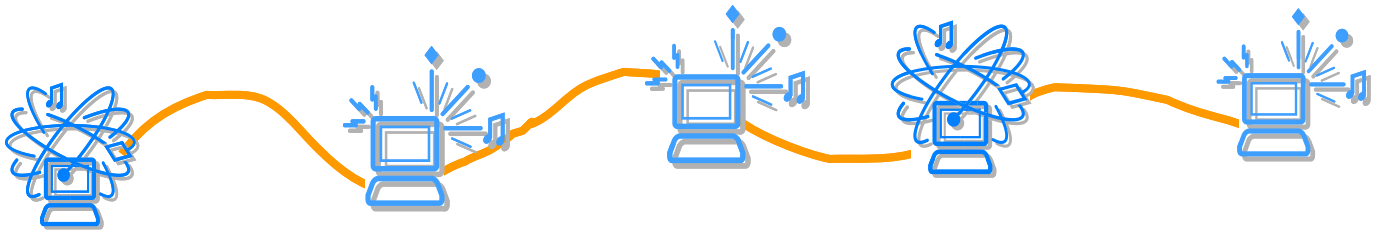
Segurança em Redes
SSH 2 - Secure Shell version 2



Redes de Comunicação de Dados
Área Departamental de Engenharia da Eletrónica e das Telecomunicações e de Computadores
Instituto Superior de Engenharia de Lisboa

- **Baseado em:**
  - IC3 - Network Security, M.Sc. in Information Security, Royal Holloway, University of London
  - “Cryptography and Network Security - Principles and Practice, Seventh edition”, William Stallings, Prentice-Hall, 2017

SSH

- SSH overview
- SSH architecture
- SSH security
- Port forwarding with SSH
- SSH applications

SSH - Introduction

Secure Shell (**SSH**) is a cryptographic network protocol for operating network services securely over an unsecured network.

- Typical applications include remote command-line, login, and remote command execution, but any network service can be secured with SSH.

- **SSH provides a secure channel over an unsecured network by using a client–server architecture**, connecting an SSH client application with an SSH server.

- The protocol specification distinguishes between **two major versions**, referred to as SSH-1 and SSH-2. It uses TCP, port 22, as transport protocol.

SSH - History

Version 1.x

In **1995**, Tatu Ylönen, a researcher at Helsinki University of Technology, Finland, designed the first version of the protocol (now called SSH-1) prompted by a password-sniffing attack at his university network.[15] The goal of SSH was to replace the earlier rlogin, TELNET, FTP[16] and rsh protocols, which did not provide strong authentication nor guarantee confidentiality.

Version 2.x

"Secsh" was the official Internet Engineering Task Force's (IETF) name for the IETF working group responsible for version 2 of the SSH protocol. In **2006**, a revised version of the protocol, SSH-2, was adopted as a standard. This version is incompatible with SSH-1. SSH-2 features both security and feature improvements over SSH-1.

[https://en.wikipedia.org/wiki/Secure_Shell]

Standards documentation

The following RFC publications by the IETF "secsh" working group document SSH-2 as a proposed Internet standard:

- RFC 4250 - The Secure Shell (SSH) Protocol Assigned Numbers
- RFC 4251 - The Secure Shell (SSH) Protocol Architecture
- RFC 4252 - The Secure Shell (SSH) Authentication Protocol
- RFC 4253 - The Secure Shell (SSH) Transport Layer Protocol
- RFC 4254 - The Secure Shell (SSH) Connection Protocol
- RFC 4255 - Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints
- RFC 4256 - Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)
- RFC 4335 - The Secure Shell (SSH) Session Channel Break Extension
- RFC 4344 - The Secure Shell (SSH) Transport Layer Encryption Modes
- RFC 4345 - Improved Arcfour Modes for the Secure Shell (SSH) Transport Layer Protocol

Standards documentation (cont.)

It was later modified and expanded by the following publications:

- RFC 4419 - Diffie-Hellman Group Exchange for the Secure Shell (SSH) Transport Layer Protocol (March 2006)
- RFC 4432 - RSA Key Exchange for the Secure Shell (SSH) Transport Layer Protocol (March 2006)
- RFC 4462 - Generic Security Service Application Program Interface (GSS-API) Authentication and Key Exchange for the Secure Shell (SSH) Protocol (May 2006)
- RFC 4716 - The Secure Shell (SSH) Public Key File Format (November 2006)
- RFC 4819 - Secure Shell Public Key Subsystem (March 2007)
- RFC 5647 - AES Galois Counter Mode for the Secure Shell Transport Layer Protocol (August 2009)
- RFC 5656 - Elliptic Curve Algorithm Integration in the Secure Shell Transport Layer (December 2009)
- RFC 6187 - X.509v3 Certificates for Secure Shell Authentication (March 2011)
- RFC 6239 - Suite B Cryptographic Suites for Secure Shell (SSH) (May 2011)

Standards documentation (cont.)

- RFC 6594 - Use of the SHA-256 Algorithm with RSA, Digital Signature Algorithm (DSA), and Elliptic Curve DSA (ECDSA) in SSHFP Resource Records (April 2012)
- RFC 6668 - SHA-2 Data Integrity Verification for the Secure Shell (SSH) Transport Layer Protocol (July 2012)
- RFC 7479 - Ed25519 SSHFP Resource Records (March 2015)
- RFC 5592 - Secure Shell Transport Model for the Simple Network Management Protocol (SNMP) (June 2009)
- RFC 6242 - Using the NETCONF Protocol over Secure Shell (SSH) (June 2011)
- draft-gerhards-syslog-transport-ssh-00 - SSH transport mapping for SYSLOG (July 2006)
- draft-ietf-secsh-filexfer-13 - SSH File Transfer Protocol (July 2006)

In addition, the OpenSSH project includes several vendor protocol specifications/extensions:

- OpenSSH PROTOCOL overview
- OpenSSH certificate/key overview

Secure Shell (SSH) (1 of 2)

