Secure SHell (SSH)

I wrote the initial version of SSH (Secure Shell) in Spring 1995. It was a time when telnet and FTP were widely used.

Anyway, I designed SSH to replace both `telnet` (port 23) and `ftp` (port 21). Port 22 was free. It was conveniently between the ports for `telnet` and `ftp`. I figured having that port number might be one of those small things that would give some aura of credibility. But how could I get that port number? I had never allocated one, but I knew somebody who had allocated a port.

The basic process for port allocation was fairly simple at that time. Internet was smaller and we were in the very early stages of the Internet boom. Port numbers were allocated by IANA (Internet Assigned Numbers Authority). At the time, that meant an esteemed Internet pioneer called Jon Postel and Joyce K. Reynolds. Among other things, Jon had been the editor of such minor protocol standards as IP (RFC 791), ICMP (RFC 792), and TCP (RFC 793). Some of you may have heard of them.

To me Jon felt outright scary, having authored all the main Internet RFCs!

Anyway, just before announcing `ssh-1.0` in July 1995, I sent this e-mail to IANA:

```
From ylo Mon Jul 10 11:45:48 +0300 1995 From: Tatu Ylonen <ylo@cs.hut.fi>
To: Internet Assigned Numbers Authority <iana@isi.edu>
Subject: request for port number
Organization: Helsinki University of Technology, Finland
Dear Sir, I have written a program to securely log from one machine into another over an insecure network.
It provides major improvements in security and functionality over existing telnet and rlogin protocols
and implementations. In particular, it prevents IP, DNS and routing spoofing.  My plan is to distribute
the software freely on the Internet and to get it into as wide use as possible. I would like to get a registered
privileged port number for the software.

The number should preferably be in the range 1-255 so that it can be used in the WKS field in name servers.
I'll enclose the draft RFC for the protocol below. The software has been in local use for several months,
and is ready for publication except for the port number. If the port number assignment can be arranged in time,
I'd like to publish the software already this week. I am currently using port number 22 in the beta test.
It would be great if this number could be used (it is currently shown as Unassigned in the lists).
The service name for the software is "ssh" (for Secure Shell).
Yours sincerely,  Tatu Ylonen <ylo@cs.hut.fi>  ... followed by protocol specification for ssh-1.0
```

The next day, I had an e-mail from Joyce waiting in my mailbox:

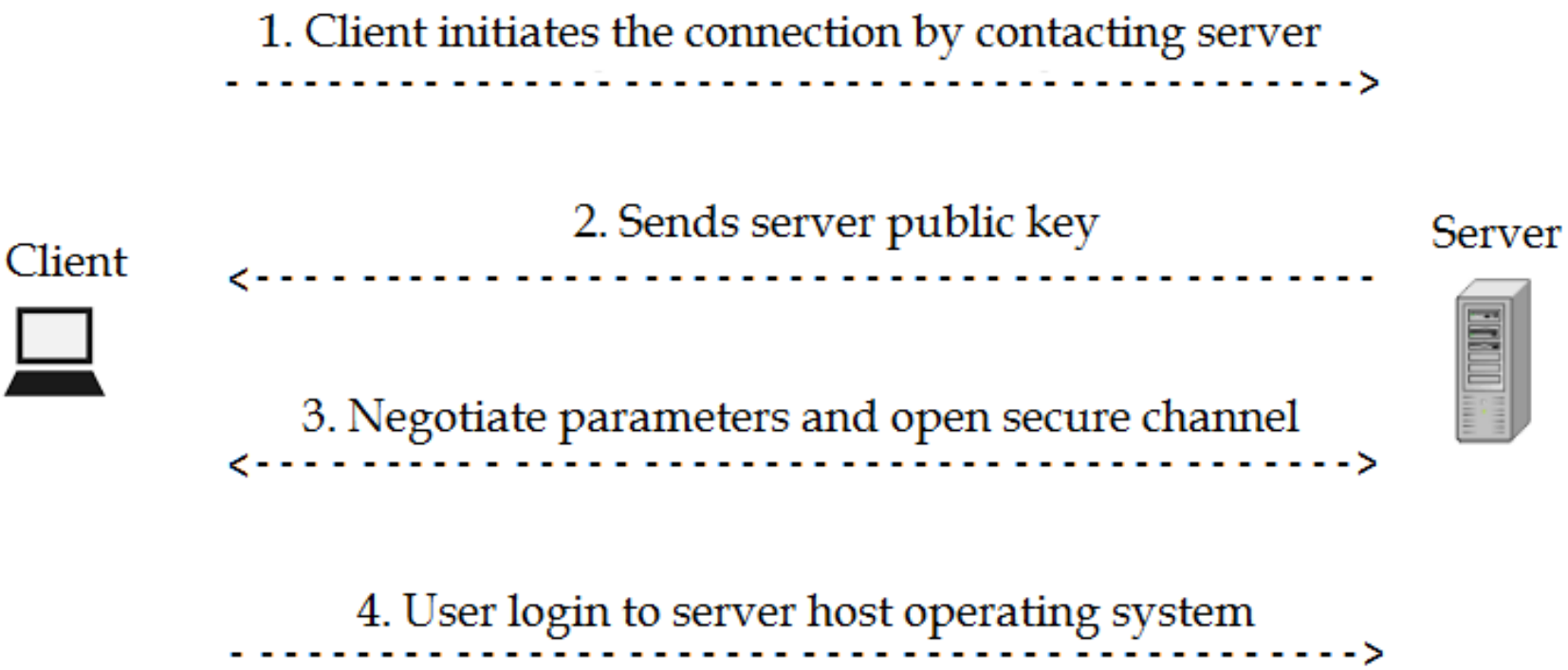
```
Date: Mon, 10 Jul 1995 15:35:33 -0700 From: jkrey@ISI.EDU To: ylo@cs.hut.fi Subject: Re: request for port number
Cc: iana@ISI.EDU  Tatu,  We have assigned port number 22 to ssh, with you as the point of contact.  Joyce
```

There we were! SSH port was 22!!!

