encryption



Pair of encryption keys



Decryption to original message by key used for encryption **is not feasible.**

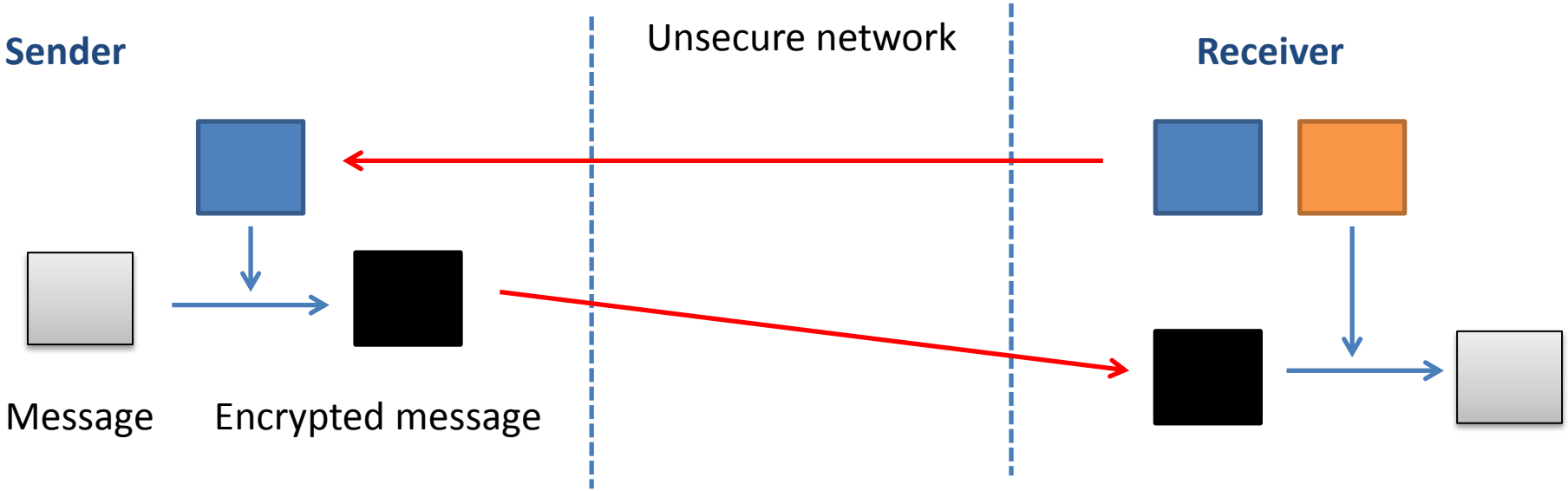


Asymmetric encryption, use I

Public key | Private key | Pair of encryption keys

Secure message transfer:

1. Obtain receiver public key.
2. Encrypt message with receiver public key.
3. Message transfer over unsecure network.
4. Receiver decrypts original message by his private key.



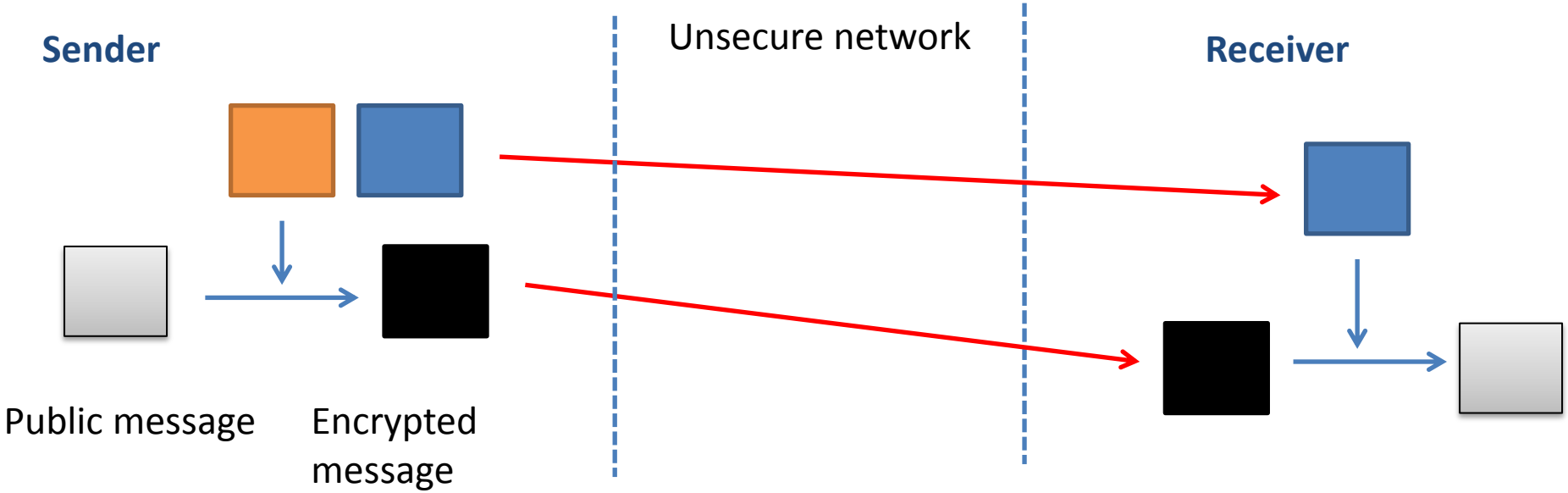
Anybody who knows receiver private key can decrypt original message!