- A protocol for secure network communications designed to be relatively simple and inexpensive to implement
- The initial version, SSH1 was focused on providing a secure remote logon facility to replace TELNET and other remote logon schemes that provided no security
- SSH also provides a more general client/server capability and can be used for such network functions as file transfer and e-mail

Secure Shell (SSH) (2 of 2)

- SSH2 fixes a number of security flaws in the original scheme and is documented as a proposed standard in IET F RFCs 4250 through 4256

- SSH client and server applications are widely available for most operating systems
  - Has become the method of choice for remote login and X tunneling
  - Is rapidly becoming one of the most pervasive applications for encryption technology outside of embedded systems

SSH Features

The major features and guarantees of the SSH protocol are:

- **Privacy of the data**, via strong encryption
- **Integrity of communications**, guaranteeing they haven't been altered
- **Authentication**, i.e., proof of identity of senders and receivers
- **Authorization**, i.e., access control to accounts
- **Forwarding or tunneling to encrypt other TCP/IP-based sessions**

SSH-1 *versus* SSH-2

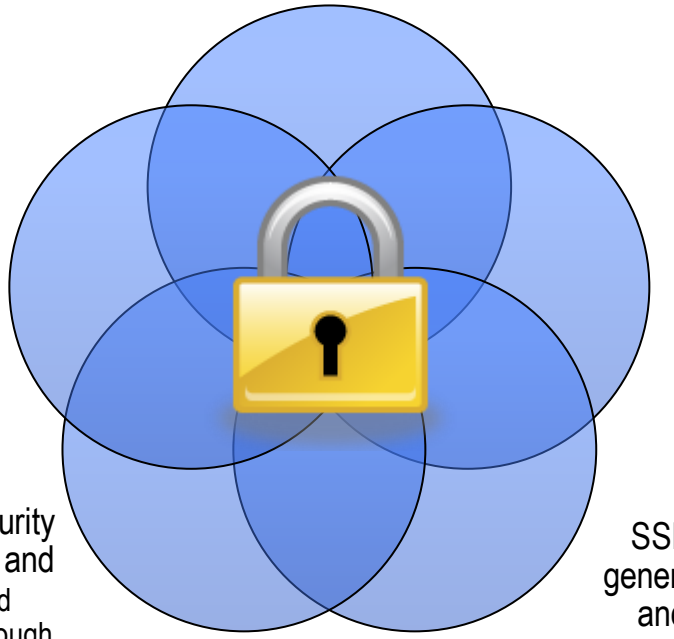
- Many vulnerabilities have been found in SSH-1 .
  - SSH-1 Insertion attack exploiting weak integrity mechanism (CRC-32) and unprotected packet length field.
  - SSHv1.5 session key retrieval attack (theoretical).
  - Man-in-the-middle attacks (using e.g. dsniff).
  - DoS attacks.
    - Overload server with connection requests.
    - Buffer overflows.
- But SSH-1 widely deployed.
- And SSH-1 supports:
  - Wider range of client authentication methods (.rhosts and Kerberos).
  - Wider range of platforms.

Secure Shell (SSH)

A protocol for secure network communications designed to be relatively simple and inexpensive to implement

SSH client and server applications are widely available for most operating systems

- Has become the method of choice for remote login and X tunneling
- Is rapidly becoming one of the most pervasive applications for encryption technology outside of embedded systems



The initial version, SSH1 was focused on providing a secure remote logon facility to replace TELNET and other remote logon schemes that provided no security

SSH2 fixes a number of security flaws in the original scheme and is documented as a proposed standard in IETF RFCs 4250 through 4256

SSH also provides a more general client/server capability and can be used for such network functions as file transfer and e-mail

SSH Features

Secure Remote Logins

- The client authenticates you to the remote computer's SSH server using an encrypted connection, meaning that your username and password are encrypted before the leave the local machine. The SSH server then logs you in, and your entire login session is encrypted as it travels between client and server.

Secure File Transfer

- Using SSH, the file can be transferred securely between machines with a single secure copy command. If the file were named yourfile, the command executed on firstaccount.com might be:

  $ scp yourfile metoo@secondaccount.com:

  When transmitted by scp, the file is automatically encrypted as it leaves firstaccount.com and decrypted as it arrives on secondaccount.com.

Secure Remote Command Execution

- The command (e.g. *ssh $machine /usr/ucb/w*) and its results are encrypted as they travel across the network, and strong authentication techniques may be used when connecting to the remote machines.

SSH Features (cont.)

Keys and Agents

- SSH has various authentication mechanisms, and the most secure is based on keys rather than passwords.

Access Control

- With SSH, you can give someone access to your account without revealing or changing your password, and with only the ability to run the email program, for example. No system-administrator privileges are required to set up this restricted access.

Port Forwarding

- Forwarding or tunneling means encapsulating another TCP-based service, such as Telnet or IMAP, within an SSH session. This brings the security benefits of SSH (privacy, integrity, authentication, authorization) to other TCP-based services. For example, an ordinary Telnet connection transmits your username, password, and the rest of your login session in the clear. By forwarding telnet through SSH, all of this data is automatically encrypted and integrity-checked, and you may authenticate using SSH credentials. SSH supports three types of forwarding: General TCP port forwarding operates as described earlier for any TCP-based service. X forwarding comprises additional features for securing the X protocol (i.e., X windows). The third type, agent forwarding, permits SSH clients to access SSH public keys on remote machines.

SSH Features (cont.)

Port Forwarding

- Forwarding or tunneling means encapsulating another TCP-based service, such as Telnet or IMAP, within an SSH session. This brings the security benefits of SSH (privacy, integrity, authentication, authorization) to other TCP-based services. For example, an ordinary Telnet connection transmits your username, password, and the rest of your login session in the clear. By forwarding telnet through SSH, all of this data is automatically encrypted and integrity-checked, and you may authenticate using SSH credentials.

