Lecture 10

Secure Shell.

Department of Computer Systems FIT, Czech Technical University in Prague

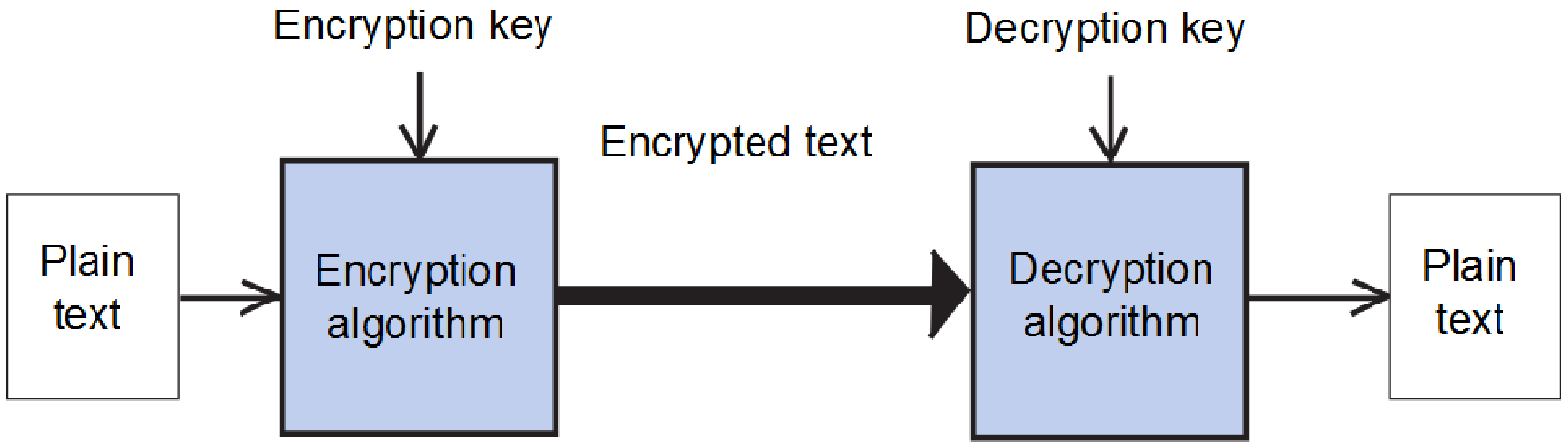
Secure shell – SSH

- **Software solution of network security** (on application level).

- Base on **client/server architectute**
  - **client**: **`ssh`**, **`slogin`**, **`scp`** (Unix)
    e.g. **`putty`**, **`winscp`** (MS Windows)
  - **server**: **`sshd`**
  - Transport protocol is TCP/IP and server usually listen to port 22
  - several implementation SSH1, SSH2, OpenSSH, ...

- **Properties:**
  - **Privacy (encryption)** = protecting data from disclosure
  - **Integrity** = data are not modified during the transfer
  - **Authentication** = verifying someone's identity (server/client)
  - **Authorization** = deciding what someone may or may not do
  - **Forwarding (Tunneling)**= encapsulating another TCP-based service

- **Encryption**
  - the process of scrambling data so that it can't be read by unauthorized parties



