Nuclear Engineering computing seminar Ondřej Chvála <ochvala@utk.edu>

Introduction to practical SSH

and a few selected notes regarding computer security

Lecture overview

- Shared/public key cryptography
- Using SSH with keys
- Secure copy SCP
- SSH tips and tricks: per session config, remote execution, output redirection, tunneling, ssh filesystem, SOCKS proxy
- Useful screen command, job control, nohup and disown

SSH!
DON'T GIVE AWAY YOUR PASSWORD TO ANYONE WHO MIGHT BE LISTENING!

Septemer 5 2013

The need for secure computing

- **Internet was designed as plain-text based**, since 1960s computers were slow, and the DARPANET lines were physically secured.
  - Security of communication has to be added by the Netizen!
- Now computers are fast, and who knows who is listening.
  - **Encrypt everything**, including hard drives: gadgets with personal information (i.e. SSN in tax returns) get stolen.
  - **Disable FireWire** in BIOS!
- Cryptography can and should be used in general: email reading, web browsing, data storage. Some examples on next slide.
- Telnet & FTP – use clear plain text passwords, clear plain text sessions.
  - Use ssh and scp – encrypted credentials and session data.
  - The problem is much more wide spread, see next slide.

Practical suggestions

- **Common Internet services** such as HTTP (standard port 80, web browsing), POP3 & IMAP (ports 110 & 143, email reading), SMTP (25, email sending), NNTP (119, news reading), and many others send **credentials** (user names and passwords) in **plain text!**

- Make sure you ALWAYS use secure alternatives: HTTPs, POP3s, IMAPs, SMTPs, NNTPs, etc. which run the original protocol over **SSL/TLS**: point to point secured transport layer. Generally they use different ports (443, 995, 993, 465, 563). See /etc/services file on a UNIX box (usha).

- Note regarding Web: Even if passwords are sent encrypted over HTTP (banks, e-shops, and web2.0 such as Facebook) **your session can be hijacked** by anyone on local network or between you and the web server: password changed, money and identity stolen, etc. Fortunately HTTPs is often enforced nowadays, but do not bet on it →

- **Install "HTTPs Everywhere" extension in your browser to be sure.**

Interlude: Practical suggestions II

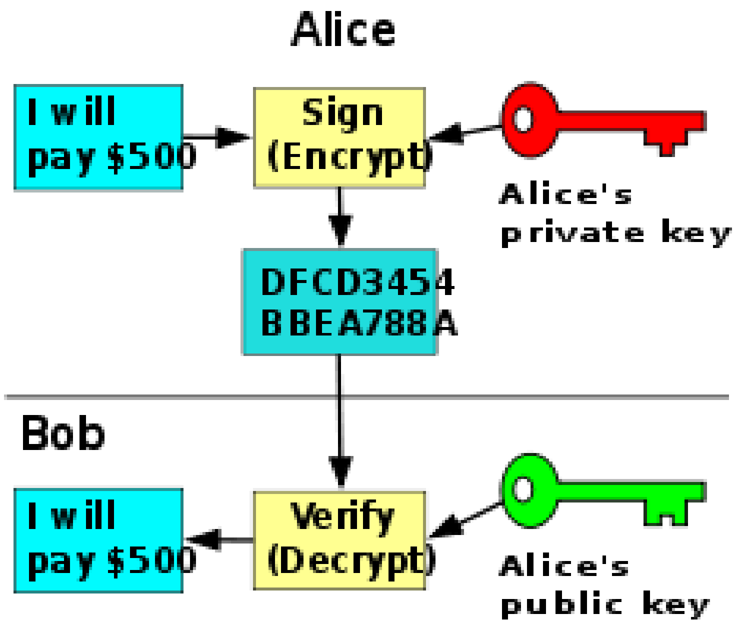
- **No proprietary OS can ever be really secured.**
  - Consider legal and personal repercussions if all sensitive information stored on your machines gets public.
  - This includes: personally identifiable information (i.e. tax returns), export controlled codes & data, trade secrets or classified information you may use for research.
  - You may be a vector for phissing attacks towards national labs, such as you worked for an ORNL employee as a summer inter.
- Anything you share via UTK email, Web based email services, or Web2.0 services (Facebook, Twitter, …) consider as **public**, unless you take precautions such as PGP.
- (Cyber) security approach needs to be appropriate to expected attack vectors. You are prime targets due to nature of your work, research, access, and associations!

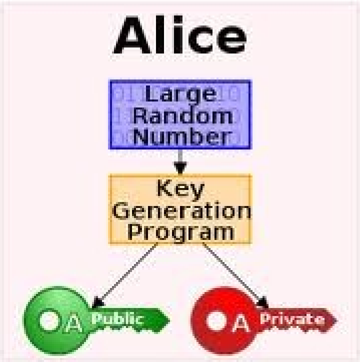
Shared key cryptography

- Both sender and receiver share the same key.
  - The only encryption method publicly available until 1976.
- SKC a.k.a symmetric key cryptography.
  - Block ciphers – blocks of data transformed by algorithm + key.
    - Examples: EAS, DES, 3DES, RC5
  - Stream ciphers – each character in the message is transformed using a pseudo-random cipher digit stream seeded by the key.
    - Examples: RC4, cell phones use A5/1, A5/2, or A5/3

Public key cryptography

- Based on trap-door mathematics a.k.a. one-way functions
  - Described by Stanley Jevons (1835 – 1882) of Jevons paradox
  - Example: factorization of very large numbers, RSA algorithm (1977)
  - Take two large primes P, Q: P*Q => R is trivial, but R => P*Q is hard
- PKC a.k.a asymmetric key cryptography: public & private key pairs
  - Public key encrypts data
  - Private key decrypts data



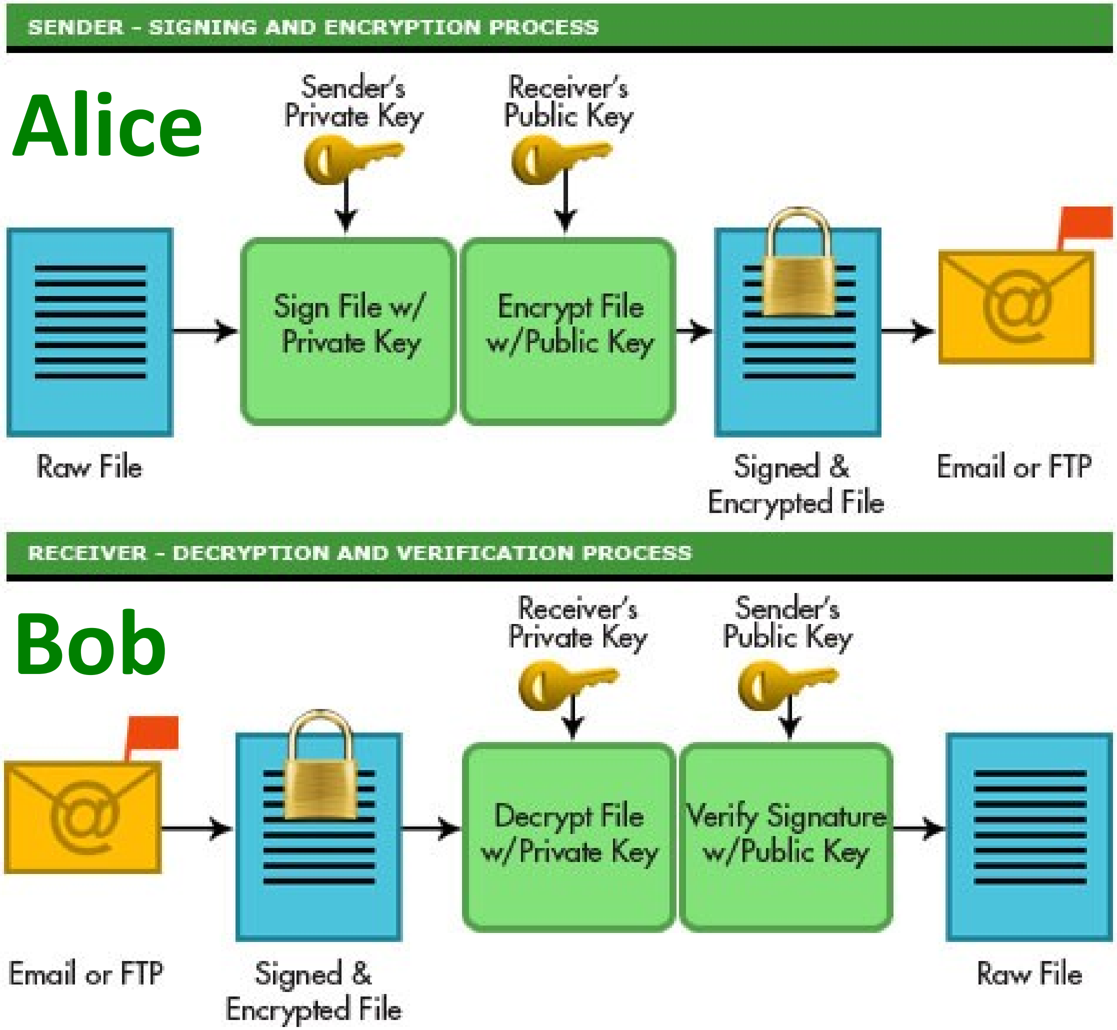


OpenPGP – RFC #4880

- PGP – Pretty Good Privacy, created by Phil Zimmerman in 1991
  - Signing and encrypting with 2 key pairs: encrypt and verify sender