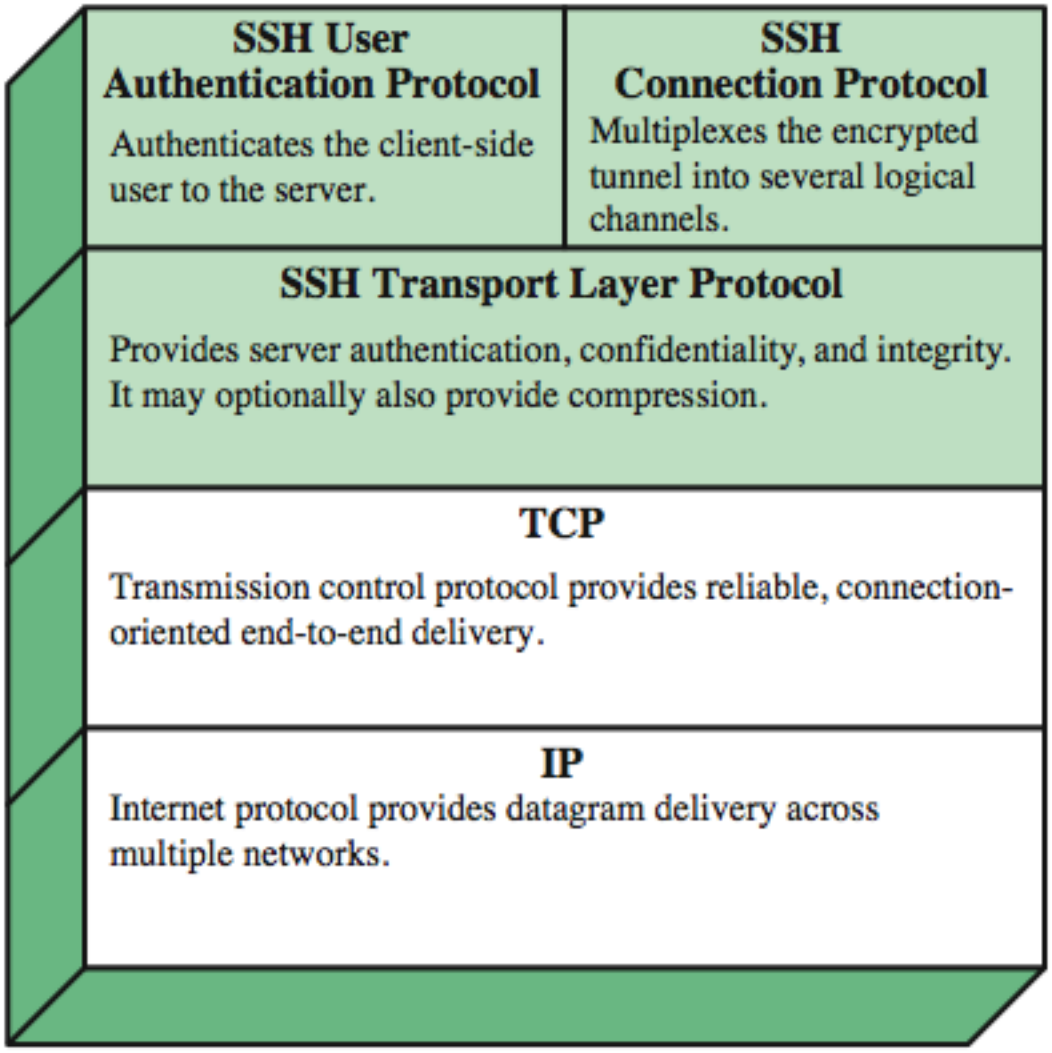
SSH Overview

- SSH = Secure Shell
  - Initially designed to replace insecure rsh, telnet utilities.
  - Secure remote administration (mostly of Unix systems).
  - Latter, provide a general secure channel for network applications.
  - Only covers traffic explicitly protected.
  - Applications need modification, but port-forwarding eases some of this

SSH Overview



SSH Protocol Stack



SSH-2 Architecture

SSH-2 adopts a three layer architecture:

- SSH Transport Layer Protocol.
  - Initial connection.
  - Server authentication
  - Sets up secure channel between client and server via key exchange etc.
- SSH Authentication Protocol
  - Client authentication over secure transport layer channel.
- SSH Connection Protocol
  - Supports multiple connections over a single transport layer protocol secure channel.
  - Efficiency (session re-use).

SSH-2 Security Goals

- Server authenticated in transport layer protocol.
- Client authenticated in authentication protocol.
  - By public key (DSS, RSA).
  - Or simple password
- Establishment of a fresh, shared secret.
  - Shared secret used to derive further keys (Enc Keys, MAC keys, IVs), similar to SSL/TLS.
  - For confidentiality and authenticity in SSH transport layer protocol.
- Secure ciphersuite negotiation.
  - Encryption, MAC, and compression algorithms.