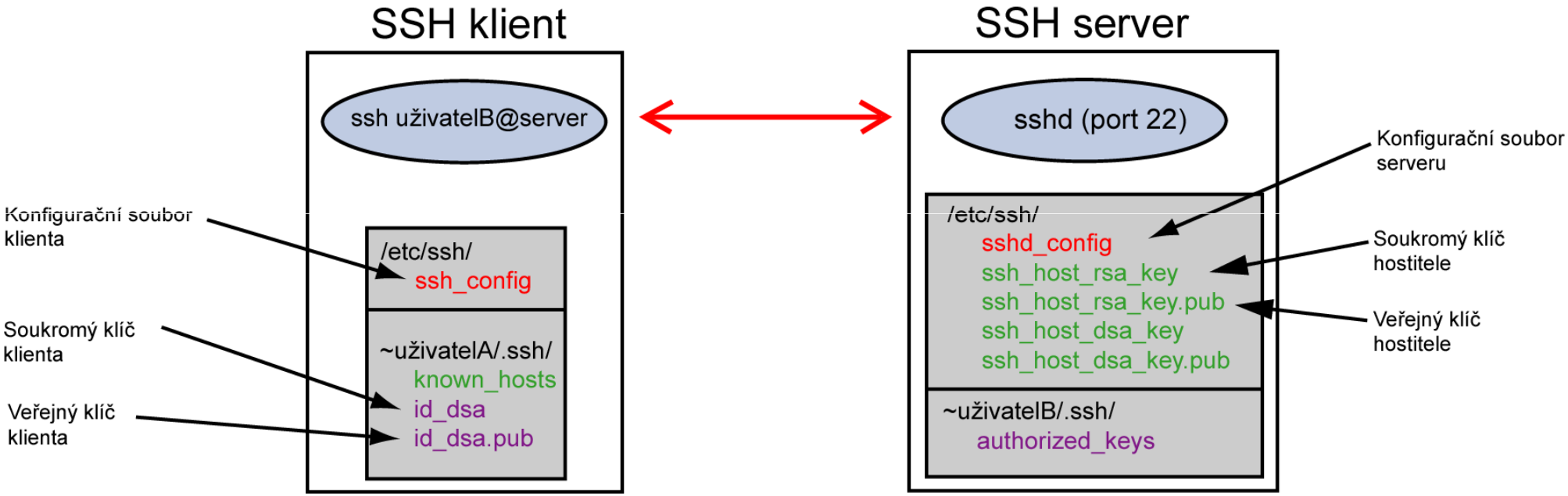
- **Encryption algorithm (cipher)**
  - method of performing the scrambling (e.g. DES, RSA, DSA,…)
  - it isn't currently possible to prove a practical cipher secure
  - attempt to decrypt data without its key

- **Cryptanalysis**
  - attempt to decrypt data without its key

- **Symmetric ciphers**
  - use **the same secret key** for encrypting and decrypting
  - **advantage:** fast algorithms
  - **drawback:** problem of key distribution
  - e.g. Blowfish, DES, IDEA, RC4

- **Asymmetric ciphers**
  - use a pair of keys: **public and private keys**
  - data encrypted with the public key may be decrypted only with its private counterpart
  - **it is infeasible to derive the private key from the public one**
  - **advantage:** no problem with key distribution
  - **drawback:** slow algorithms
  - e.g. RSA, DSA, ElGamal, Elliptic Curve, ...

Secure connection



1. **Client contact serve** (TCP port 22 by default)
2. Client and server disclose the SSH protocol **versions they support**.
3. **The server identifies itself** to the client and provides session parameters.
4. The **client sends the server a secret (session) key**.
5. **Both sides turn on encryption and complete** server authentication.

Examples

- **Connection to unknown server (first connection)**

```
$ ssh trdlicka@dray1.feld.cvut.cz
The authenticity of host 'dray1.feld.cvut.cz (147.32.192.154)' can't be established
RSA key fingerprint is d8:d4:05:fe:a7:b5:a1:42:6b:79:d4:58:3e:fe:44:1f.
Are you sure you want to continue connecting (yes/no)?  yes
```

- **Fingerprint of key**
  - enable to compare to keys

- **How to get fingerprint?**

```
$ ssh-keygen -l -f ssh_host_rsa_key.pub
1024 d8:d4:05:fe:a7:b5:a1:42:6b:79:d4:58:3e:fe:44:1f ssh_host_rsa_key.pub
```

Client authentication

- **Password**

- **Client public key**
  1. client sends its public key to the server

     (e.g. ~userA/.ssh/id_dsa.pub)
  2. server tries to find client public key record

     (např ~userB/.ssh/authorized_keys)
  3. server creates random text, encrypts it by client public key and sends it to client
  4. client decrypts text by its private key and sends back to server

Examples

- **Key generation**

  `$ ssh-keygen -t dsa`

- Keys are saved at

  `~userA/.ssh/id_dsa` (private key)

  `~userA/.ssh/id_dsa.pub` (public key)