- Sender Alice
  - signs with her private key
  - encrypts with Bob's public key

- Recipient Bob
  - decrypts with his private key
  - verifies sender using Alice's public key



SSH: Secure SHell

- Secure replacement for remote shells, with other benefits:
  - compression, secure file copy, secure remote GUI, port forwarding.
- Server-client architecture: server/daemon on server, client connects.
  - Server listens on TCP port 22 per standard, can be changed.
- Authenticates the session by public key cryptography, generates random shared key for each session, uses the shared key to encrypt the session data (faster).
- 1995: SSH-v1 designed by Tatu Ylönen at Helsinki University in Finland
  - This version is vulnerable, and should be disabled by default.
- 2006: SSH-v2 adopted by IETF as a new standard.
- Most popular implementation is OpenSSH developed by the OpenBSD project.

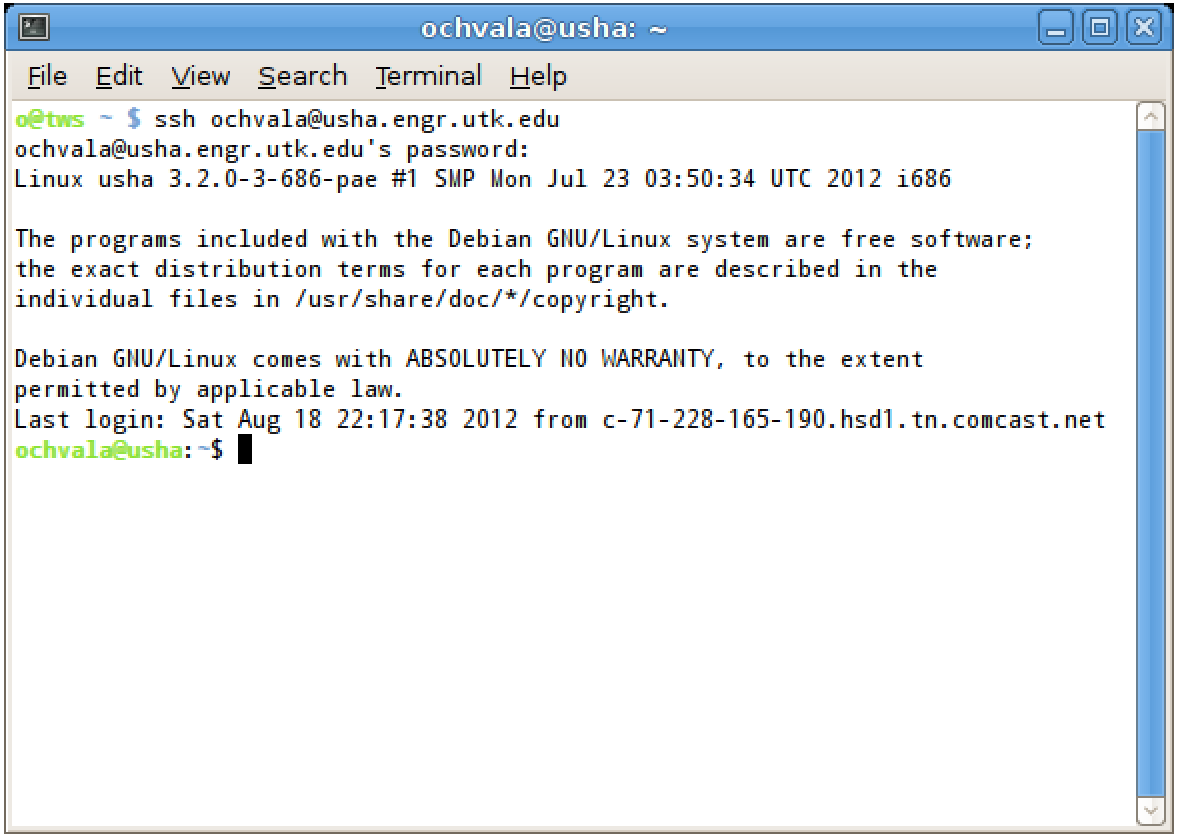
Connecting to Usha, overview

- Use **ssh client** to connect to an **ssh server**, a daemon on remote box
  - Linux/Mac: ssh command, Windows: PuTTY
- **Generate** public/private key pair on your local machine
  - Linux/Mac: ssh-keygen command, Windows: PuTTYgen
- **Copy** the public key to the remote machine
  - Linux/Mac: scp command, Windows: WinSCP; Filezilla GUI for all OS
- Configure a **shortcut** on your local machine for NEcluster
  - Linux/Mac: edit file ~/.ssh/config, Windows: save session in PuTTY
  - Enable X11 forwarding, Windows: install X11 server

Practical SSH on Linux/Mac

- Simplest connection: *ssh <user>@<machine>*
  - Type password when prompted

For Usha <machine> = **usha.engr.utk.edu**



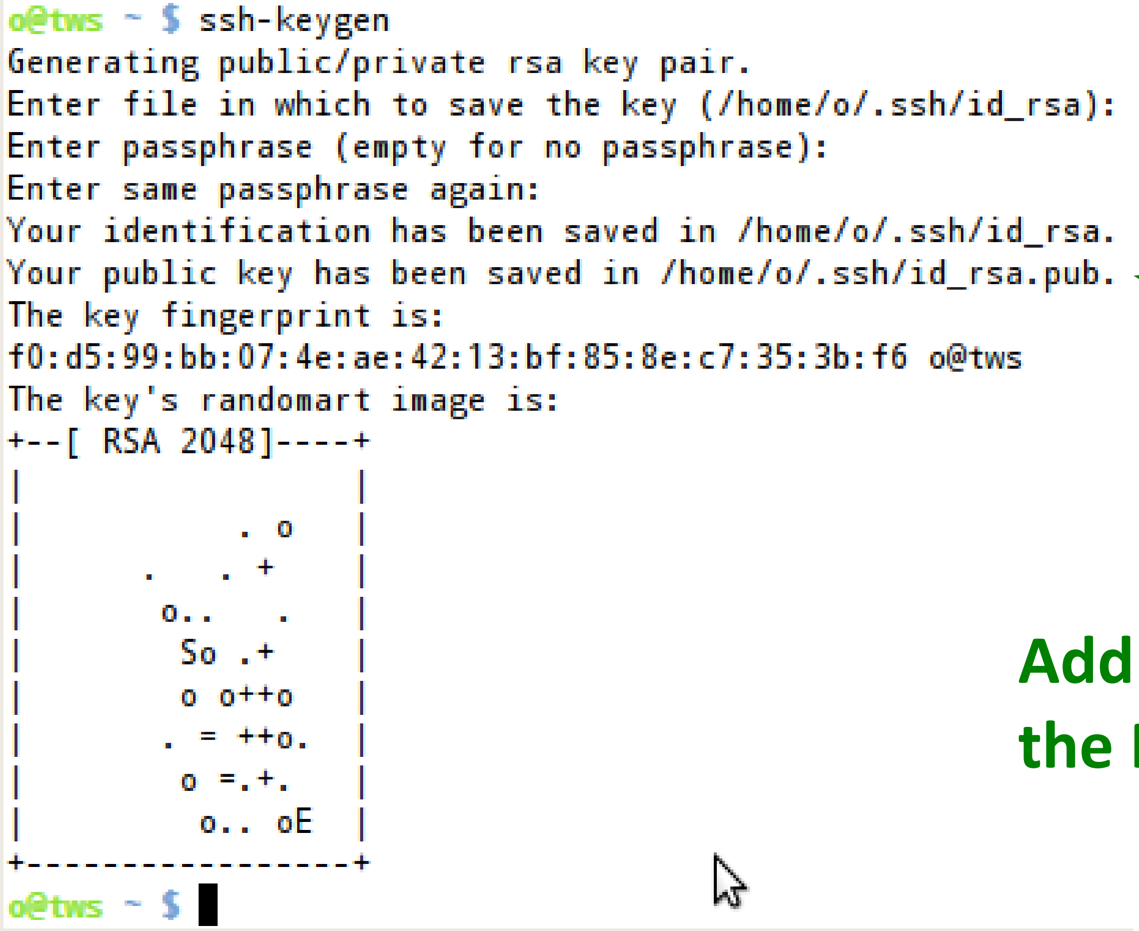
```
o@tws ~ $ ssh ochvala@usha.engr.utk.edu
ochvala@usha.engr.utk.edu's password:
Linux usha 3.2.0-3-686-pae #1 SMP Mon Jul 23 03:50:34 UTC 2012 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 18 22:17:38 2012 from c-71-228-165-190.hsd1.tn.comcast.net
ochvala@usha:~$
```