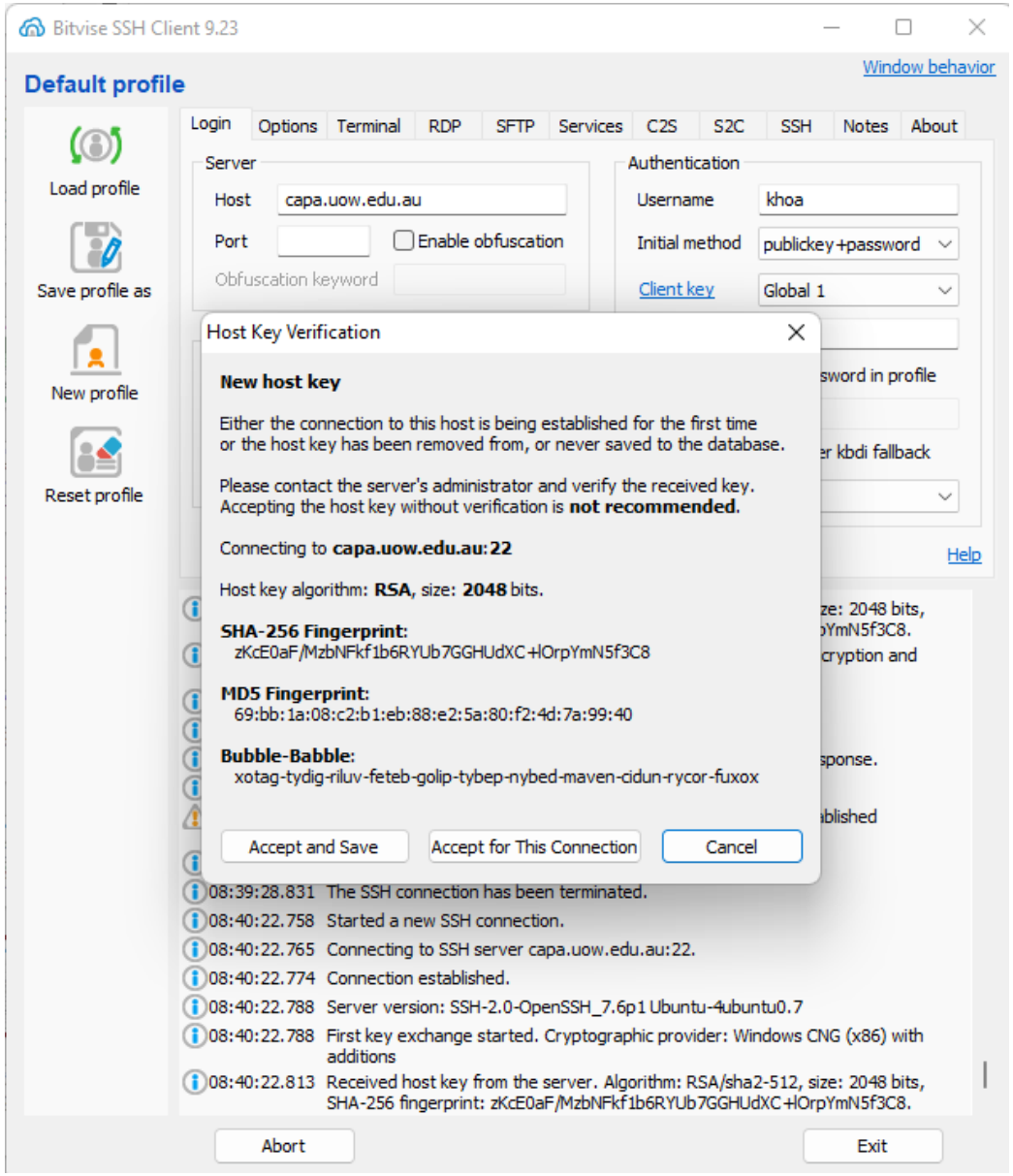
SSH Transport Layer Protocol

- Server authentication, based on server's host key pair(s)
- Packet exchange
  - establish TCP connection
  - can then exchange data (packet exchange)
    - identification string exchange, algorithm negotiation, key exchange, end of key exchange, service request
  - service request: either the User Authentication or the Connection Protocol

SSH Key Fingerprints

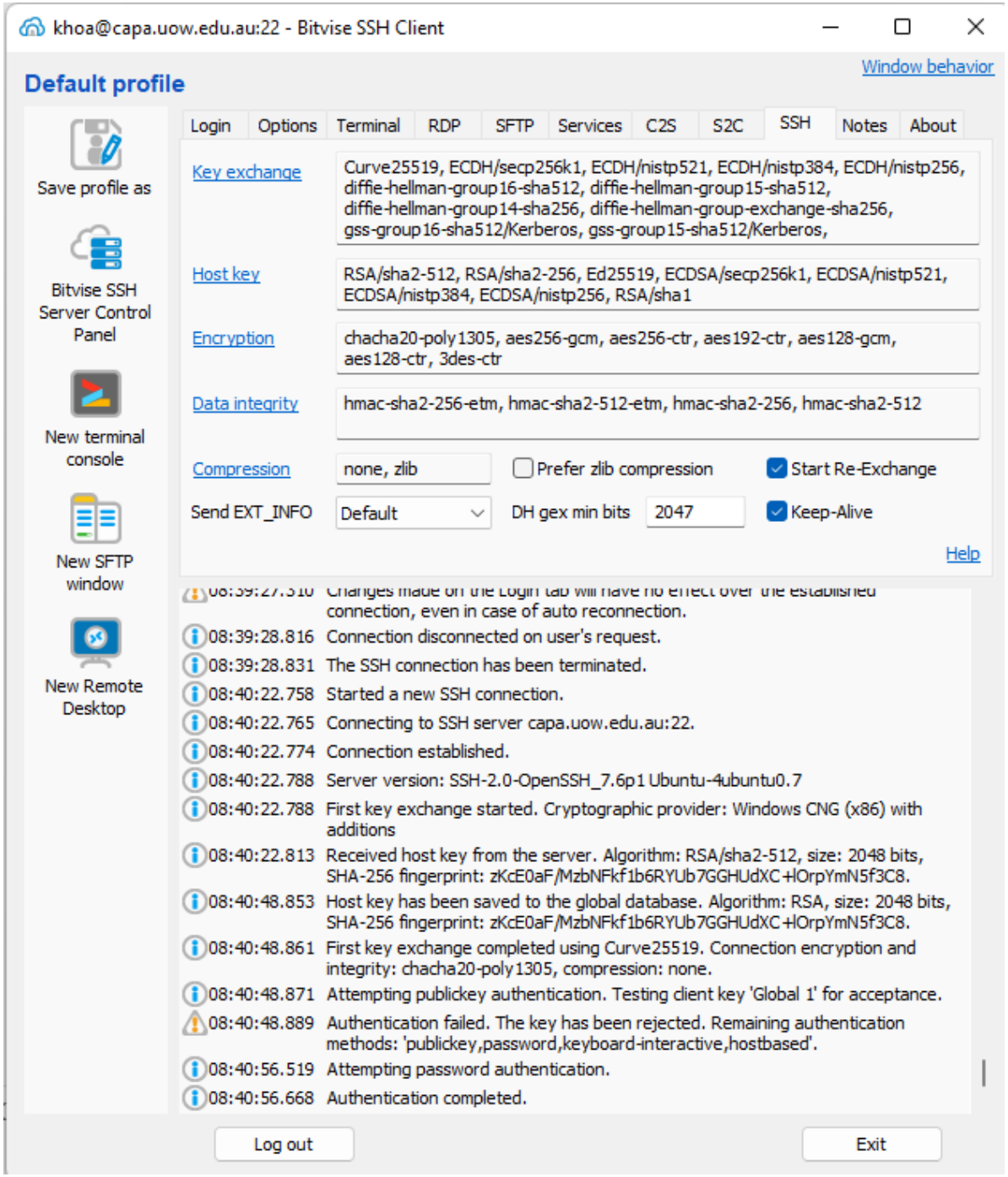
- The security of the connection relies on *the server authenticating itself to the client*.
- When you connect to a remote host computer for the first time, the host sends your local computer its **public key** in order to identify itself. To help you to verify the host's identity, a **fingerprint** of the host's public key is presented to you for verification.
- Many users just blindly accept the presented key.