- **Adding client public key in server**
  `~userB/.ssh/authorized_keys`

- **Authorization by client public key**

  userA@client$ `ssh userB@server`

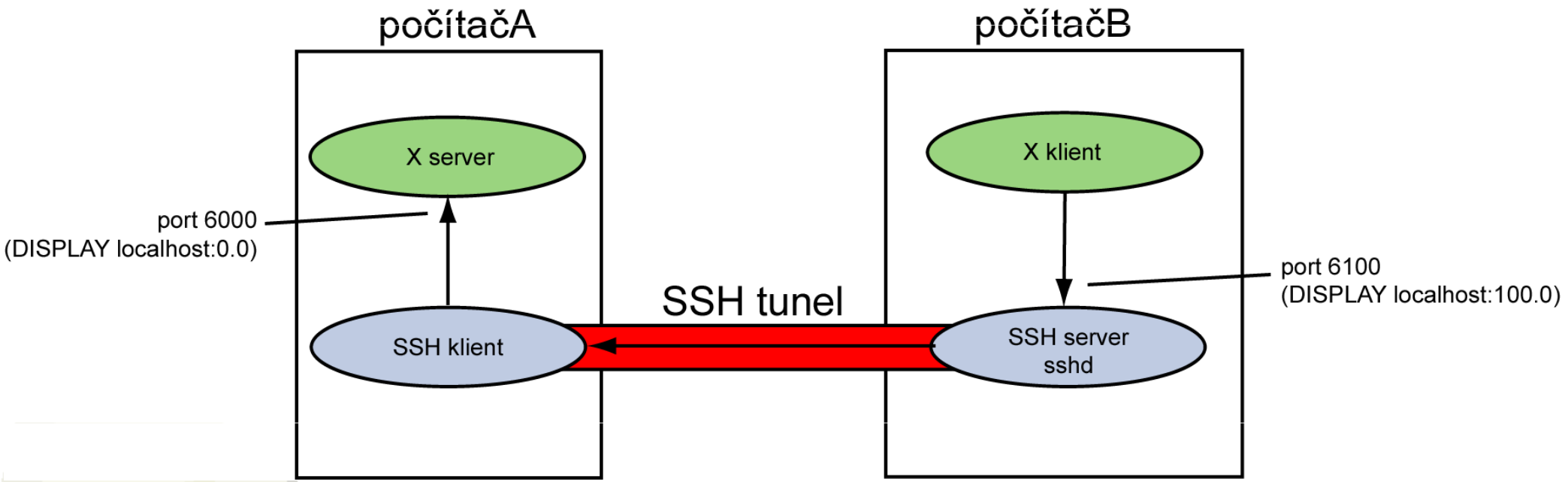
  userB@server$

Examples

- **How to find problem?**

```
ssh –v user@server
```

X11 port forwarding

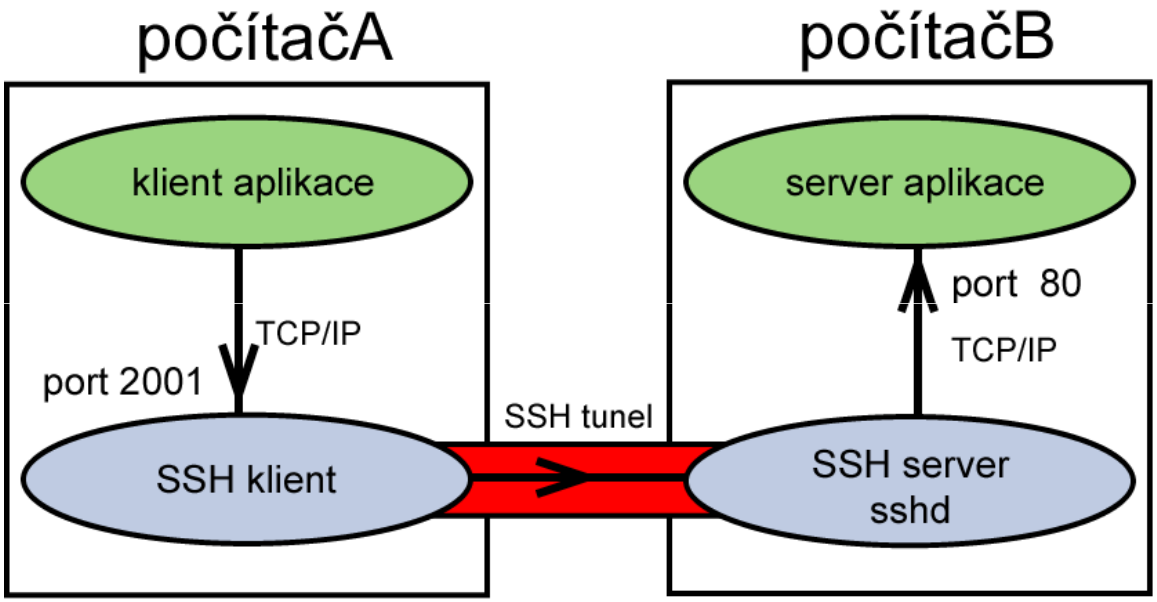


Example

- **Connection from počítačA to počítačB using X11 port forwarding**