Asymmetric encryption, use II

Public key | Private key — Pair of encryption keys

Authentication of public message sender:

1. Encryption of message by sender private key
2. Receiver obtains encrypted message and sender public key.
3. Receiver decrypts message by sender public key.



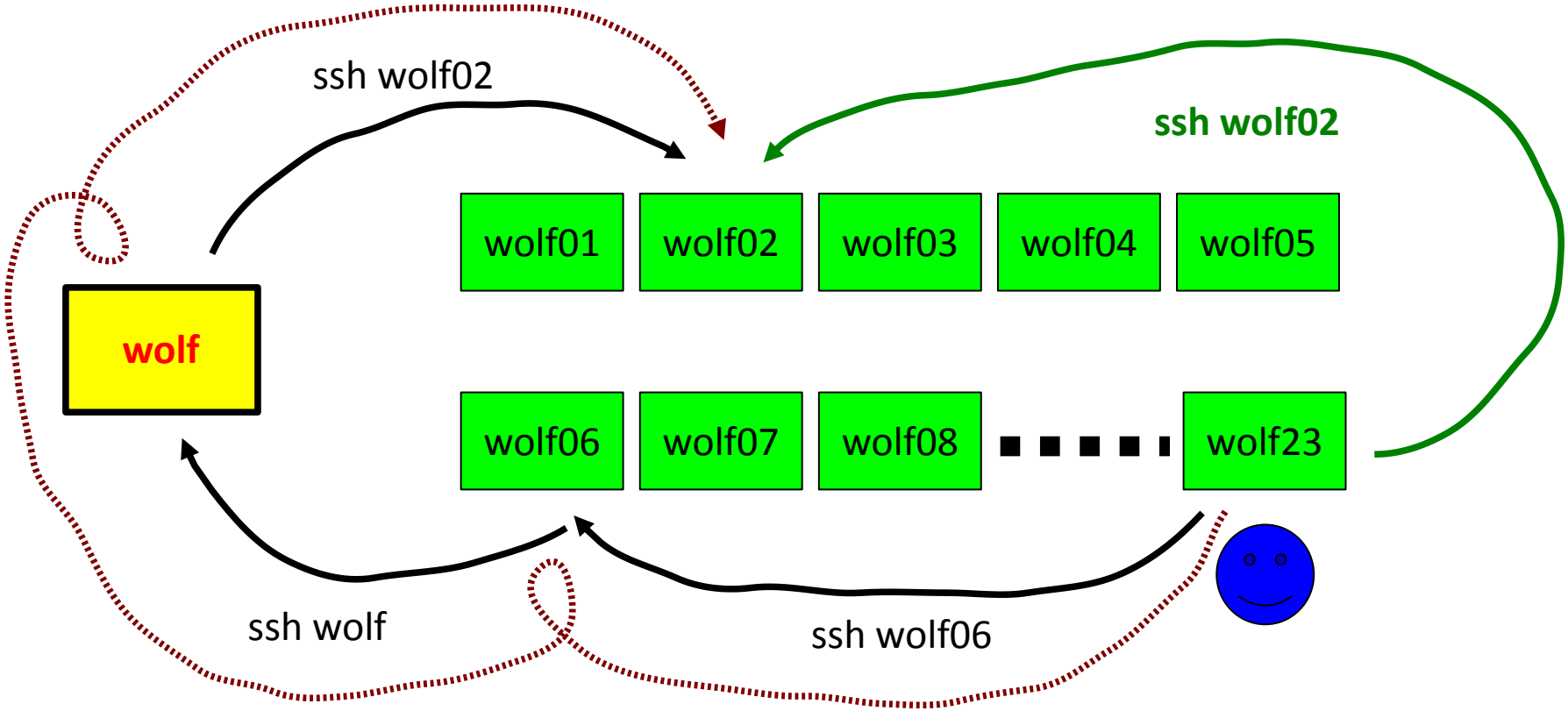
Anybody who steels sender private key, can pretend to his identity!