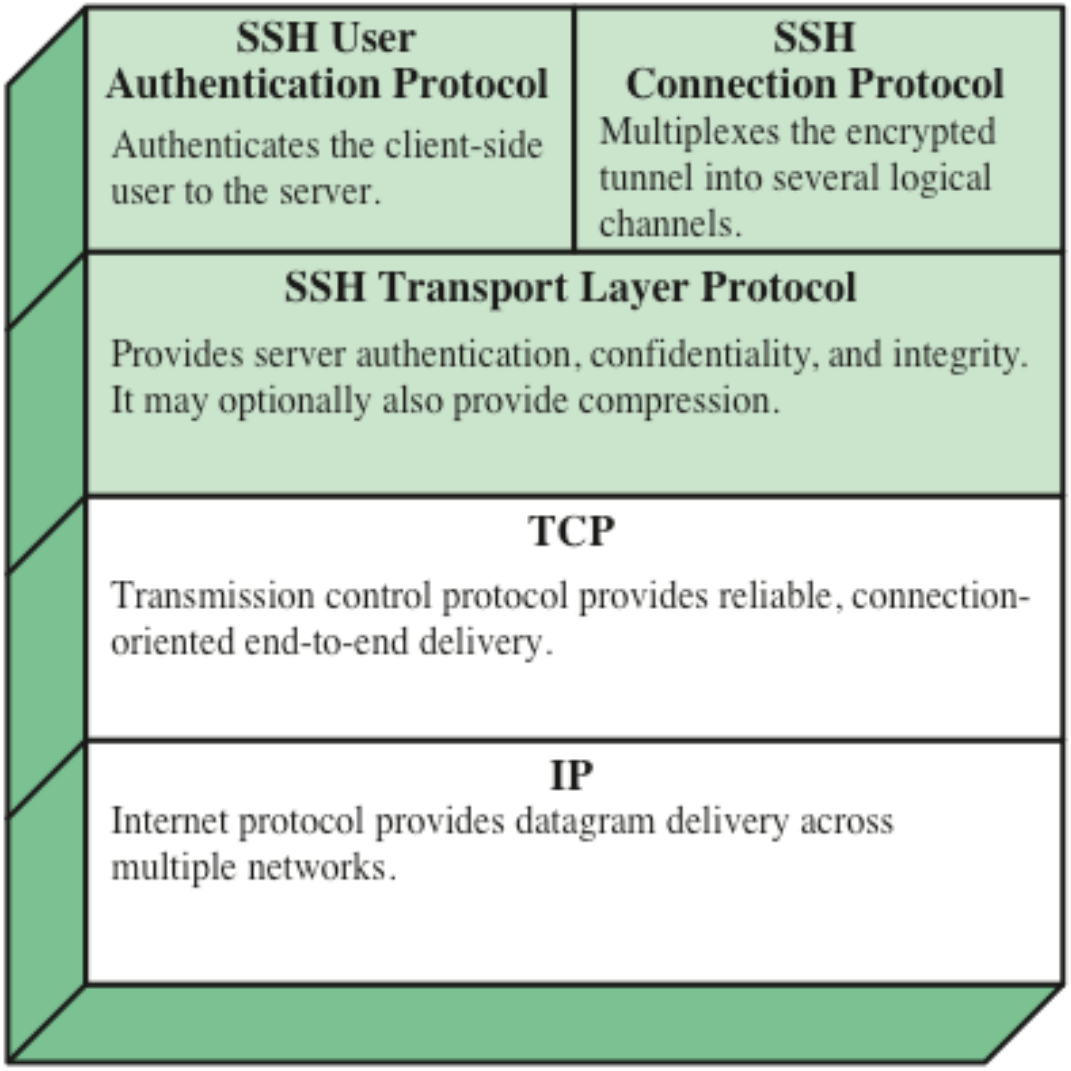
  SSH supports three types of forwarding:

  - **General TCP port forwarding** operates as described earlier for any TCP-based service.
  - **X forwarding** comprises additional features for securing the X protocol (i.e., X windows).
  - The third type, **agent forwarding**, permits SSH clients to access SSH public keys on remote machines.

SSH-2: Security objectives

- Server (nearly) always authenticated in transport layer protocol
- Client (nearly) always authenticated in authentication protocol
  - By public key (DSS, RSA, SPKI, OpenPGP).
  - Or simple password for particular application over secure channel.
- Establishment of a fresh, shared secret
  - Shared secret used to derive further keys, similar to SSL/IPSec.
  - For confidentiality and authentication in SSH transport layer protocol.
- Secure ciphersuite negotiation
  - Encryption, MAC, and compression algorithms.
  - Server authentication and key exchange methods.

SSH Protocol Stack



SSH-2: Architecture

SSH-2 (RFC 4251) adopts a three layer architecture:

- **SSH Transport Layer** (RFC 4253)
  - Initial connection.
  - Server authentication (almost always).
  - Sets up secure channel between client and server.

- **SSH User Authentication Layer** (RFC 4252)
  - Client authentication over secure transport layer channel.

- **SSH Connection Layer** (RFC 4254)
  - Supports multiple connections over a single transport layer protocol secure channel.
  - Efficiency (session re-use).