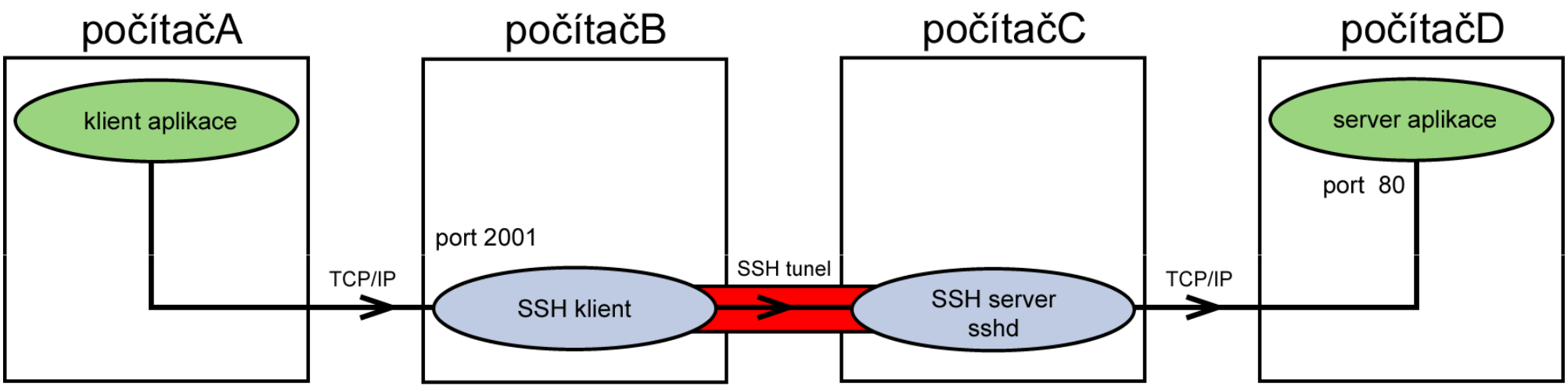
```
$ ssh -X user@počítačB
```

Port forwarding (local) I



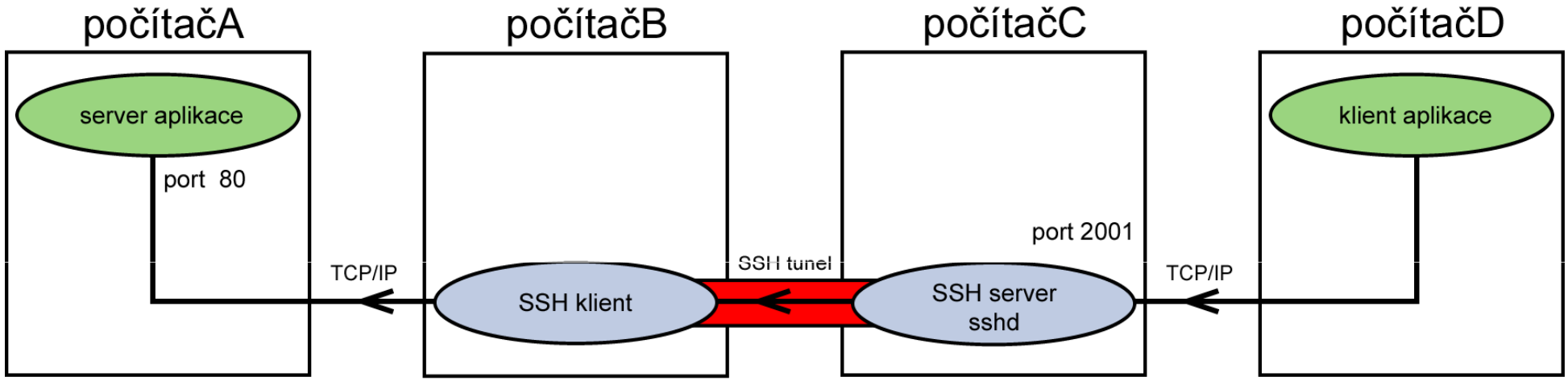

```
ssh -L 2001:localhost:80 uživatel@počítačB
```

Port forwarding (local) II




```
ssh -g -L 2001:počítačD:80 uživatel@počítačC
```

Port forwarding (remote)




```
ssh -g -R 2001:počítačD:80 uživatel@počítačC
```