SSH Key Fingerprints



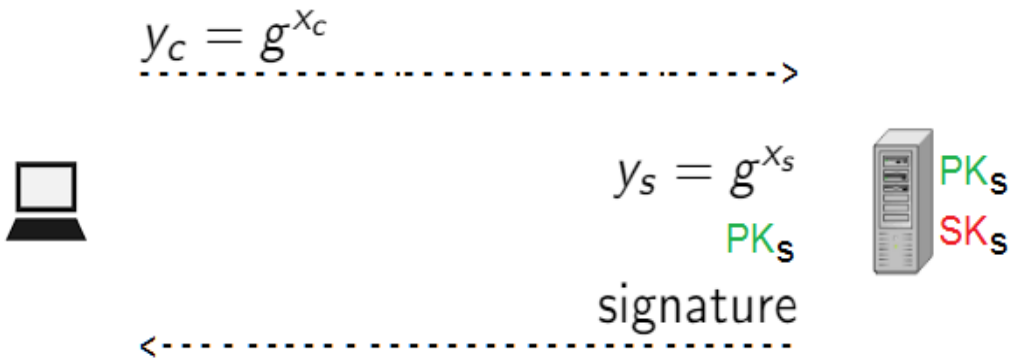
SSH-2 Algorithms

- Key establishment through Diffie-Hellman key exchange.
  - Ephemeral Diffie-Hellman
- Server authentication via RSA or DSS signatures
- HMAC-SHA1 or HMAC-SHA256 for MAC algorithm.
- 3DES, AES, RC4, etc. for Encryption algorithm


khoa@capa.uow.edu.au:22 - Bitvise SSH Client
Default profile
Window behavior
Save profile as
Bitvise SSH Server Control Panel
New terminal console
New SFTP window
New Remote Desktop
Login
Options
Terminal
RDP
SFTP
Services
C2S
S2C
SSH
Notes
About
Key exchange
Curve25519, ECDH/secp256k1, ECDH/nistp521, ECDH/nistp384, ECDH/nistp256, diffie-hellman-group16-sha512, diffie-hellman-group15-sha512, diffie-hellman-group14-sha256, diffie-hellman-group-exchange-sha256, gss-group16-sha512/Kerberos, gss-group15-sha512/Kerberos,
Host key
RSA/sha2-512, RSA/sha2-256, Ed25519, ECDSA/secp256k1, ECDSA/nistp521, ECDSA/nistp384, ECDSA/nistp256, RSA/sha1
Encryption
chacha20-poly1305, aes256-gcm, aes256-ctr, aes192-ctr, aes128-gcm, aes128-ctr, 3des-ctr
Data integrity
hmac-sha2-256-etm, hmac-sha2-512-etm, hmac-sha2-256, hmac-sha2-512
Compression
none, zlib
Prefer zlib compression
Start Re-Exchange
Send EXT_INFO
Default
DH gex min bits
2047
Keep-Alive
Help
connection, even in case of auto reconnection.
08:39:28.816 Connection disconnected on user's request.
08:39:28.831 The SSH connection has been terminated.
08:40:22.758 Started a new SSH connection.
08:40:22.765 Connecting to SSH server capa.uow.edu.au:22.
08:40:22.774 Connection established.
08:40:22.788 Server version: SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.7
08:40:22.788 First key exchange started. Cryptographic provider: Windows CNG (x86) with additions
08:40:22.813 Received host key from the server. Algorithm: RSA/sha2-512, size: 2048 bits, SHA-256 fingerprint: zKcE0aF/MzbNFkf1b6RYUb7GGHUdXC+lOrpYmN5f3C8.
08:40:48.853 Host key has been saved to the global database. Algorithm: RSA, size: 2048 bits, SHA-256 fingerprint: zKcE0aF/MzbNFkf1b6RYUb7GGHUdXC+lOrpYmN5f3C8.
08:40:48.861 First key exchange completed using Curve25519. Connection encryption and integrity: chacha20-poly1305, compression: none.
08:40:48.871 Attempting publickey authentication. Testing client key 'Global 1' for acceptance.
08:40:48.889 Authentication failed. The key has been rejected. Remaining authentication methods: 'publickey,password,keyboard-interactive,hostbased'.
08:40:56.519 Attempting password authentication.
08:40:56.668 Authentication completed.
Log out
Exit

SSH Transport Layer Protocol

Diffie-Hellman Key Exchange



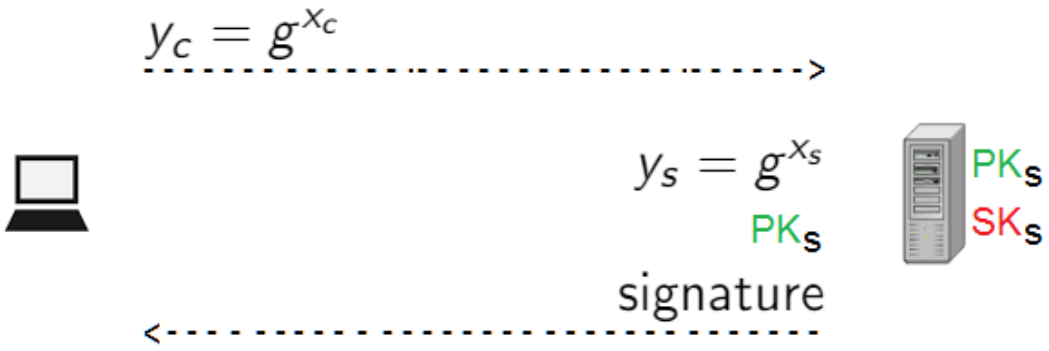
Client generates a random number x_c and computes

$$y_c = g^{x_c} \pmod{p}.$$

Client sends y_c to Server.