Transport Layer Protocol (1 of 2)

- Server authentication occurs at the transport layer, based on the server possessing a public/private key pair
- A server may have multiple host keys using multiple different asymmetric encryption algorithms
- Multiple hosts may share the same host key
- The server host key is used during key exchange to authenticate the identity of the host

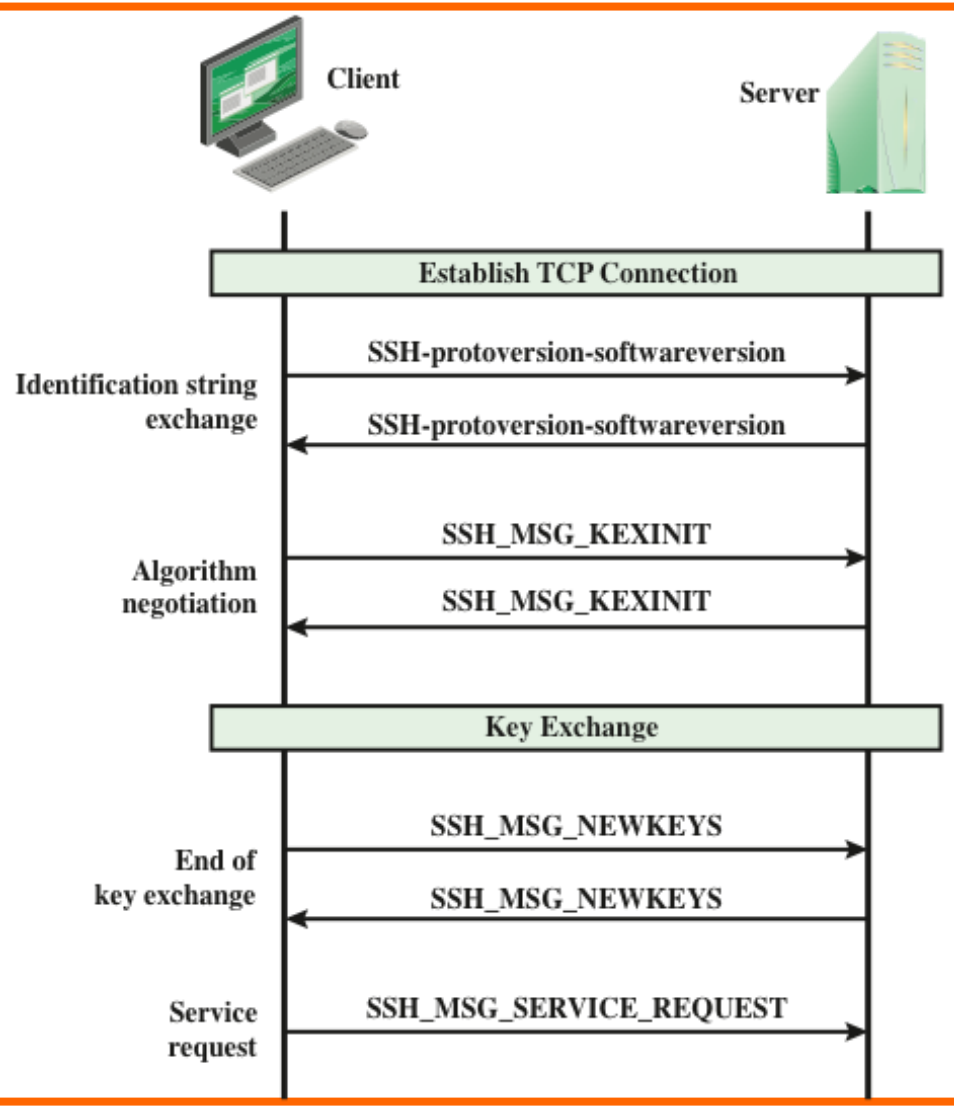
Transport Layer Protocol (2 of 2)

- RFC 4251 dictates two alternative trust models:
  - The client has a local database that associates each host name with the corresponding public host key
  - The host name-to-key association is certified by a trusted certification authority (CA); the client only knows the CA root key and can verify the validity of all host keys certified by accepted CAs