- NOTE: *man ssh* for command-line options and other tricks

- Problem: one has to remember the password.
  Often either **bad password** (weak or shared with other accounts) or **bad password management** (written on a stick-it note).

Using keys

- Generate key: *ssh-keygen*
  - Generates public & private key pair



```
o@tws ~ $ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/o/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/o/.ssh/id_rsa.
Your public key has been saved in /home/o/.ssh/id_rsa.pub.
The key fingerprint is:
f0:d5:99:bb:07:4e:ae:42:13:bf:85:8e:c7:35:3b:f6 o@tws
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|           . o   |
|      .   . +    |
|       o..   .   |
|        So .+    |
|        o o++o   |
|       . = ++o.  |
|        o =.+.   |
|         o.. oE  |
+-----------------+
o@tws ~ $ 
```

~/.ssh/id_rsa
Private key – keep on your computer!

~/.ssh/id_rsa.pub
Public key – copy over to the computer you want to connect to.

Add into ~/.ssh/authorized_keys on the REMOTE machine (Usha)

NB: See *man ssh-keygen* for options such as key length, changing passphrase, validity intervals, change options related to the key, etc.

Copying files using *scp*

- To copy files: *scp <local_file>   <user>@<machine>:<remote_path>* or *scp <user>@<machine>:<remote_file>   <local_path>*

  NB: dot "." means current directory

  NB: *man scp* for options. Ex.: *-r* copy dir., *-p* preserve attributes

- To copy the public key to usha using scp:

```
o@tws ~ $ scp .ssh/id_rsa.pub ochvala@usha.engr.utk.edu:
id_rsa.pub                        100%  387     0.4KB/s   00:00
o@tws ~ $
```

- Connect to usha, create ~/.ssh/, add id_rsa.pub into file ~/.ssh/authorized_keys

```
ochvala@usha:~$ mkdir .ssh
ochvala@usha:~$ chmod 700 .ssh
ochvala@usha:~$ cat id_rsa.pub >> .ssh/authorized_keys
```

Using keys (2)

- **NB:** Easier way which works with OpenSSH: *ssh-copy-id <user>@<box>*
- After we added the keys, *ssh <user>@<machine>* works without password. Still needs to unlock the key by passphrase.
  (*ssh-agent* can help with that)
- Potential issues with manual copying: access rights: *chmod 700 ~/.ssh*
  - See: *man chmod*

```
ochvala@usha:~$ ls -la .ssh/
total 16
drwx------ 2 ochvala ochvala 4096 Sep  5 16:43 .
drwx------ 7 ochvala ochvala 4096 Sep  5 16:44 ..
-rw-r--r-- 1 ochvala ochvala  387 Sep  5 16:42 authorized_keys
-rw-r--r-- 1 ochvala ochvala  222 Sep  5 16:42 known_hosts
```

- File known_hosts contains public keys of machines you connected to.

ssh-agent

- To keep ssh keys unlocked, i.e. avoid typing passphrases, use *ssh-agent*
- Most distributions start ssh-agent with X session (“GUI”), so you dont need to worry about that. Otherwise run: *ssh-agent bash* to open new shell with ssh-agent wrapped around it.
- To add keys: *ssh-add <private key file>*
  - Options: -l lists keys in memory, -D deletes all identities;
    - *man ssh-add*
- **Agent forwarding – chaining ssh authorization**
  - Laptop (has my private key) → server1 → server2 → … → serverN works as long as each server has the relevant **public key** in ~/.ssh/authorized_keys
  - Magic: ssh daemons running on intermediate machines act as forwarding agents!

Lets make life easy: ~/.ssh/config

- Instead of typing the <user>@<machine> and command line options, place all into **~/.ssh/config** and use a nickname. See: *man ssh_config*

```
Compression yes
ForwardX11 yes
ForwardAgent yes
ForwardX11Trusted yes

Host usha
        HostName usha.engr.utk.edu
        User ochvala
        IdentityFile ~/.ssh/id_rsa
Host cl
        HostName necluster.engr.utk.edu
        User ondrejch
        IdentityFile ~/.ssh/id_rsa.UTKNEcluster
```

Default options for all sessions

Per-host configurations

- Instead *ssh -XYC ochvala@usha.engr.utk.edu* much simpler: *ssh usha*. Also *scp <local_file> usha:<remote_path>* etc.

Remote execution & I/O redirection

- Run program on a remote machine: *ssh usha <what_to_run>*
  - Example: ssh usha w
- Redirect output: *ssh usha tar -tzf MyArchive.tgz > ListOfFiles.txt*
  - This will list remote archive content into local file.
- Redirect input: *ssh usha tar -xz < LocalArchive.tgz*
  - Extracts LocalArchive.tgz on usha
- Pipes work in and out: *cat myfile.txt | ssh usha lpr*
  - Will print myfile on usha

Mounting remote filesystems via sshfs

- SSH-FS = ssh file system. User-space implementation of file system client over ssh. Works on any system you can ssh to.
- Typically sshfs has to be installed: *sudo apt-get install sshfs*
  - Usha has it. See *man sshfs* for all options.
  - Using: *sshfs <user>@<host>:[remote_path] <mount-directory>*

```
o@usha:~$ mkdir ~/clusterhome
o@usha:~$ sshfs cl: clusterhome
o@usha:~$ df -h
Filesystem     Size  Used Avail Use% Mounted on
rootfs          38G  9.2G   27G  26% /
[..]
/dev/sdc2      107G  1.8G  100G   2% /home
cl:            3.6T  2.2T  1.3T  65% /home/o/clusterhome
```

- Note: user has to be member of fuse group:

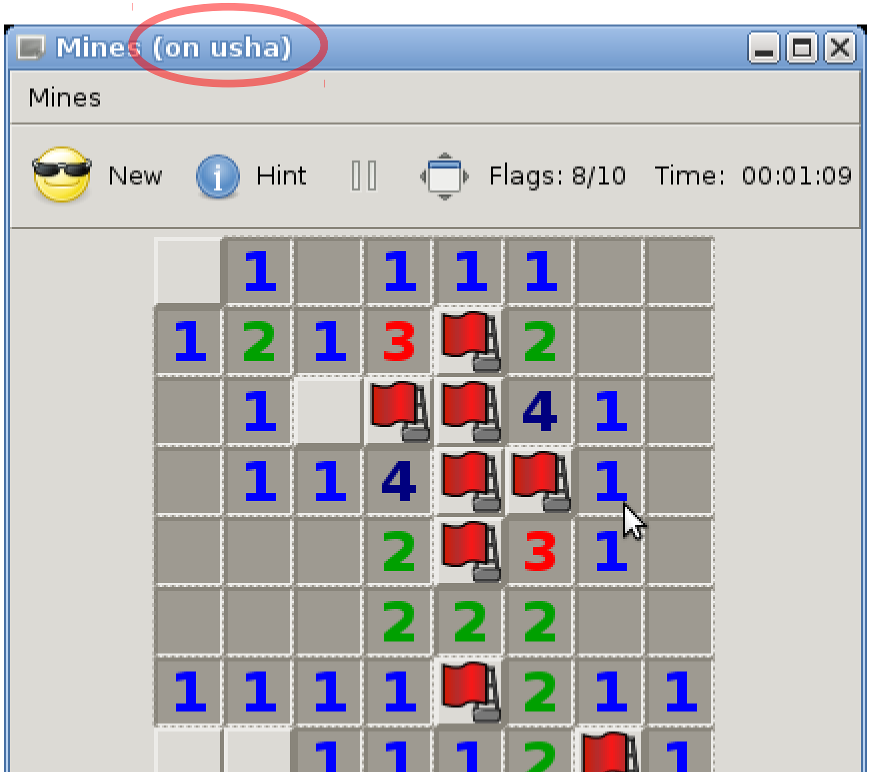
  sudo usermod -a -G fuse <username>

X11 forwarding

- If allowed, ssh will automatically create a fake X server, and send all X11 traffic via an encrypted tunnel.

```
ochvala@usha:~$ env | grep DISPLAY
DISPLAY=localhost:13.0
ochvala@usha:~$ gnomine &
```