SSH Transport Layer Protocol

Diffie-Hellman Key Exchange



Server generates a random number x_S and computes

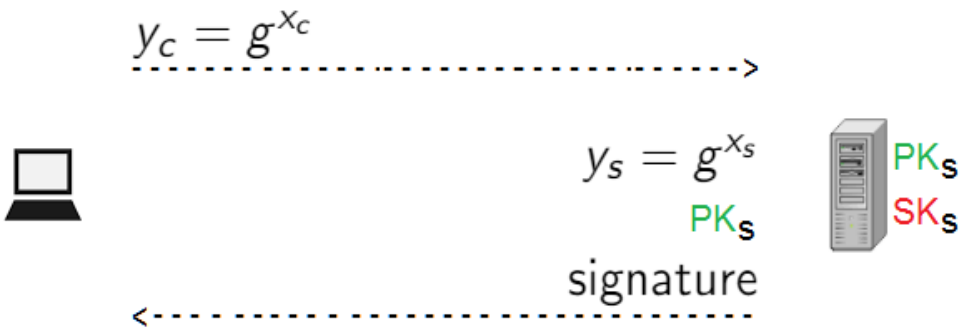
$$y_S = g^{x_S} (\text{mod } p).$$

Server computes the shared secret

$$K = y_c^{x_S} = g^{x_c x_S} \quad (\text{mod } p).$$

SSH Transport Layer Protocol

Diffie-Hellman Key Exchange



Server computes the **exchange hash** value

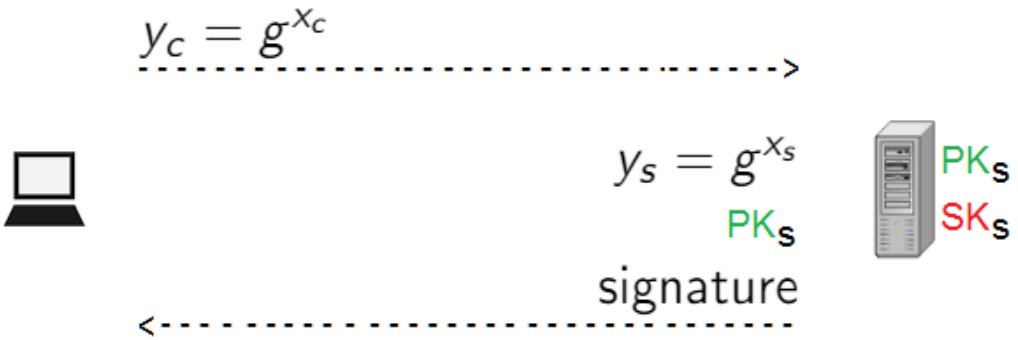
$$H = hash(id_C \,||id_S\,||init_C\,||init_S\,||PK_S\,||y_c\,||y_s\,||K\,)$$

id_S , id_C : Server's and Client's identification strings

$init_S$, $init_C$: Server's and Client's Initial Messages

SSH Transport Layer Protocol

Diffie-Hellman Key Exchange



Server generates the signature on the exchange hash value *signature* = $Sign_{SK_S}(H)$ and sends

(y_s, PK_S, *signature*)

to Client.

SSH Transport Layer Protocol

Key Derivation

- After the key exchange, both Server and Client obtain two common values:
  - a **shared secret** value *K* , and
  - an **exchange hash** value *H*.
- Encryption keys and MAC keys are derived from *K* and *H*.
- The exchange hash value *H* from the first key exchange is additionally used as the session identifier.

SSH Transport Layer Protocol

Key Derivation

Encryption keys must be computed as hash of the shared secret *K* as follows:

- Initial IV client to server: *hash*(*K*||*H*||“A”||*session id*)
- Initial IV server to client: *hash*(*K*||*H*||“B”||*session id*)
- Encryption key client to server: *hash*(*K*||*H*||“C”||*session id*)
- Encryption key server to client: *hash*(*K*||*H*||“D”||*session id*)
- MAC key client to server: *hash*(*K*||*H*||“E”||*session id*)
- MAC key server to client: *hash*(*K*||*H*||“F”||*session id*)