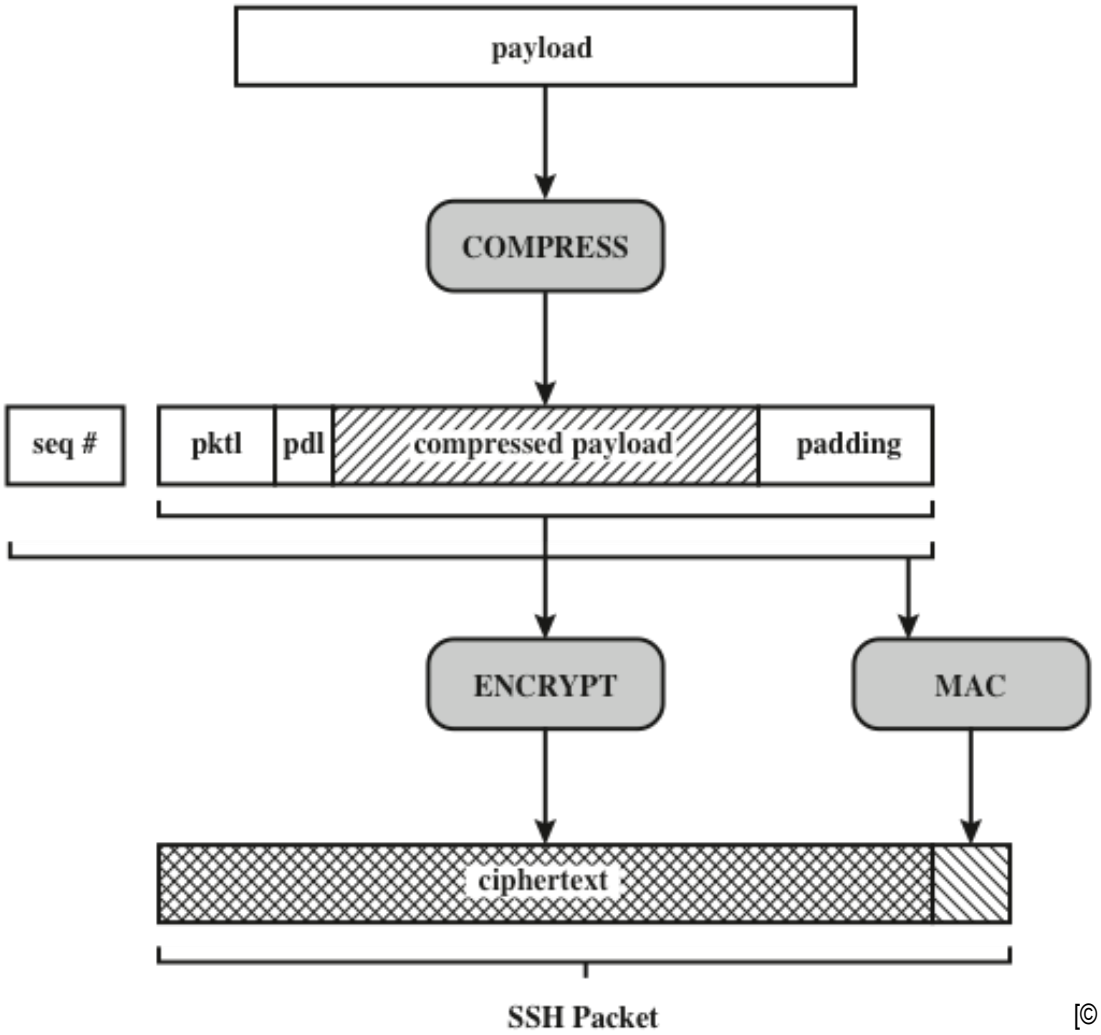
Transport Layer Protocol

- Server authentication occurs at the transport layer, based on the server possessing a public/private key pair
- A server may have multiple host keys using multiple different asymmetric encryption algorithms
- Multiple hosts may share the same host key
- The server host key is used during key exchange to authenticate the identity of the host
- RFC 4251 dictates two alternative trust models:
  - The client has a local database that associates each host name with the corresponding public host key
  - The host name-to-key association is certified by a trusted certification authority (CA); the client only knows the CA root key and can verify the validity of all host keys certified by accepted CAs

SSH Transport Layer Protocol Packet Exchange



SSH Transport Layer Protocol Packet Formation



pktl = packet length
pdl = padding length

SSH Transport Layer Cryptographic Algorithms

Cipher	
3des-cbc*	Three-key 3DES in CBC mode
blowfish-cbc	Blowfish in CBC mode
twofish256-cbc	Twofish in CBC mode with a 256-bit key
twofish192-cbc	Twofish with a 192-bit key
twofish128-cbc	Twofish with a 128-bit key
aes256-cbc	AES in CBC mode with a 256-bit key
aes192-cbc	AES with a 192-bit key
aes128-cbc**	AES with a 128-bit key
Serpent256-cbc	Serpent in CBC mode with a 256-bit key
Serpent192-cbc	Serpent with a 192-bit key
Serpent128-cbc	Serpent with a 128-bit key
arcfour	RC4 with a 128-bit key
cast128-cbc	CAST-128 in CBC mode

MAC algorithm	
hmac-sha1*	HMAC-SHA1; digest length = key length = 20
hmac-sha1-96**	First 96 bits of HMAC-SHA1; digest length = 12; key length = 20
hmac-md5	HMAC-MD5; digest length = key length = 16
hmac-md5-96	First 96 bits of HMAC-MD5; digest length = 12; key length = 16

Compression algorithm	
none*	No compression
zlib	Defined in RFC 1950 and RFC 1951

* = Required
** = Recommended

SSH Transport Layer Cryptographic Algorithms (1 of 4)

Cipher	
3des-cbc*	Three-key 3DES in CBC mode
blowfish-cbc	Blowfish in CBC mode
twofish256-cbc	Two fish in CBC mode with a 256-bit key
twofish192-cbc	Two fish with a 192-bit key
twofish128-cbc	Two fish with a 128-bit key
aes256-cbc	AES in CBC mode with a 256-bit key

SSH Transport Layer Cryptographic Algorithms (2 of 4)

Cipher	
aes192-cbc	AES with a 192-bit key
aes128-cbc**	AES with a 128-bit key
Serpent256-cbc	Serpent in CBC mode with a 256-bit key
Serpent192-cbc	Serpent with a 192-bit key
Serpent128-cbc	Serpent with a 128-bit key
arcfour	RC4 with a 128-bit key
cast128-cbc	CAST-128 in CBC mode

SSH Transport Layer Cryptographic Algorithms (3 of 4)

MAC Algorithm	
hmac-sha1*	HMAC-SHA1; digest length = key length = 20
hmac-sha1-96**	First 96 bits of HMACSHA1; digest length = 12; key length = 20
hmac-md5	HMAC-MD5; digest length = key length = 16
hmac-md5-96	First 96 bits of HMAC-MD5; digest length = 12; key length = 16

SSH Transport Layer Cryptographic Algorithms (4 of 4)

Compression algorithm	
none*	No compression
zlib	Defined in RFC 1950 and RFC 1951

Authentication Methods (1 of 2)

- **Public key**
  - The client sends a message to the server that contains the client's public key, with the message signed by the client's private key
  - When the server receives this message, it checks whether the supplied key is acceptable for authentication and, if so, it checks whether the signature is correct
- **Password**
  - The client sends a message containing a plaintext password, which is protected by encryption by the Transport Layer Protocol

Authentication Methods

- Publickey
  - The client sends a message to the server that contains the client's public key, with the message signed by the client's private key
  - When the server receives this message, it checks whether the supplied key is acceptable for authentication and, if so, it checks whether the signature is correct
- Password
  - The client sends a message containing a plaintext password, which is protected by encryption by the Transport Layer Protocol
- Hostbased
  - Authentication is performed on the client's host rather than the client itself
  - This method works by having the client send a signature created with the private key of the client host
  - Rather than directly verifying the user's identity, the SSH server verifies the identity of the client host

Authentication Methods (2 of 2)