- These calls will be captured by local X server: voilà, remotely run graphical programs.
- Linux, Mac, *BSD, … come with native Xservers. There are free Xservers for Windows, see previous seminar slides for details.



Local port forwarding

- Say there is an web server behind a firewall at UTK: intranet.utk.edu
- Create a tunnel via usha:
  - *ssh usha -L 8888:intranet.utk.edu:80*
  - Connect to intranet.utk.edu by browsing to http://localhost:8888
- Say you have to connect to unsecure service provider at UTK, such as IMAP (versus IMAPs). You can wrap the connection in an ssh tunnel:
  - *ssh usha -L 8143:unsecure.utk.edu:143*
  - Point your mailer to localhost, port 8143
- In general: ssh <ssh-server> -L <local-port>:<target-box>:<target-port>
- Config file option:

```
Host intranet
HostName usha.engr.utk.edu
LocalForward 8888 intranet.utk.edu:80
```

Remote port forwarding

- Inverse situation: how to make a local port available on remote box.
- Say a firewall blocks all incoming connections.
- Create a tunnel at usha, “home” is alias for home machine.
  - *ssh home -R 8889:intranet.utk.edu:80* (executed at usha)
  - This will connect to home machine creating an ssh tunnel, waiting for incoming requests to port 8889 to be re-routed though the tunnel to intranet.utk.edu:80
  - Now you can connect to intranet.utk.edu from home by browsing to http://localhost:8889 (at home)
- Config file option (at usha):

```
Host remote-intranet
HostName home.dyndns.org
RemoteForward 8889 intranet.utk.edu:80
```

Dynamic port forwarding (SOCKS proxy)

- General version of local port forwarding, which maps all ports.
- Useful for connecting to Internet at untrusted network (hotel, mall, …)
- At local machine create dynamic port forward session:
  - *ssh usha -D 9999*
  - At local machine open Firefox, Menu/.../Connection settings
    Manual Proxy Configuration, fill SOCKS fields
    SOCKS Host: localhost
    SOCKS Port: 9999
  - Voilà, browsing via a secure channel (up to usha)!

Notes on port forwarding

- Usha is used as an example of a machine running the ssh daemon, any other will do as well.

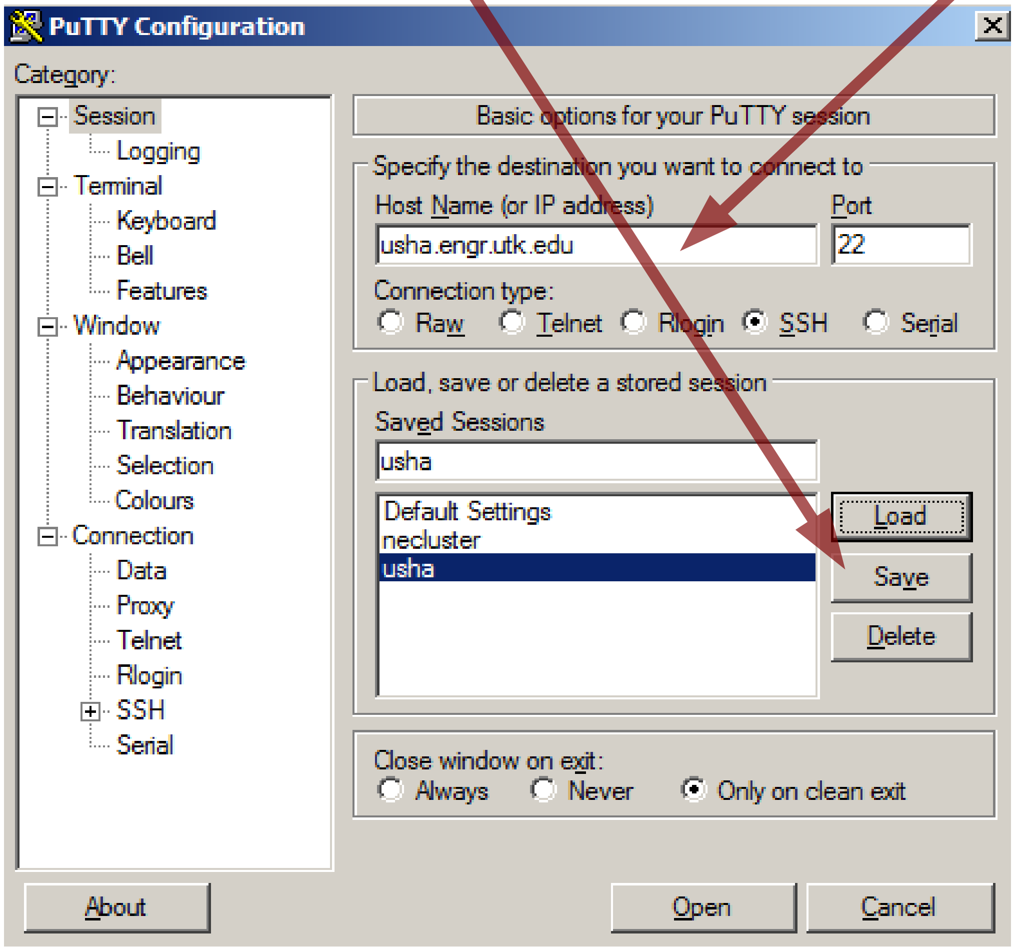
- Port numbers in examples are arbitrary, however:
  - You would need to log in as root if you want services to listen on a port < 1024.
  - Remember to open necessary ports on any firewall between your machine and usha.
  - Unfortunately you can only forward services running on TCP, but there is a way to forward UDP through SSH using netcat.

- **Make sure you are not breaking Acceptable Use Policy or other applicable cybersecurity rules. In particular national labs (ORNL) prohibit punching holes in firewalls, and you will get caught!**

Practical SSH on Windows

- Download ssh client for Windows named PuTTY: (Google PuTTY) http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