SSH Transport Layer Protocol

Binary Packet Protocol

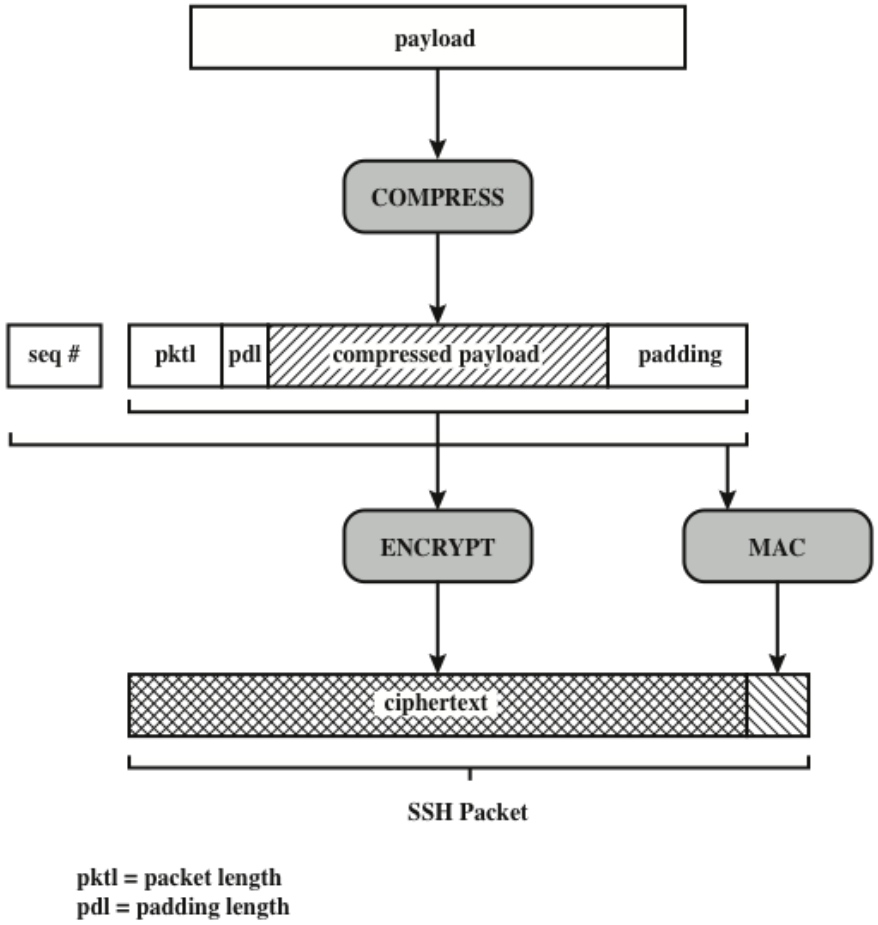


Figure 17.10 SSH Transport Layer Protocol Packet Formation

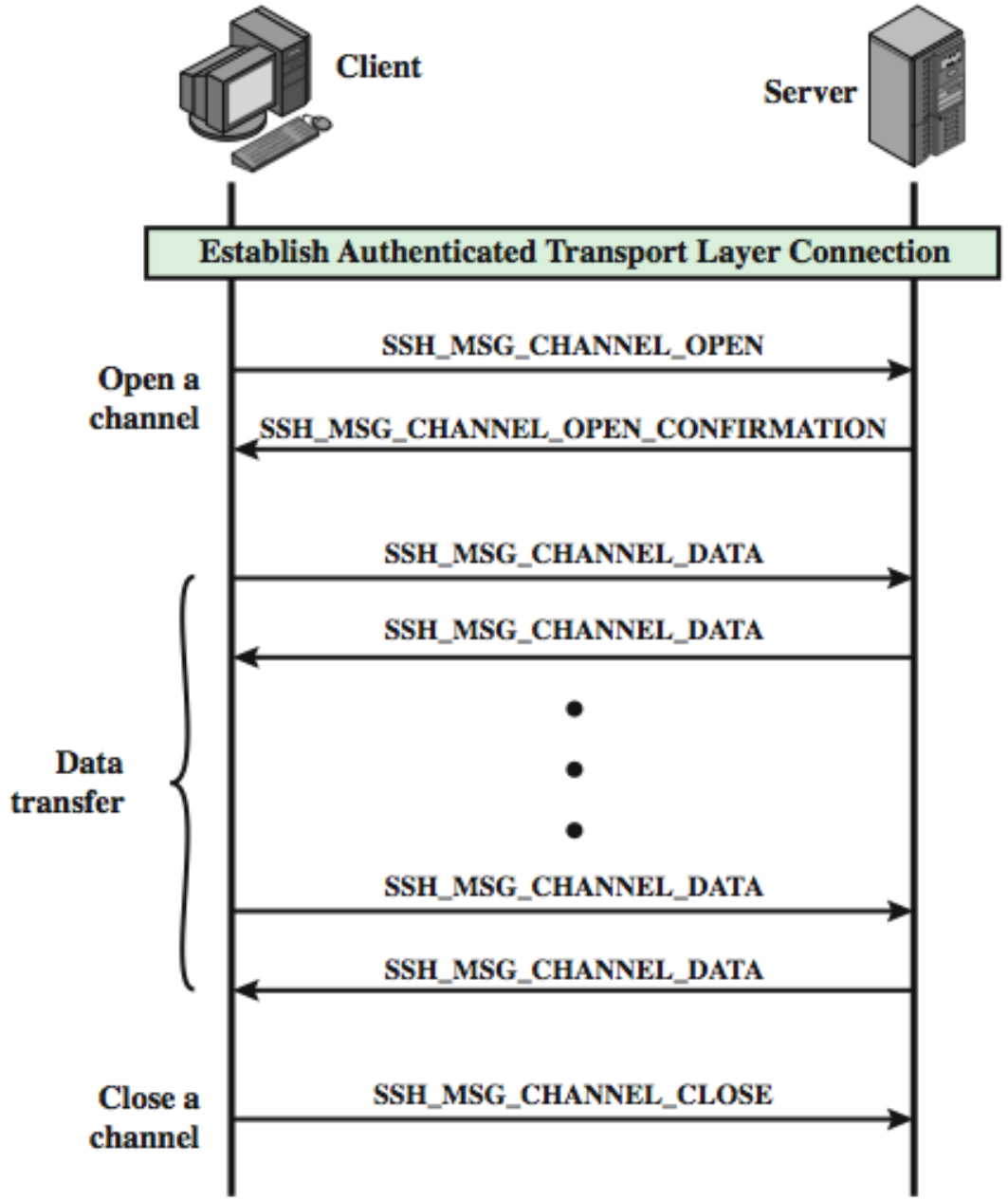
SSH User Authentication Protocol

- Authenticates client to server
- Authentication methods used
  - public key (digital signature)
  - password
  - host-based