Exercise

1. Log on to remote machine **wolf01.wolf.inet**
2. Print all connected users by command **w**.
3. Logout from machine wolf01.wolf.inet.
4. Print all users logged on wolf01.wolf.inet without **interactive** login to node.

Remote login

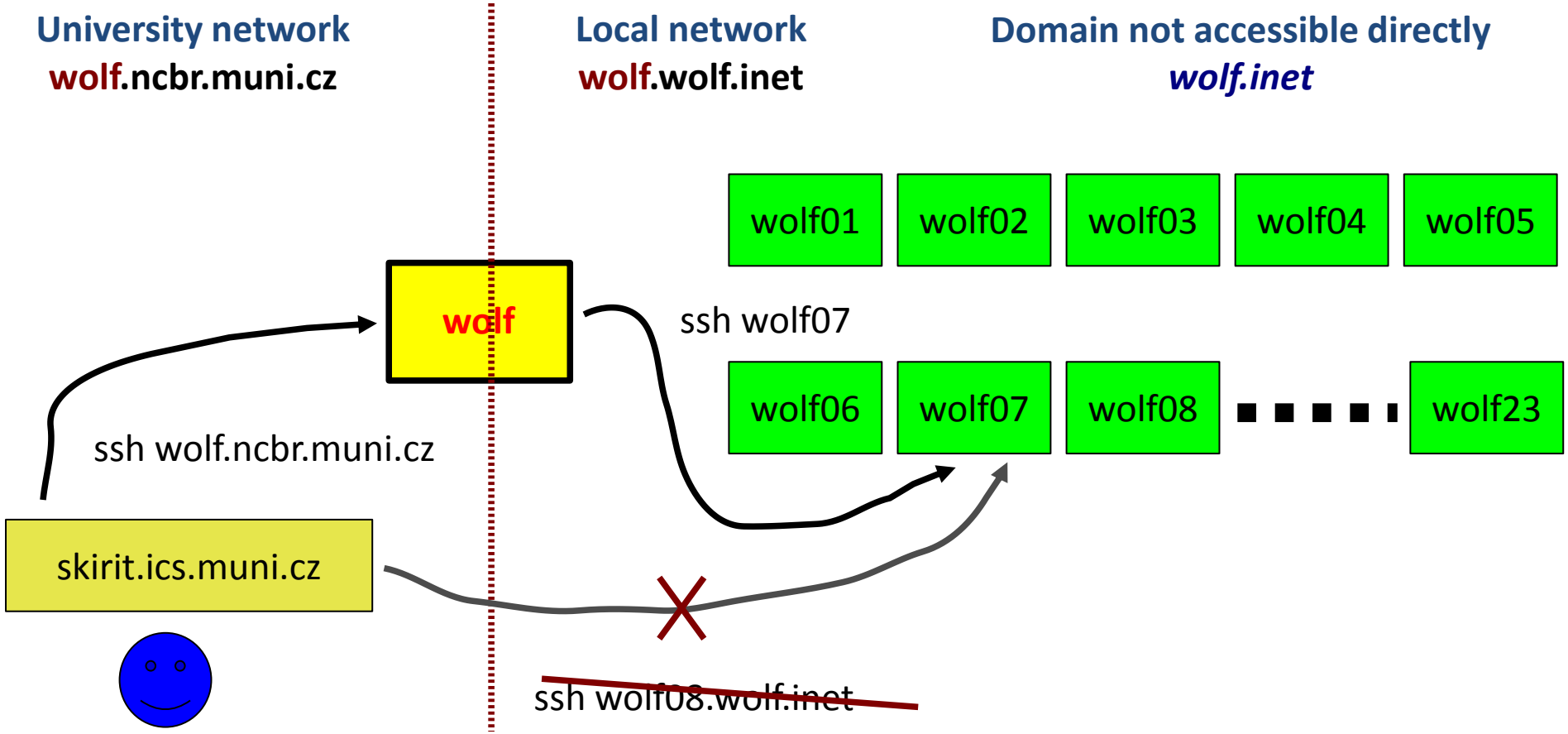
Command ssh can be used for **recursive remote login**.