- Put “usha.engr.utk.edu” into Host Name, enable X11 forwarding, save session



Specify the destination you want to connect to

Host Name (or IP address): usha.engr.utk.edu

Port: 22

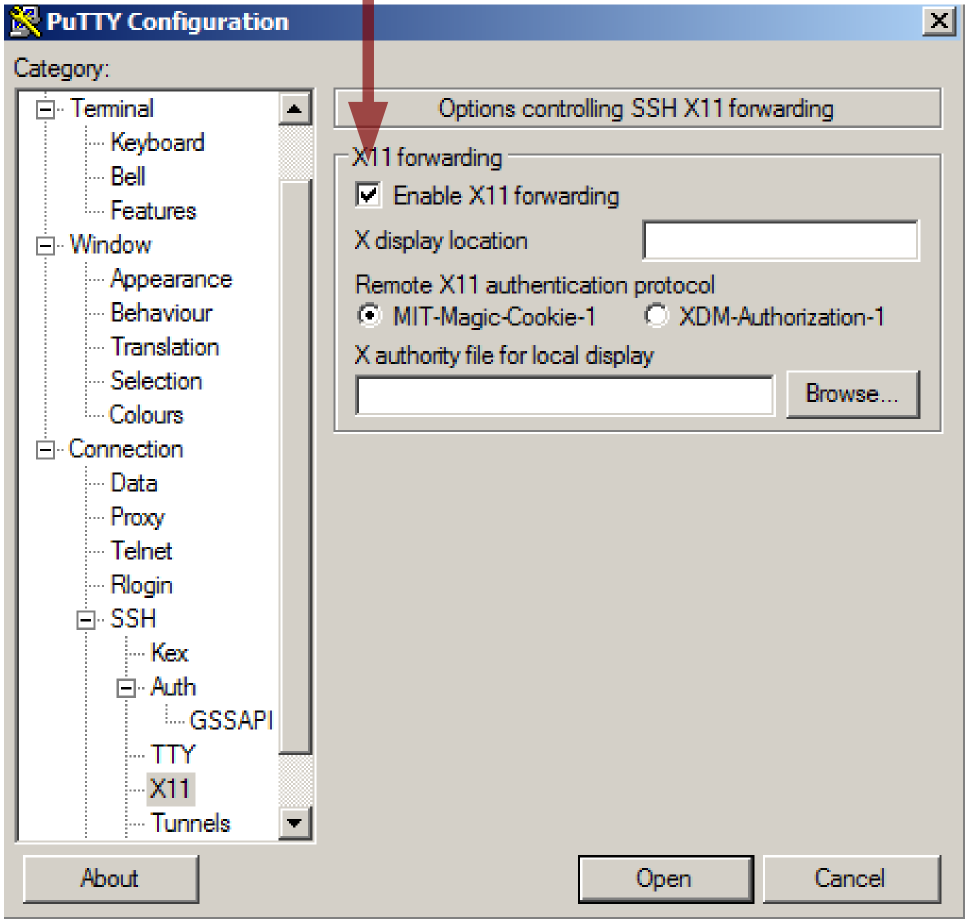
Connection type: ○ Raw ○ Telnet ○ Rlogin ● SSH ○ Serial

Load, save or delete a stored session

Saved Sessions: usha

Default Settings
necluster
usha

Close window on exit: ○ Always ○ Never ● Only on clean exit



X11 forwarding

☑ Enable X11 forwarding

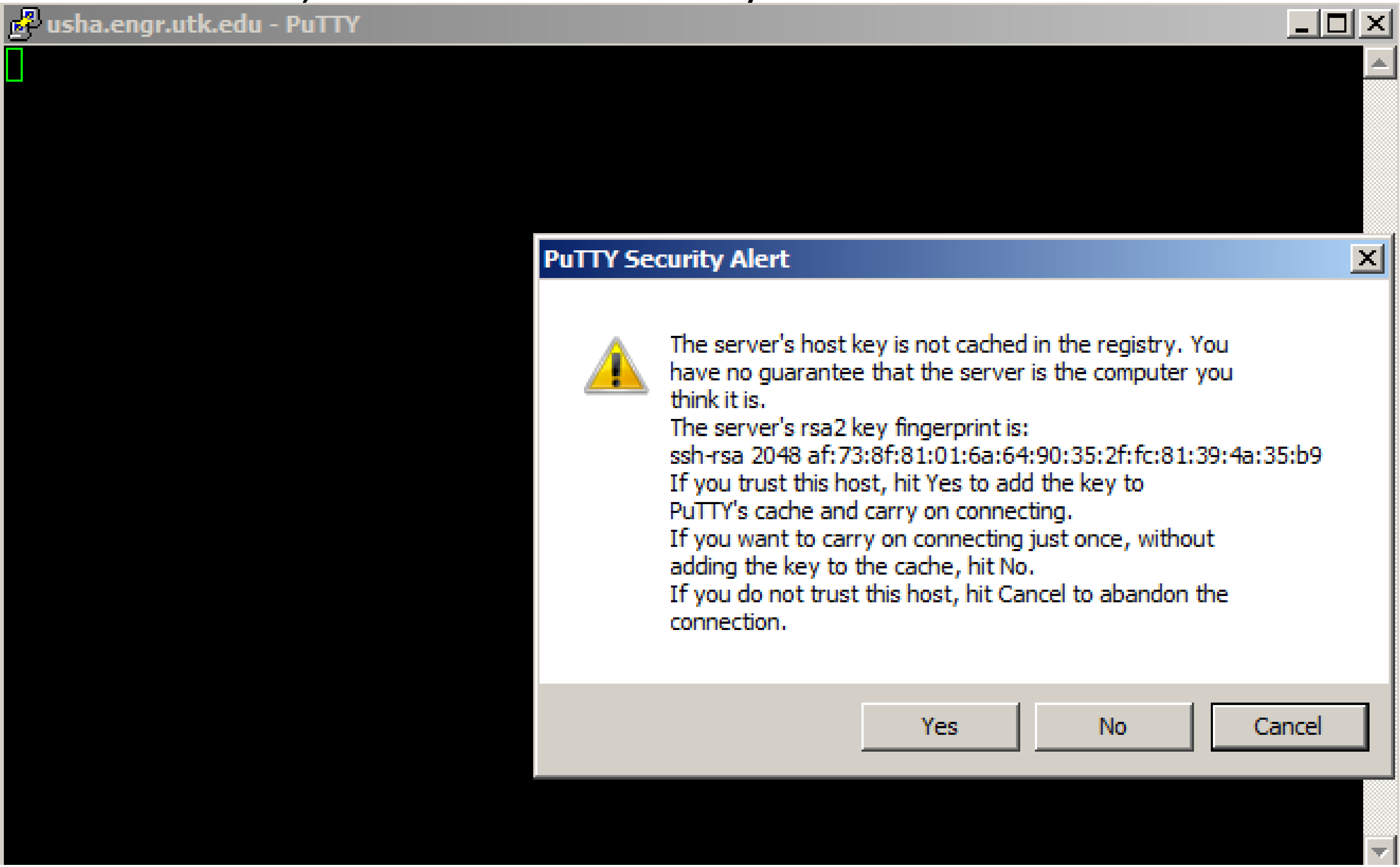
X display location:

Remote X11 authentication protocol: ● MIT-Magic-Cookie-1 ○ XDM-Authorization-1

X authority file for local display:

Windows, Connecting to Usha (2)

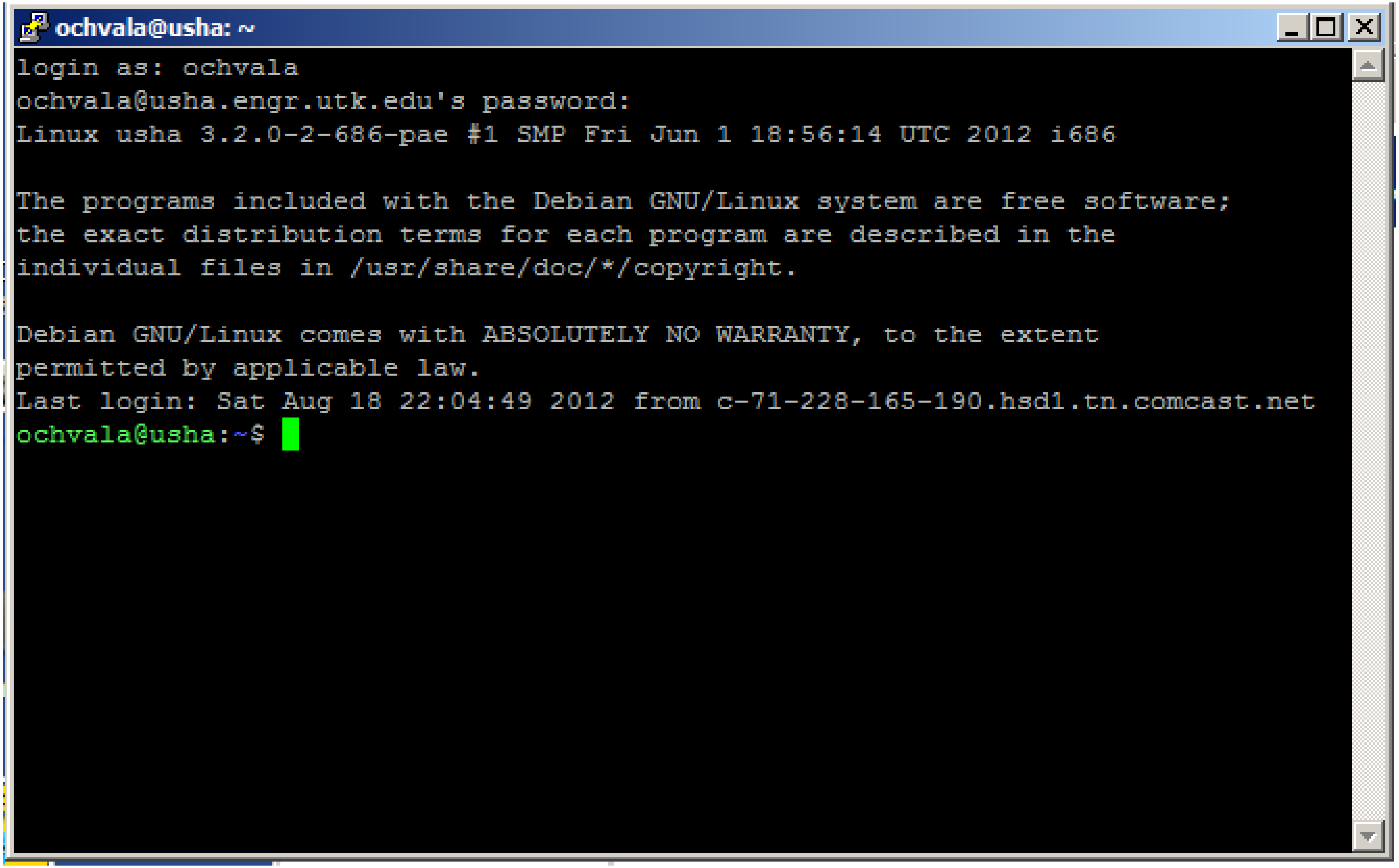
- Click Connect, confirm ssh server key:



The server's host key is not cached in the registry. You have no guarantee that the server is the computer you think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 2048 af:73:8f:81:01:6a:64:90:35:2f:fc:81:39:4a:35:b9
If you trust this host, hit Yes to add the key to PuTTY's cache and carry on connecting.
If you want to carry on connecting just once, without adding the key to the cache, hit No.
If you do not trust this host, hit Cancel to abandon the connection.

Windows, Connecting to Usha (3)

- Type your username and password, and you are in:

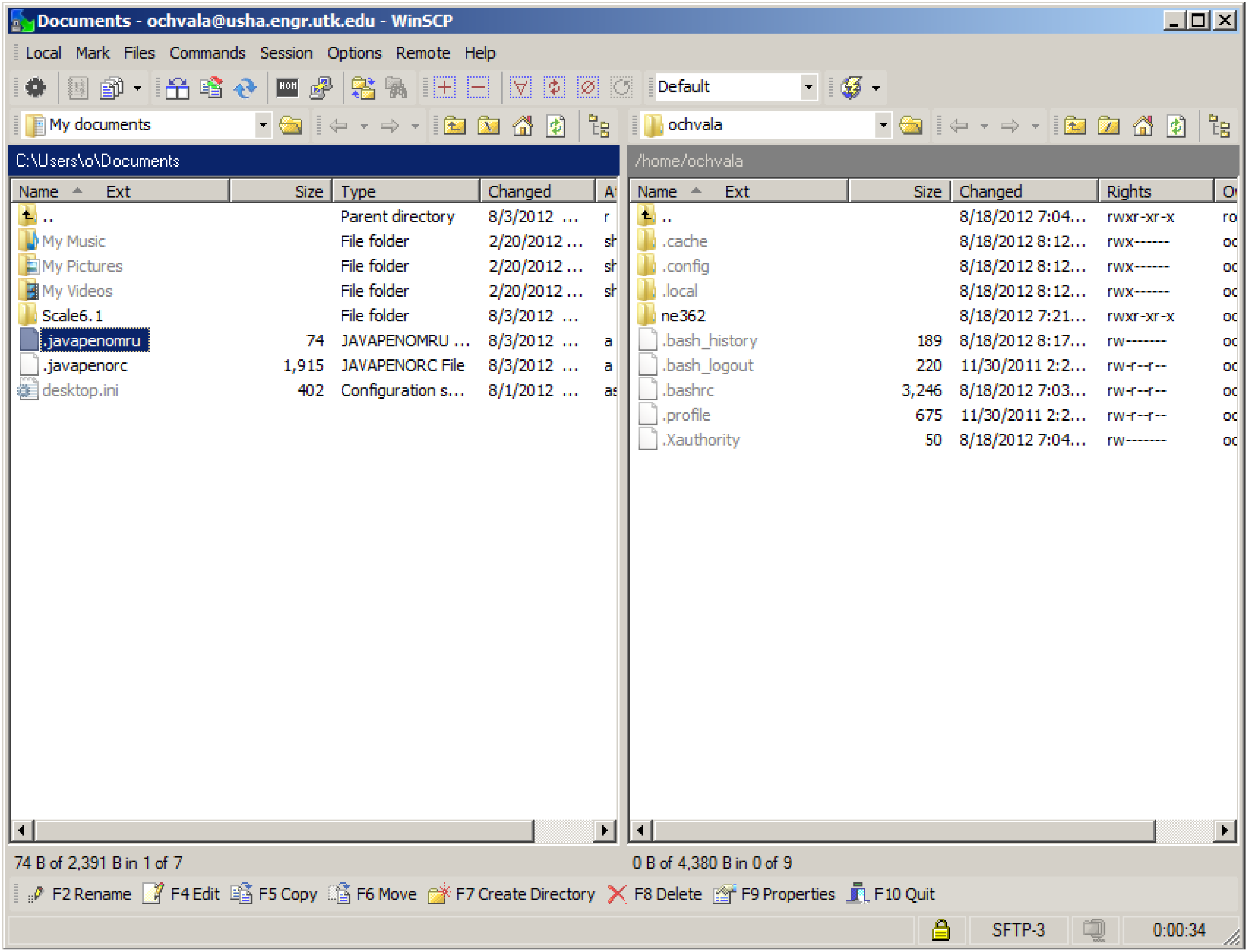


```
login as: ochvala
ochvala@usha.engr.utk.edu's password:
Linux usha 3.2.0-2-686-pae #1 SMP Fri Jun 1 18:56:14 UTC 2012 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Sat Aug 18 22:04:49 2012 from c-71-228-165-190.hsd1.tn.comcast.net
ochvala@usha:~$
```

- **Change your password using *passwd* command!**

Copying files between Usha and Windows

- Use WinSCP http://winscp.net/eng/download.php



C:\Users\o\Documents

Name	Ext	Size	Type	Changed	A
..			Parent directory	8/3/2012 ...	r
My Music			File folder	2/20/2012 ...	sh
My Pictures			File folder	2/20/2012 ...	sh
My Videos			File folder	2/20/2012 ...	sh
Scale6.1			File folder	8/3/2012 ...	
.javapenomru		74	JAVAPENOMRU ...	8/3/2012 ...	a
.javapenorc		1,915	JAVAPENORC File	8/3/2012 ...	a
desktop.ini		402	Configuration s...	8/1/2012 ...	as