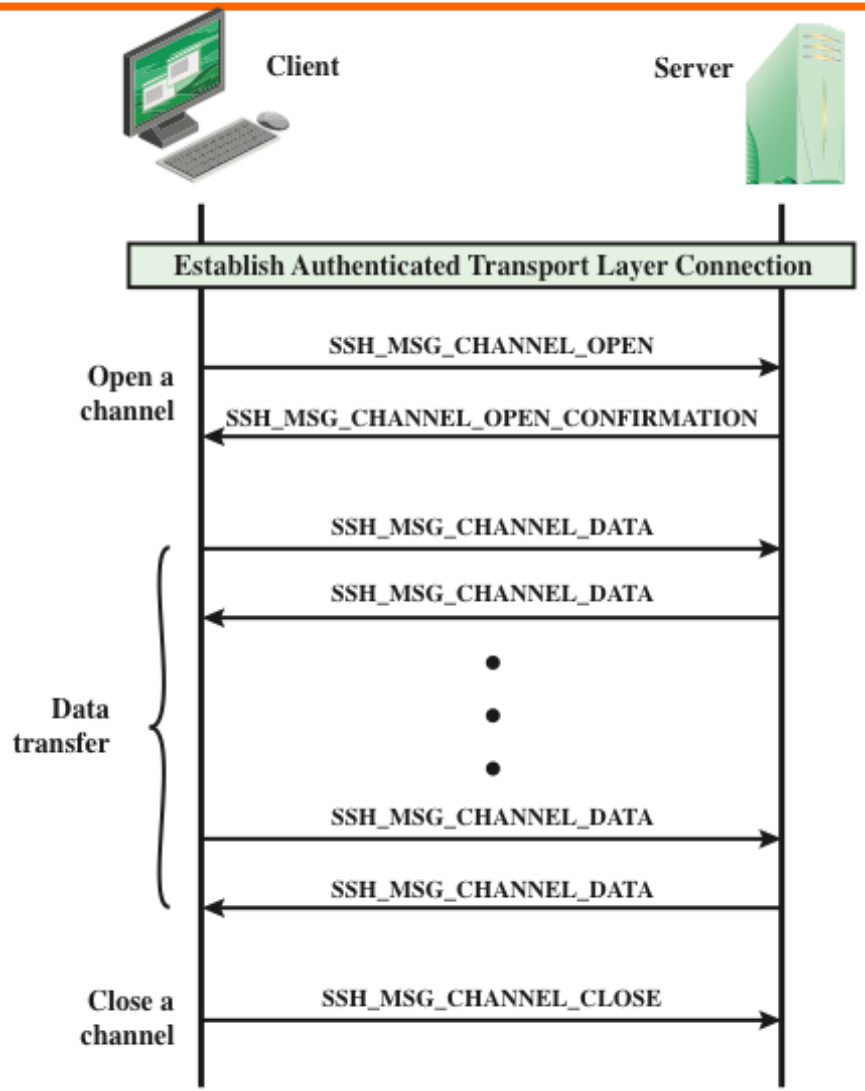
- **Hostbased**
  - The client sends a message containing a plaintext password, which is protected by encryption by the Transport Layer Protocol
  - Authentication is performed on the client's host rather than the client itself
  - This method works by having the client send a signature created with the private key of the client host
  - Rather than directly verifying the user's identity, the SSH server verifies the identity of the client host

Connection Protocol

- The SSH Connection Protocol runs on top of the SSH Transport Layer Protocol and assumes that a secure authentication connection is in use
  - The secure authentication connection, referred to as a *tunnel,* is used by the Connection Protocol to multiplex a number of logical channels

- Channel mechanism
  - All types of communication using SSH are supported using separate channels
  - Either side may open a channel
  - For each channel, each side associates a unique channel number
  - Channels are flow controlled using a window mechanism
  - No data may be sent to a channel until a message is received to indicate that window space is available
  - The life of a channel progresses through three stages: opening a channel, data transfer, and closing a channel

Example SSH Connection Protocol Message Exchange



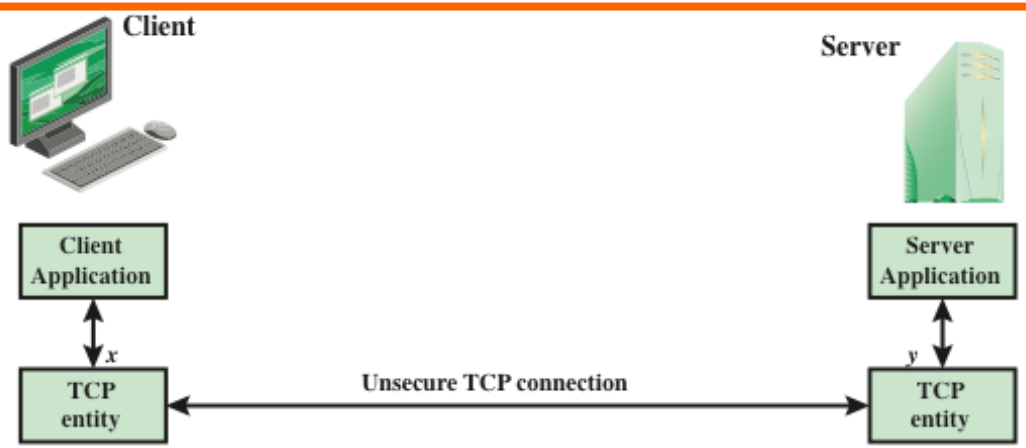
Connection Layer Protocol (1 of 2)

- The SSH Connection Protocol runs on top of the SSH Transport Layer Protocol and assumes that a secure authentication connection is in use
  - The secure authentication connection, referred to as a tunnel, is used by the Connection Protocol to multiplex a number of logical channels
- Channel mechanism
  - All types of communication using SSH are supported using separate channels