Each new remote login level increases **overhead costs**, thus we use the **most direct possible** remote login

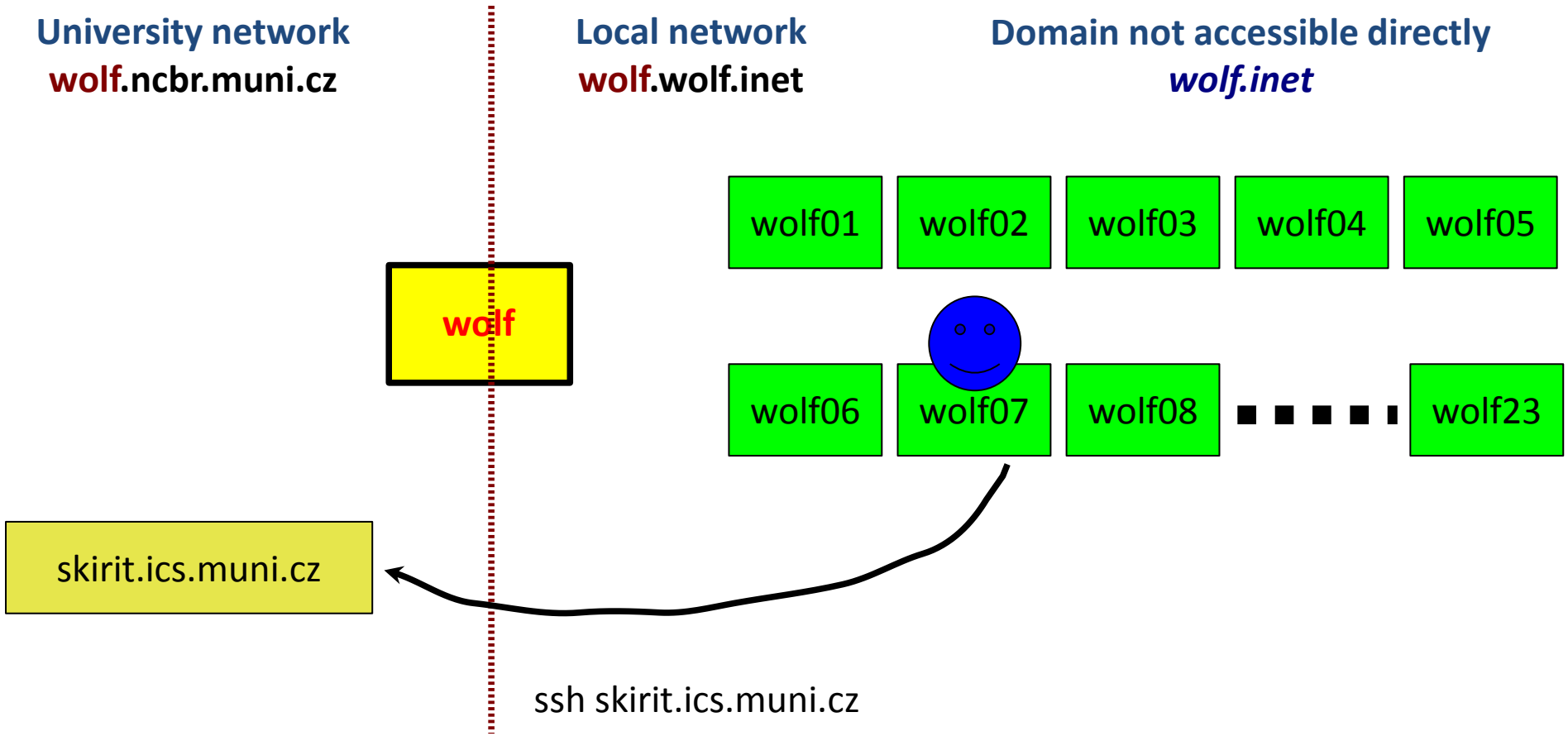
Remote login

Recursive remote login is **necessary** for access of computers in private networks.