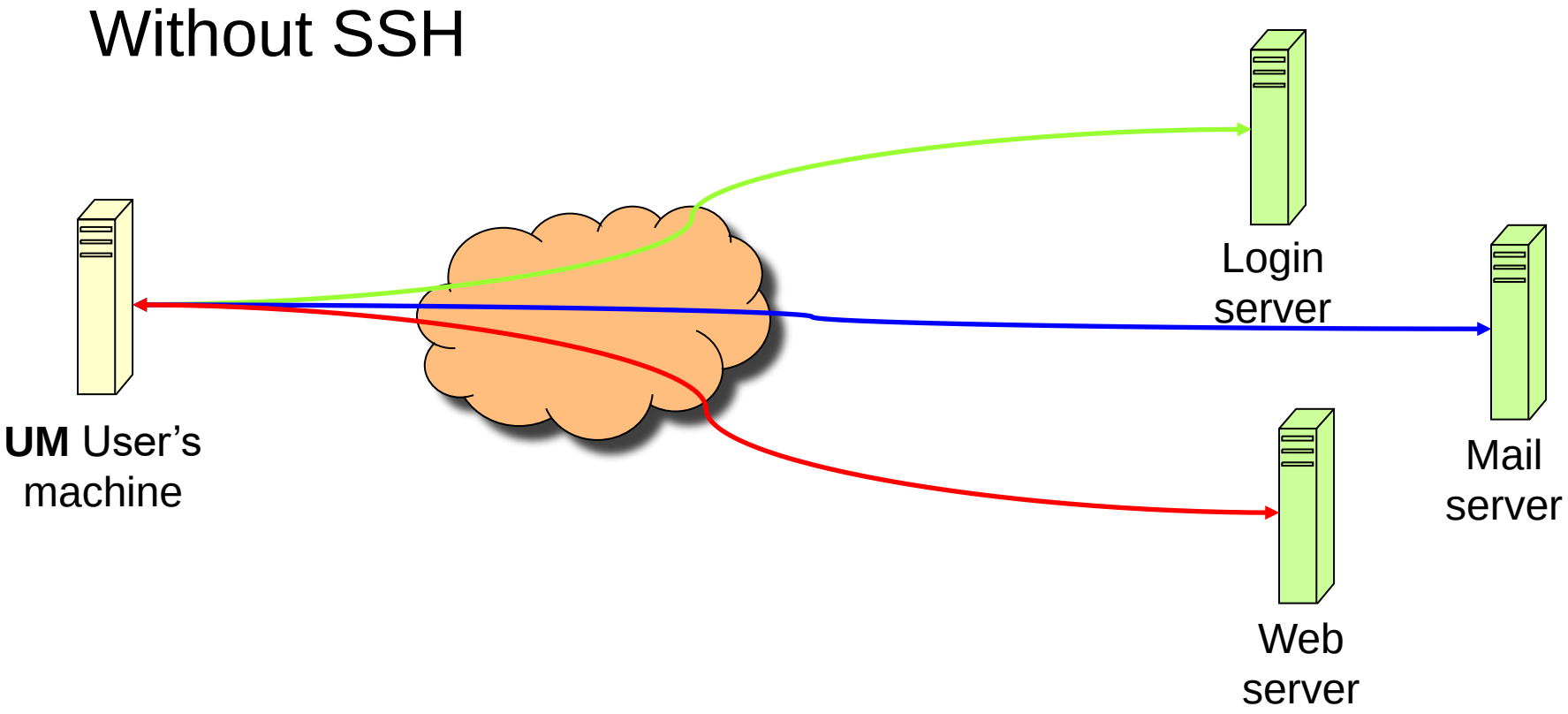
SSH Connection Protocol

- Run on top of the SSH Transport Layer Protocol
- Assume secure authentication connection
- Used for multiple **logical** channels
  - Different SSH communications use separate channels
  - either side can open a channel with unique id number
  - have three stages:
    - opening a channel, data transfer, closing a channel
  - four types:
    - Session: The remote execution of a program.
    - X11: This refers to the X Window System
    - forwarded-tcpip: This is remote port forwarding
    - direct-tcpip: This is local port forwarding