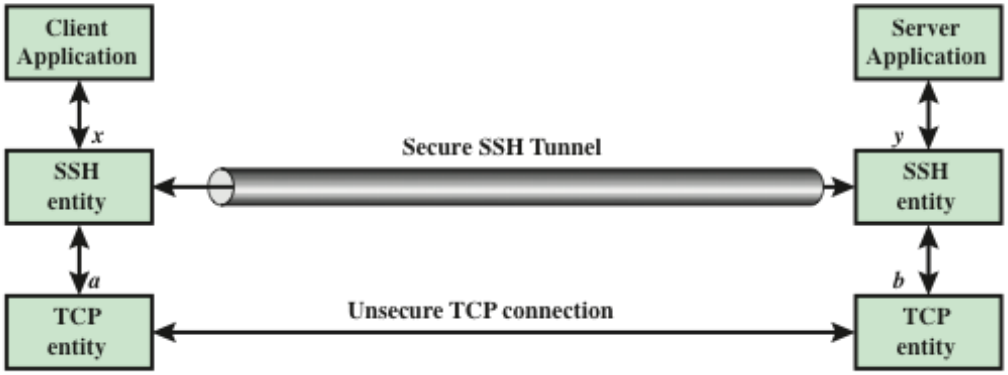
Connection Layer Protocol (2 of 2)

- Either side may open a channel
- For each channel, each side associates a unique channel number
- Channels are flow controlled using a window mechanism
- No data may be sent to a channel until a message is received to indicate that window space is available
- The life of a channel progresses through three stages: opening a channel, data transfer, and closing a channel