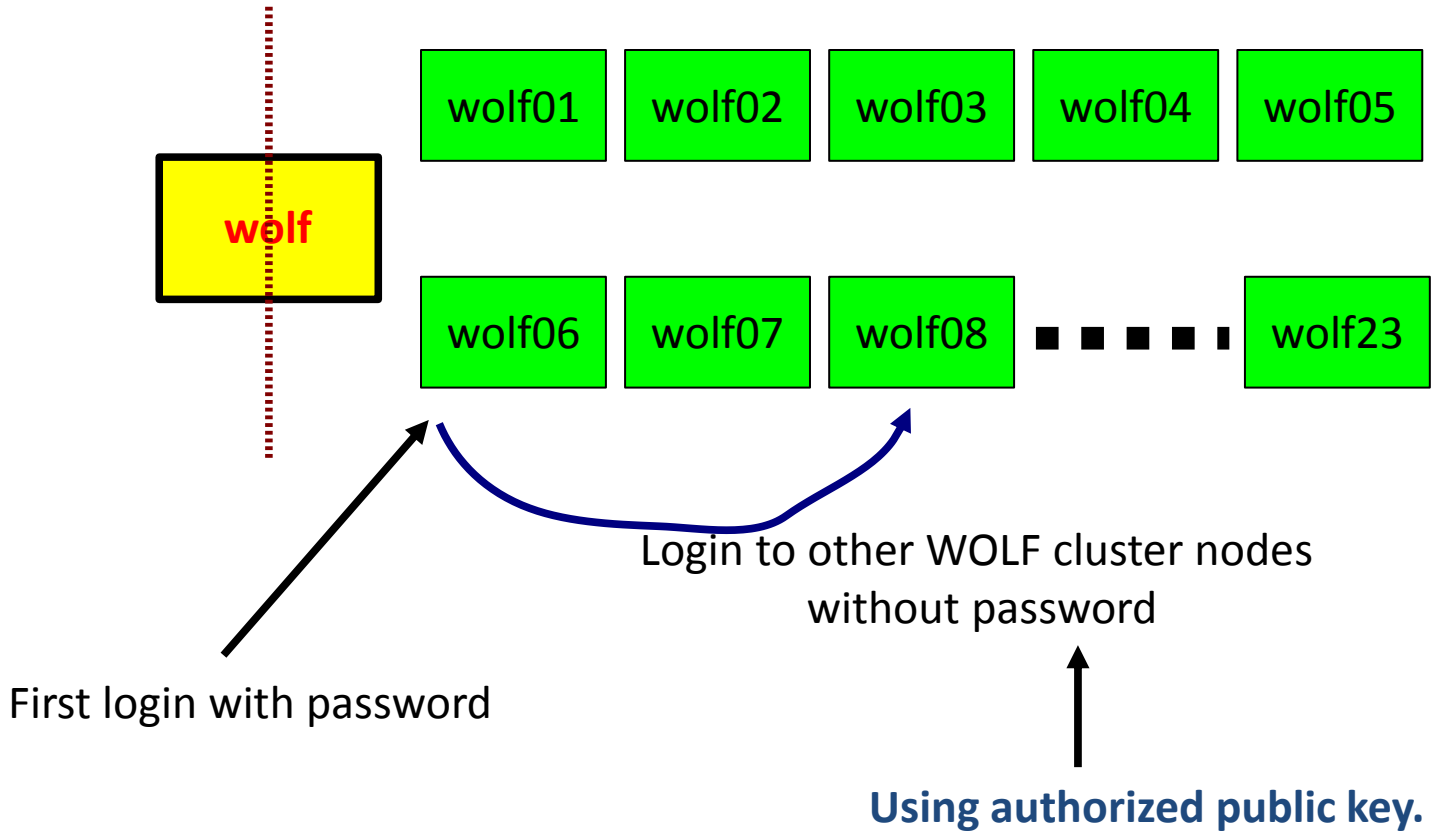


Remote login

Remote login from local private networks to machines located in public network can be done directly.



Remote login without password



There are also different techniques with similar extent:
system Kerberos (http://web.mit.edu/Kerberos/)

Remote login without password

1. Create private and public key pair:

No input!

```
[kulhanek@wolf01 ~]$ cd .ssh
[kulhanek@wolf01 .ssh]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/kulhanek/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/kulhanek/.ssh/id_rsa.
Your public key has been saved in /home/kulhanek/.ssh/id_rsa.pub.
The key fingerprint is:
e9:07:0b:fc:17:23:b3:c5:1a:8a:0c:1a:98:8f:fe:28 kulhanek@wolf01.wolf.inet
```

2. Paste your public key to list of authorized keys:

```
[kulhanek@wolf01 .ssh]$ cat id_rsa.pub >> authorized_keys
```

Advantages:

- No need to input password each time
- More secure usage of ssh and scp commands in scripts.
- Faster work

Disadvantages:

- In case of loosing one account, all nodes with authorized keys can be accesed.

Description: man ssh

Exercise

1. Activate **remote login without password** in WOLF cluster.
2. **Verify** that remote connection works. Connetc to node wolf01.
3. Try recursive remote login within WOLF cluster.
4. Monitor who is logged on your machine.