SSH Connection Protocol Exchange

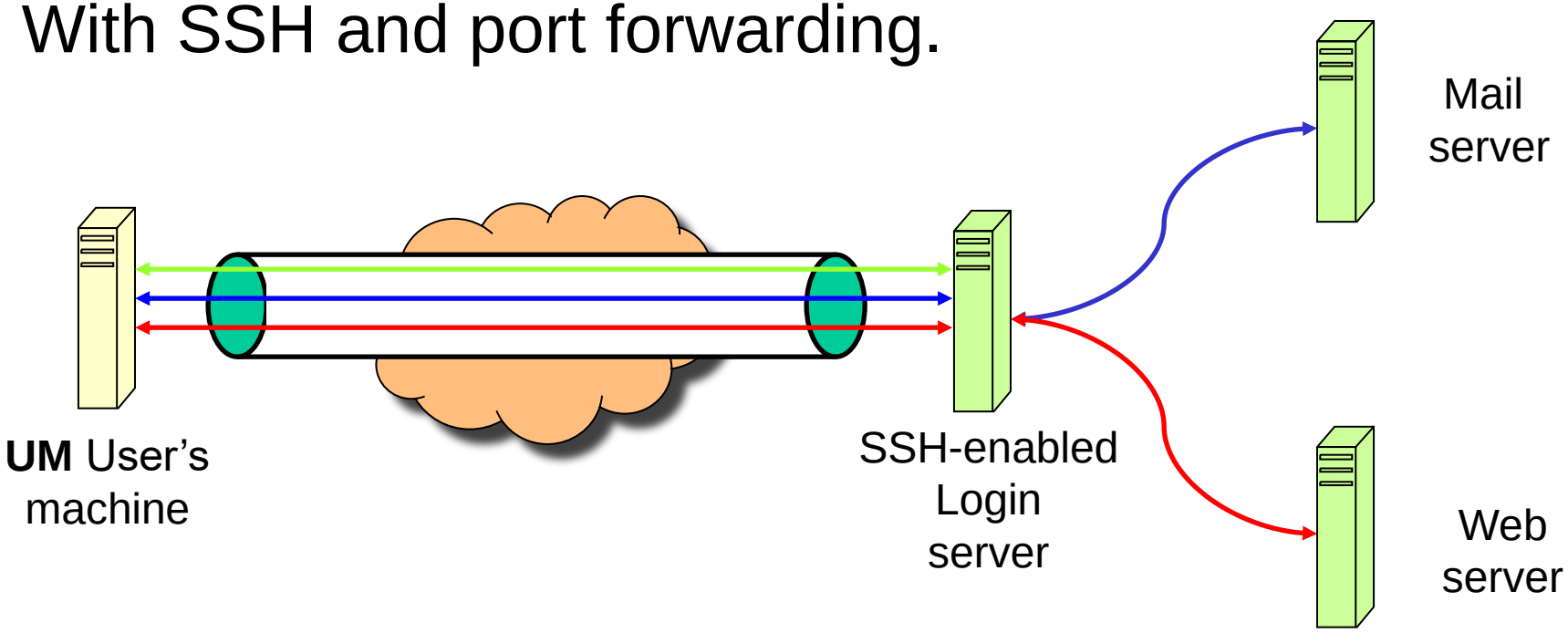


SSH Port Forwarding

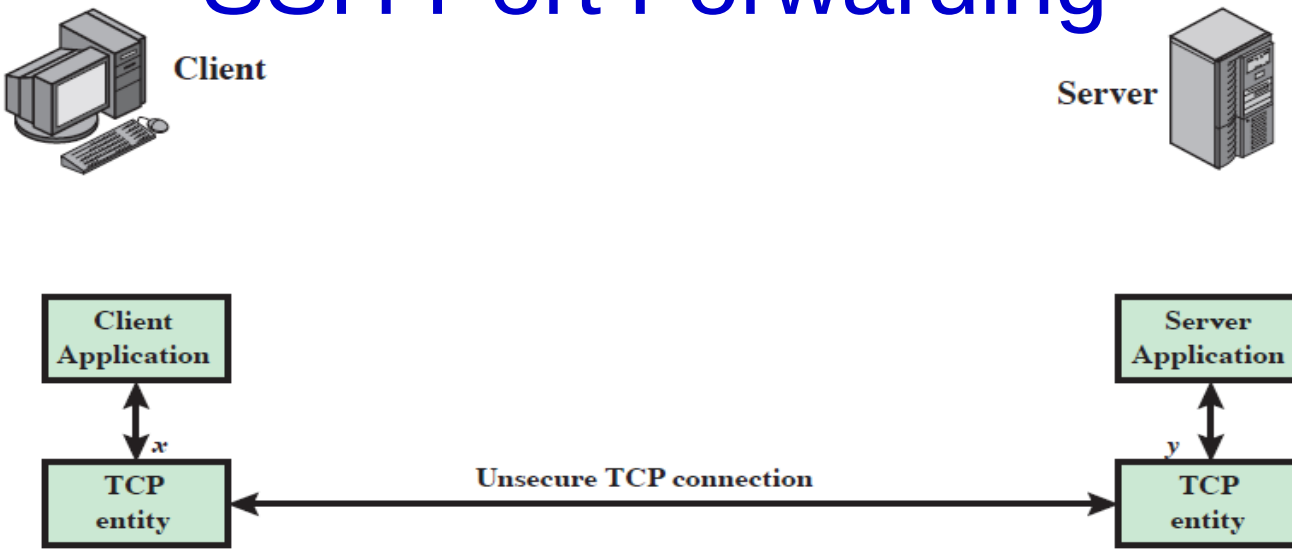


SSH Port Forwarding

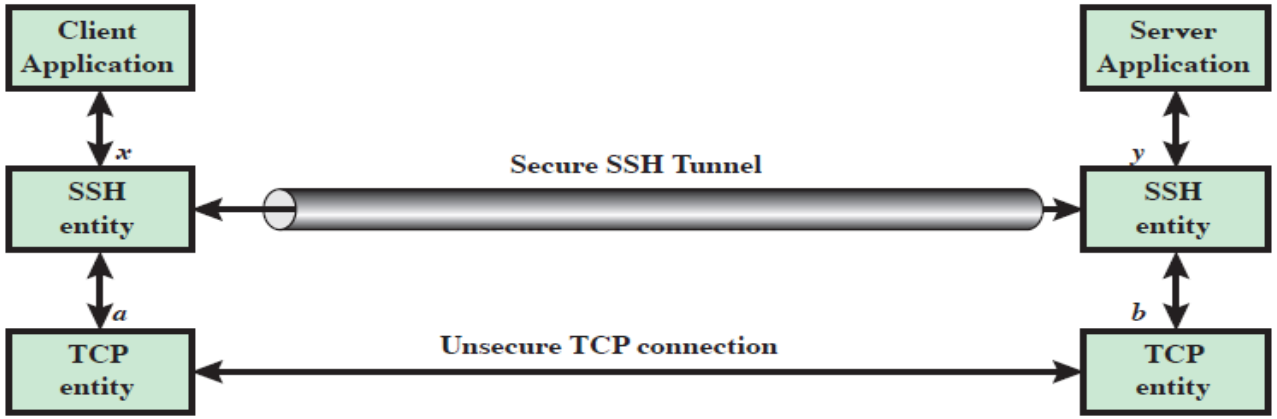
With SSH and port forwarding.



SSH Port Forwarding



(a) Connection via TCP



(b) Connection via SSH Tunnel

Port Forwarding

- Client sets up an SSH connection to the remote SSH server
- Select a local port x and configure SSH to accept traffic from this port destined for port y on a remote application server
- Client informs SSH Server to create a connection to the destination (application server port y)
- Client takes any bits sent to local port x and sends them to the server via an SSH session. The SSH server decrypts the bits and sends the plaintext to port y of the application server

SSH Applications

- Anonymous ftp for software updates, patches...
  - No client authentication needed, but clients want to be sure of origin and integrity of software.
- Secure ftp.
  - E.g.upload of webpages to webserver using sftp.
  - Server now needs to authenticate clients.
  - Username and password sufficient, transmitted over secure SSH transport layer protocol.
- Secure remote administration.
  - SysAdmin (client) sets up terminal on remote machine.
  - SysAdmin password protected by SSH transport layer protocol.
- Virtual Private Network.
  - E.g. use SSH + port forwarding to secure the communications of other applications.

What is a Virtual Private Network (VPN)?

- **VPN** is a generic term used to describe any combination of technologies used to secure a connection through an otherwise insecure network.
- A VPN *extends* a private network into a transit inter-network, such as the Internet or a shared network.
- The extension logically behaves like a *private*, *point-to-point* link in the transit inter-network, connecting two private networks across the inter-network.
- It allows, for example, authorised remote users to access a company network.
- Traditional VPN's rely on IPSec. There are also SSL/TLS and SSH based VPN's.