SSH Transport Layer Packet Exchanges

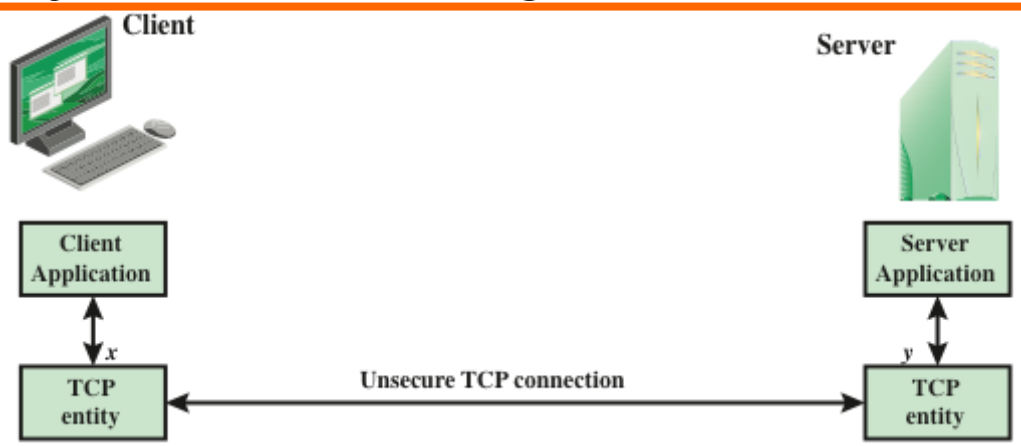


(a) Connection via TCP

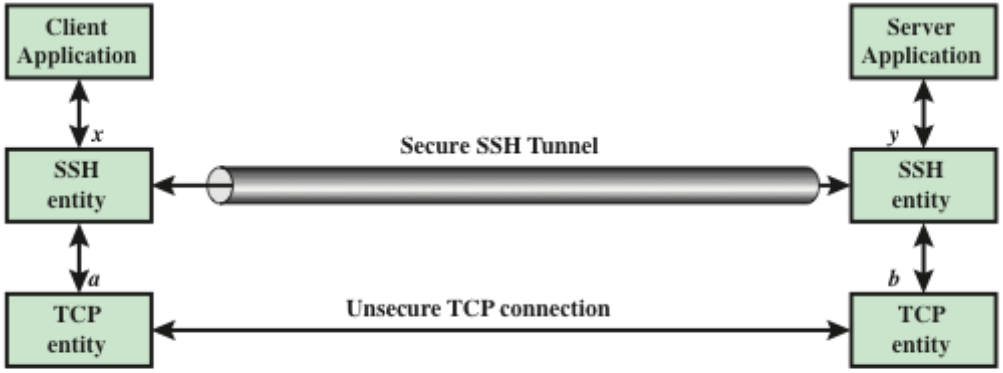


(b) Connection via SSH Tunnel

SSH Transport Layer Packet Exchanges

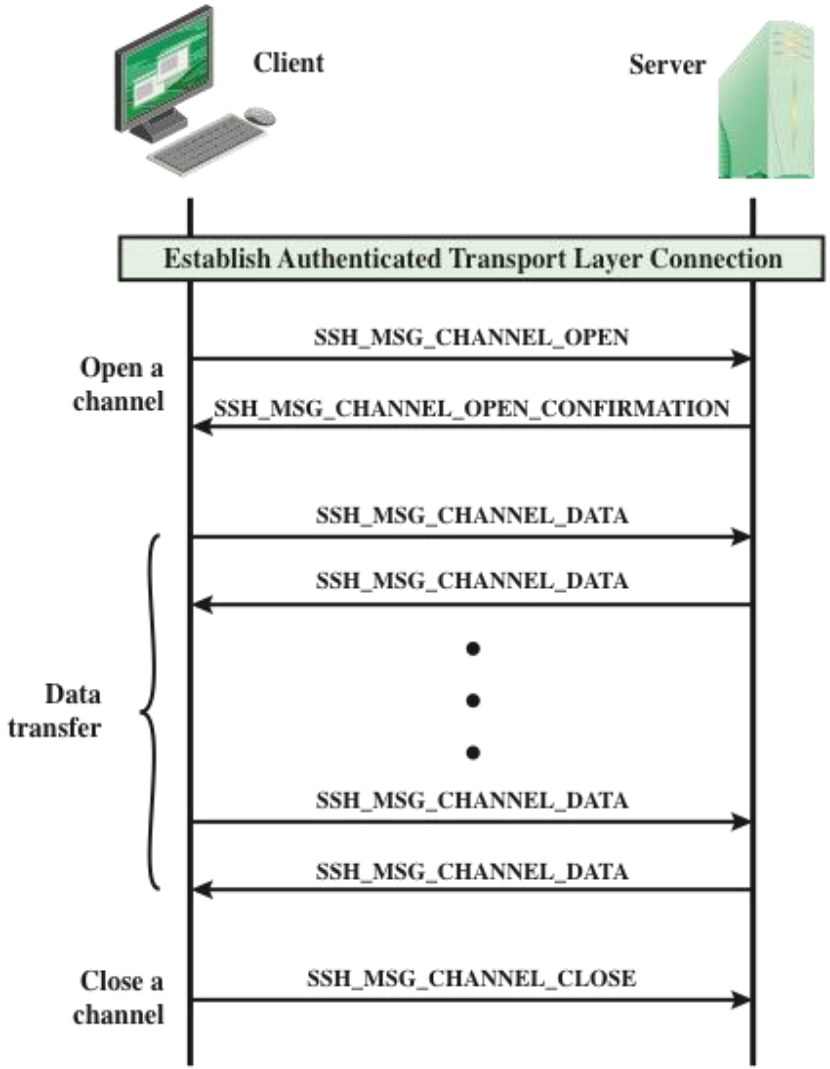


(a) Connection via TCP



(b) Connection via SSH Tunnel

Example of SSH Connection Protocol Message Exchanges



Channel Types

Four channel types are recognized in the SSH Connection Protocol specification

Session

- The remote execution of a program
- The program may be a shell, an application such as file transfer or e-mail, a system command, or some built-in subsystem
- Once a session channel is opened, subsequent requests are used to start the remote program

X11

- Refers to the X Window System, a computer software system and network protocol that provides a graphical user interface (GUI) for networked computers
- X allows applications to run on a network server but to be displayed on a desktop machine

Forwarded-tcpip

- Remote port forwarding

Direct-tcpip

- Local port forwarding

Channel Types (1 of 2)

- Four channel types are recognized in the SSH Connection Protocol specification

Session

- The remote execution of a program
- The program may be a shell, an application such as file transfer or e-mail, a system command, or some built-in subsystem
- Once a session channel is opened, subsequent requests are used to start the remote program

Channel Types (2 of 2)

X11

- Refers to the X Window System, a computer software system and network protocol that provides a graphical user interface (GUI) for networked computers
- X allows applications to run on a network server but to be displayed on a desktop machine

Forwarded-tcpip

- Remote port forwarding

Direct-tcpip

- Local port forwarding

Port Forwarding

- One of the most useful features of SSH
- Provides the ability to convert any insecure TCP connection into a secure SSH connection (also referred to as SSH tunneling)
- Incoming TCP traffic is delivered to the appropriate application on the basis of the port number (a port is an identifier of a user of TCP)
- An application may employ multiple port numbers

Port Forwarding

- One of the most useful features of SSH
- Provides the ability to convert any insecure TCP connection into a secure SSH connection (also referred to as SSH tunneling)
- Incoming TCP traffic is delivered to the appropriate application on the basis of the port number (a port is an identifier of a user of TCP)
- An application may employ multiple port numbers

SSH-2 Enhancements

These are intended for performance enhancements of SSH products:

- **SSH-over-SCTP**: support for SCTP rather than TCP as the connection oriented transport layer protocol.[27]
- **ECDSA**: support for elliptic curve DSA rather than DSA or RSA for signing.[28]
- **ECDH**: support for elliptic curve Diffie–Hellman rather than plain Diffie–Hellman for encryption key exchange.[28]
- **UMAC**: support for UMAC rather than HMAC for MAC/integrity.[29]

6.3 Comparing IPSec, SSL/TLS, SSH

- All three have initial (authenticated) key establishment then key derivation.
  - IKE in IPSec
  - Handshake Protocol in SSL/TLS (can be unauthenticated!)
  - Authentication Protocol in SSH
- All protect ciphersuite negotiation.
- All three use keys established to build a ‘secure channel’.

Comparing IPSec, SSL/TLS, SSH

- Operate at different network layers.
  - This brings pros and cons for each protocol suite.
  - Recall `Where shall we put security?' discussion.
  - Naturally support different application types, can all be used to build VPNs.
- All practical, but not simple.
  - Complexity leads to vulnerabilities.
  - Complexity makes configuration and management harder.
  - Complexity can create computational bottlenecks.
  - Complexity necessary to give both flexibility and security.

Comparing IPSec, SSL/TLS, SSH

Security of all three undermined by:

- Implementation weaknesses.
- Weak server platform security.
  - Worms, malicious code, rootkits,…
- Weak user platform security.
  - Keystroke loggers, malware,…
- Limited deployment of certificates and infrastructure to support them.
  - Especially client certificates.
- Lack of user awareness and education.
  - Users click-through on certificate warnings.
  - Users fail to check URLs.
  - Users send sensitive account details to bogus websites (“phishing”) in response to official-looking e-mail.

Summary

- Web security considerations
  - Web security threats
  - Web traffic security approaches
- Secure sockets layer
  - SSL architecture
  - SSL record protocol
  - Change cipher spec protocol
  - Alert protocol
  - Handshake protocol
  - Cryptographic computations
  - Heartbeat protocol
  - SSL/TLS attacks
  - TLSv1.3
- Secure shell (SSH)
  - Transport layer protocol
  - User authentication protocol
  - Communication protocol
- HTTPS
  - Connection initiation
  - Connection closure

Summary

- Web security considerations
  - Web security threats
  - Web traffic security approaches
- Secure sockets layer
  - SSL architecture
  - SSL record protocol
  - Change cipher spec protocol
  - Alert protocol
  - Handshake protocol
  - Cryptographic computations
  - Heartbeat protocol
  - SSL/TLS attacks
  - TLSv1.3

- Secure shell (SSH)
  - Transport layer protocol
  - User authentication protocol
  - Communication protocol
- HTTPS
  - Connection initiation
  - Connection closure