74 B of 2,391 B in 1 of 7

/home/ochvala

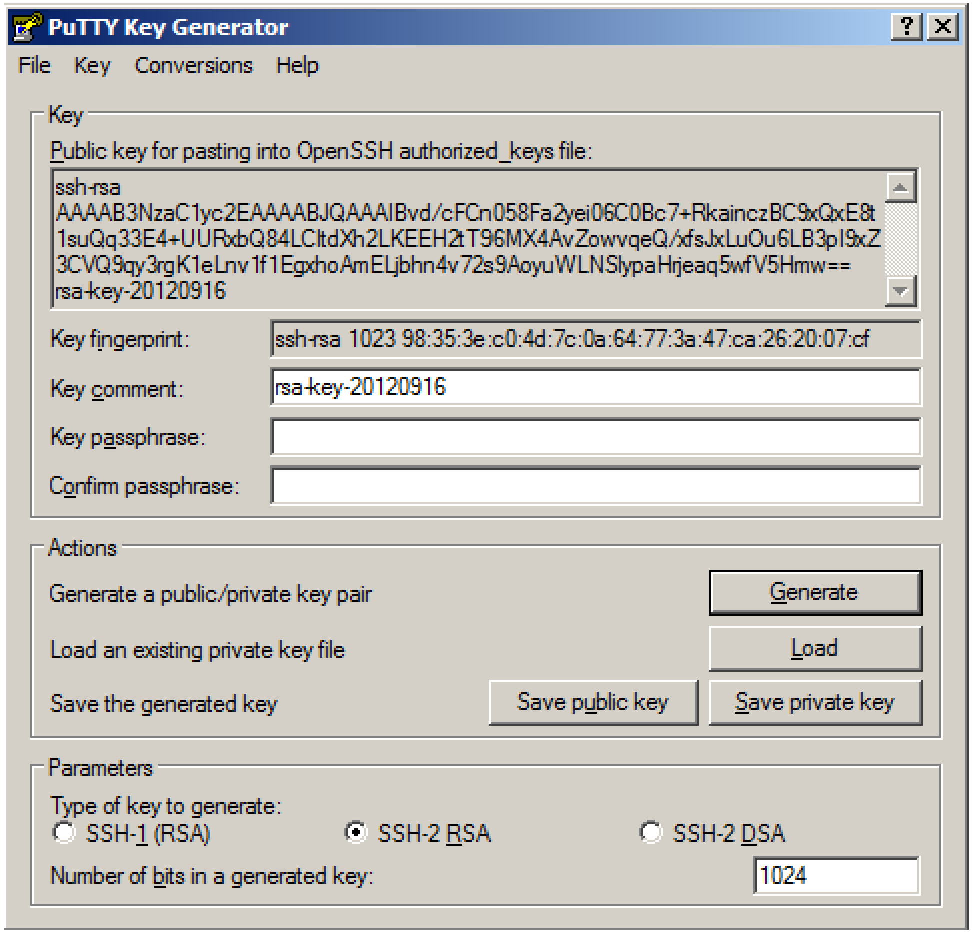
Name	Ext	Size	Changed	Rights	O
..			8/18/2012 7:04...	rwxr-xr-x	ro
.cache			8/18/2012 8:12...	rwx------	oc
.config			8/18/2012 8:12...	rwx------	oc
.local			8/18/2012 8:12...	rwx------	oc
ne362			8/18/2012 7:21...	rwxr-xr-x	oc
.bash_history		189	8/18/2012 8:17...	rw-------	oc
.bash_logout		220	11/30/2011 2:2...	rw-r--r--	oc
.bashrc		3,246	8/18/2012 7:03...	rw-r--r--	oc
.profile		675	11/30/2011 2:2...	rw-r--r--	oc
.Xauthority		50	8/18/2012 7:04...	rw-------	oc

0 B of 4,380 B in 0 of 9

- Note: Filezilla is another alternative, works also on Mac and Linux http://filezilla-project.org/download.php

Generating ssh keys in Windows: PuTTYgen

- Windows see Configuring PuTTY to use Identities i.e. keys http://www.mtu.net/~engstrom/ssh-agent.php#PuTTY



Key

Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAABJQAAAIBvd/cFCn058Fa2yei06C0Bc7+RkainczBC9xQxE8t
1suQq33E4+UURxbQ84LCltdXh2LKEEH2tT96MX4AvZowvqeQ/xfsJxLuOu6LB3pI9xZ
3CVQ9qy3rgK1eLnv1f1EgxhoAmELjbhn4v72s9AoyuWLNSlypaHrjeaq5wfV5Hmw==
rsa-key-20120916
```

Key fingerprint: ssh-rsa 1023 98:35:3e:c0:4d:7c:0a:64:77:3a:47:ca:26:20:07:cf

Key comment: rsa-key-20120916

Key passphrase:

Confirm passphrase:

Actions

Generate a public/private key pair
Generate
Load an existing private key file
Load
Save the generated key
Save public key
Save private key

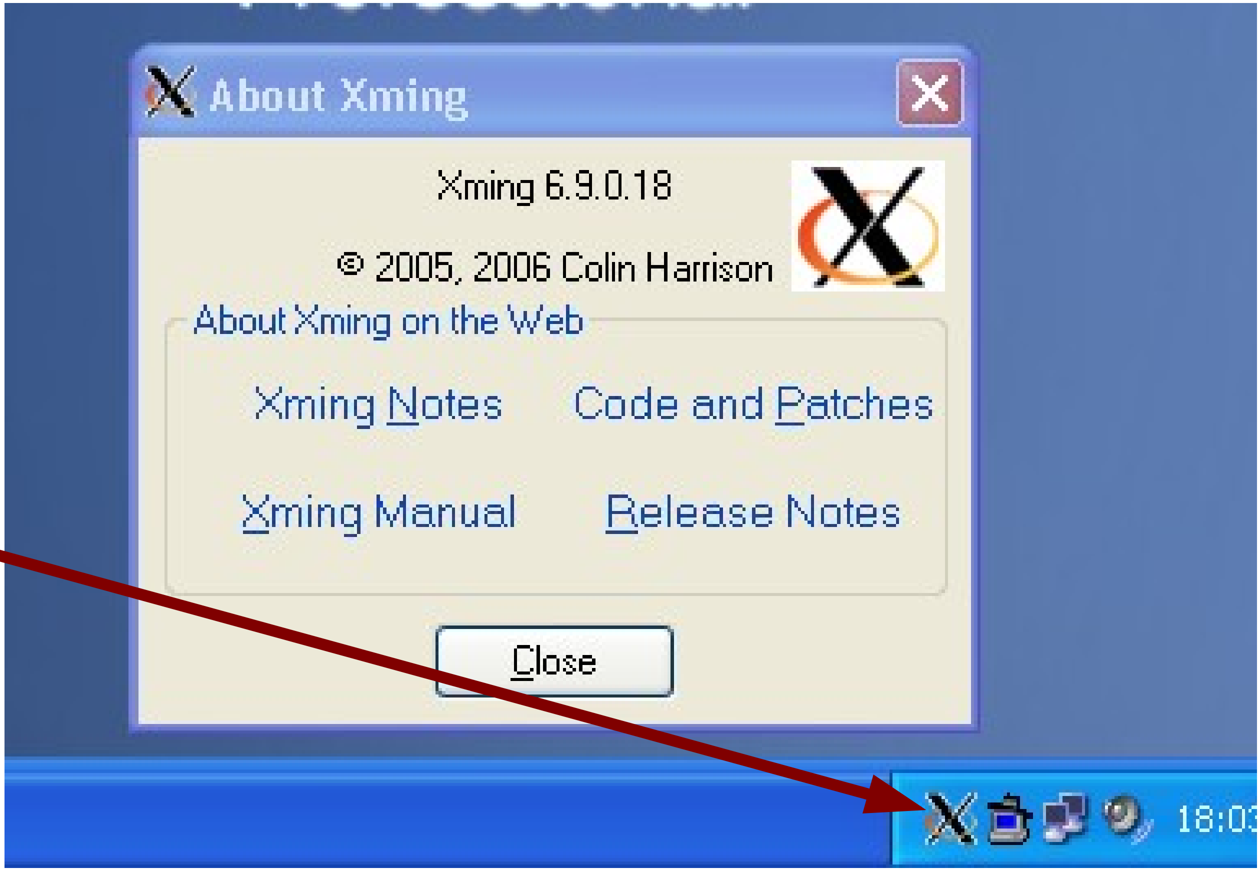
Parameters

Type of key to generate:
○ SSH-1 (RSA) ● SSH-2 RSA ○ SSH-2 DSA

Number of bits in a generated key: 1024

X11 in Windows

- Linux and Mac come with X11 server implementation.
- There are several Xsevers for Windows. A nice freeware is Xming.
- First install the package Xming (by clicking on this link) and then install the package Xming-fonts.



About Xming

Xming 6.9.0.18

© 2005, 2006 Colin Harrison

- When started, you should see an icon in the dock, click it to get info window.

Navigating Linux environment

- List files in a directory: *ls -lah*
- Copy file: *cp <from> <to>*; move: *mv <from> <to>*
- Remove file: *rm <file>*; Remove directory: *rmdir <file>*
- Editors: *vi*, *nano*, *emacs*, *geany*, *kate*, ...
- Need help? Use *man <command>,* Google is your friend.
- See “resources” links at http://usha.engr.utk.edu/welcome.html and remember that Google is your good friend indeed!
- **Midnight Commander** (command *mc*) is a useful tool to navigate around a Linux computer, similar to Norton/Far/Volkov Commanders.
  - View/change directory, view/edit/copy/move files, ...

More ssh related resources

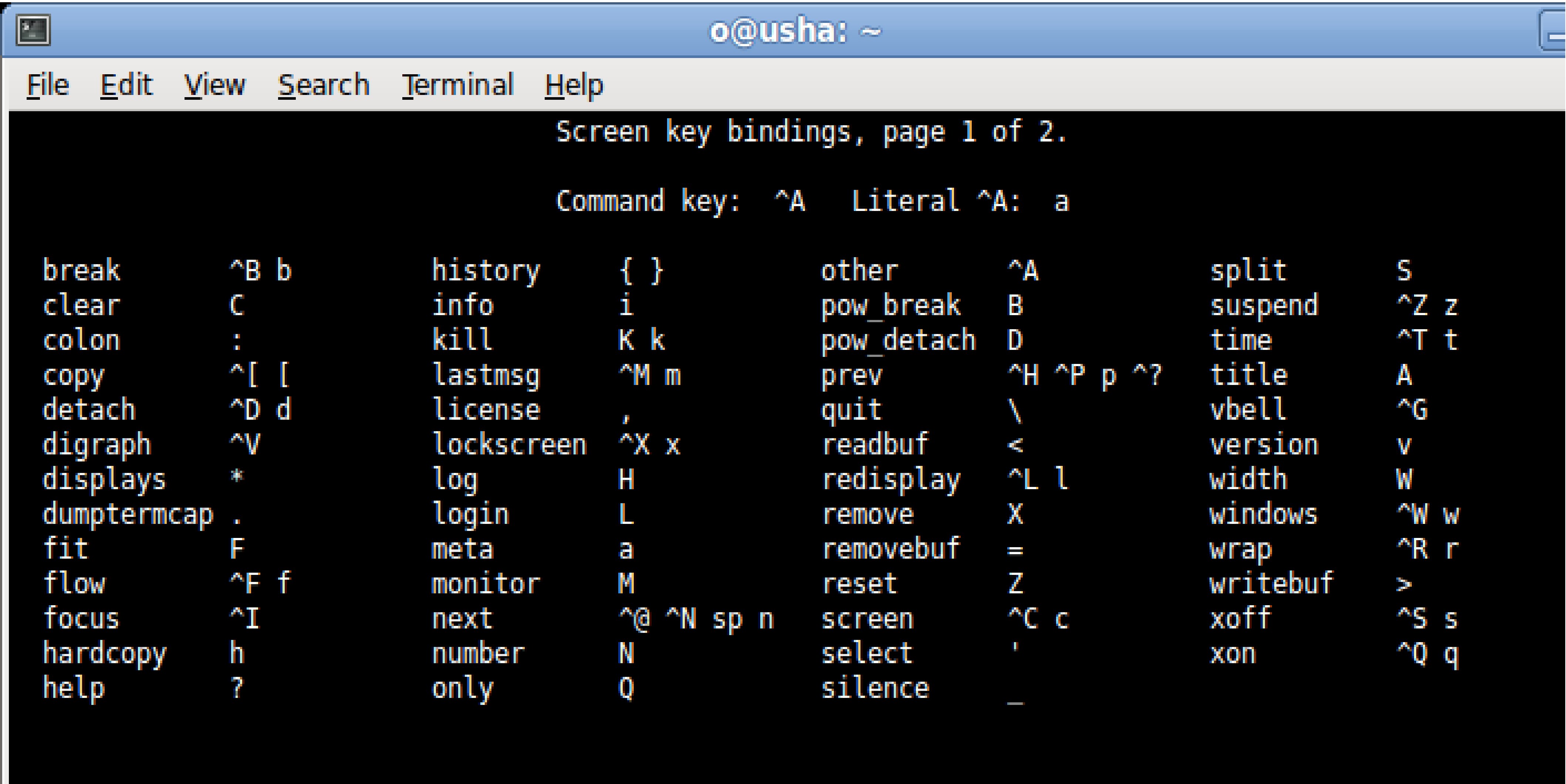
- Practical Cryptography SSH: youtube talk
- An Illustrated Guide to SSH Agent Forwarding
- SSH with Keys HOWTO
- SSH Port Forwarding - UbuntuDoc
- Short series on ssh port forwarding
- SSH Dynamic Port Forwarding (SOCKS proxy)
- SSH one-liners from http://www.commandlinefu.com
- Windows: Configuring PuTTY to use Identities i.e. keys
- Windows: SSH Tunneling: Using Putty to Bypass Web Filters
- Windows: Another article about PuTTY tunneling, with useful links

Summary for ssh and cryptography

- Internet is fundamentally plain-text based, and you need to worry about security. Hardware gets stolen, passwords get sniffed, http connections hijacked → personal identities get stolen.
- Computers are fast, strong encryption is available. Therefore, encrypt everything: communication channels (https, imaps, smtps, etc.), storage media, backups, disk drives, USB keys, phone storage, …
- Set strong passwords, use keys for authentication, set convenient aliases for your connections in ~/.ssh/config
- SSH is much more than just secure shell:
  - Remote execution, file transfer, X11 forwarding, mounting filesystems via ssh, local/remote port forwarding, SOCKS proxy, and more. Practice and investigate on your own.
- **Make sure you follow applicable cybersecurity rules!**

screen

- Screen is a full-screen window manager that multiplexes a physical terminal between several processes (typically interactive shells).
- Type *screen* in terminal to start. “ctrl+A ?” for help. Also *man screen*.



```
                                   Screen key bindings, page 1 of 2.

                                   Command key:  ^A   Literal ^A:  a

  break       ^B b         history     { }          other       ^A           split       S
  clear       C            info        i            pow_break   B            suspend     ^Z z
  colon       :            kill        K k          pow_detach  D            time        ^T t
  copy        ^[ [         lastmsg     ^M m         prev        ^H ^P p ^?   title       A
  detach      ^D d         license     ,            quit        \            vbell       ^G
  digraph     ^V           lockscreen  ^X x         readbuf     <            version     v
  displays    *            log         H            redisplay   ^L l         width       W
  dumptermcap .            login       L            remove      X            windows     ^W w
  fit         F            meta        a            removebuf   =            wrap        ^R r
  flow        ^F f         monitor     M            reset       Z            writebuf    >
  focus       ^I           next        ^@ ^N sp n   screen      ^C c         xoff        ^S s
  hardcopy    h            number      N            select      '            xon         ^Q q
  help        ?            only        Q            silence     _
```

Selected screen control commands

- “ctrl+A <something>” is used to control screen
- Useful commands:
  - “ctrl+A c” **c**reate and open a new shell window.
  - “ctrl+A n” switch to **n**ext window
  - “ctrl+A p” switch to **p**revious window
  - “ctrl+A <N>” switch to window # **N**
  - “ctrl+A C” **C**lear screen
  - “ctrl+A h” save current window's **h**ardcopy into hardcopy.<N> file
  - “ctrl+A H” Begins/ends logging into screenlog.<N> file
  - “ctrl+A d” **d**etach screen. Reattach with *screen -rd*

job control and nohup

- **nohup**: When you want to let a process running even after you logout.
  - Usage: *nohup <what-to-run> &*
  - Writes output to a file *nohup.out*
  - To save output to *FILE*, use *nohup COMMAND > FILE*
  - To redirect standard error: *nohup COMMAND 2> ERRFILE*
- Ampersand "*&*" after a command will detach the command from the standard input, and the job will run in background.
- Use *jobs* to see how many detached jobs are running.
- To detach a running job, use suspend "ctrl+z", then *bg* command.
- To reconnect a job, use *fg* command.
- *bg* and *fg* accept argument <job number>. Use *jobs* command to see which job corresponds to which number.

disown

- **disown**: When you want to let already executed process running even after you logout.
  - Usage: *disown [-ar] [-h] [job_number …]*
  - Without options, each *job_number* is removed from the table of active jobs.
  - If the -*h* option is given, each *job_number* is not removed from the table, but is marked so that it is not terminated if shell terminates.
  - If no *job_number* is present, and neither the -*a* nor the -*r* option is supplied, the current job is used.
  - If no *job_number* is supplied, the -*a* option means to remove or mark all jobs;
  - -*r* option without a *job_number* restricts operation to running jobs.

Summary for job control and screen

- Commands can run on foreground (stdin connected to terminal), or on background (stdin disconnected).
- “ctrl+z”, *jobs*, *bg*, *fg*
- *disown* to prevent jobs from killing on shell termination (i.e. logout).
- *nohup* to start a job such that it runs disowned.
- *screen* to keep terminal shells running even when you disconnect.