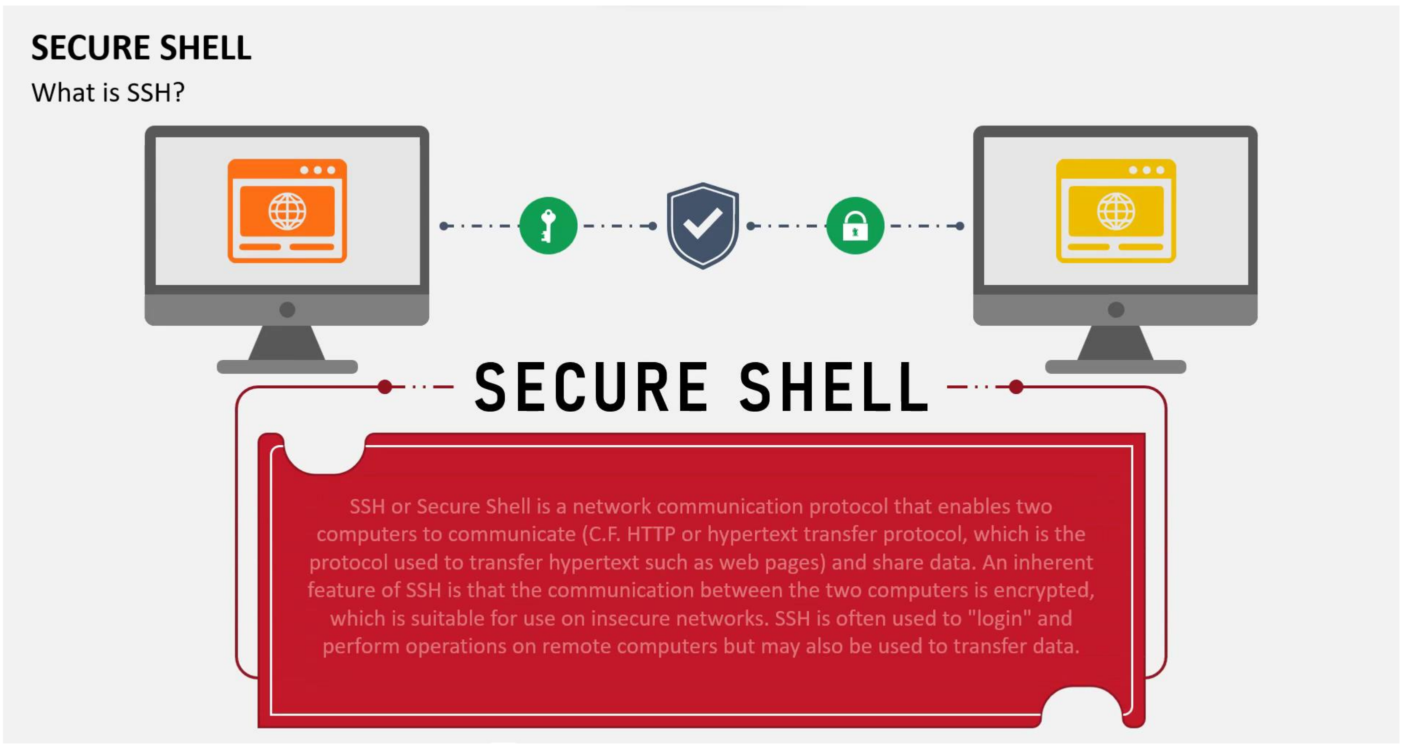
SECURE SHELL (SSH Protocol)

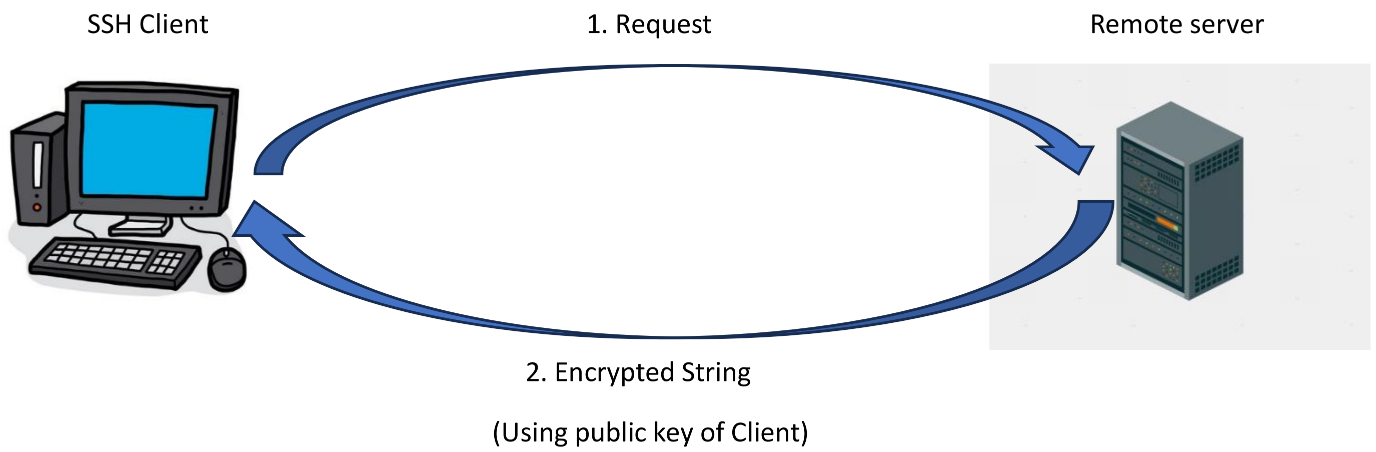
- It is Cryptographical network protocol for operating network services securely over an unsecure network.
- It is secure alternative to the non-protected login protocol (like Telnet, Rlogin, rsh ,rcp , ftp) and insecure file transfer method (like FTP-File Transfer Protocol).
- It uses Client Server Architecture.
- It uses public key cryptography/asymmetric key cryptography to authenticated the remote server, i.e. to verify it identify to the remote server.
- Comes with all Linux distribution, Mac OS X, AIX, Sun Solaris, OpenBSD and other Unix variants
- Ported to other operating systems, such as Windows, Palm OS, Amiga, etc.
- It is not a command interpreter.
- It creates secure channel for running commands on remote computer.
- It will not protect against trojans, viruses, etc.



SSH or Secure Shell is a network communication protocol that enables two computers to communicate (C.F. HTTP or hypertext transfer protocol, which is the protocol used to transfer hypertext such as web pages) and share data. An inherent feature of SSH is that the communication between the two computers is encrypted, which is suitable for use on insecure networks. SSH is often used to "login" and perform operations on remote computers but may also be used to transfer data.

TERMINOLOGY:

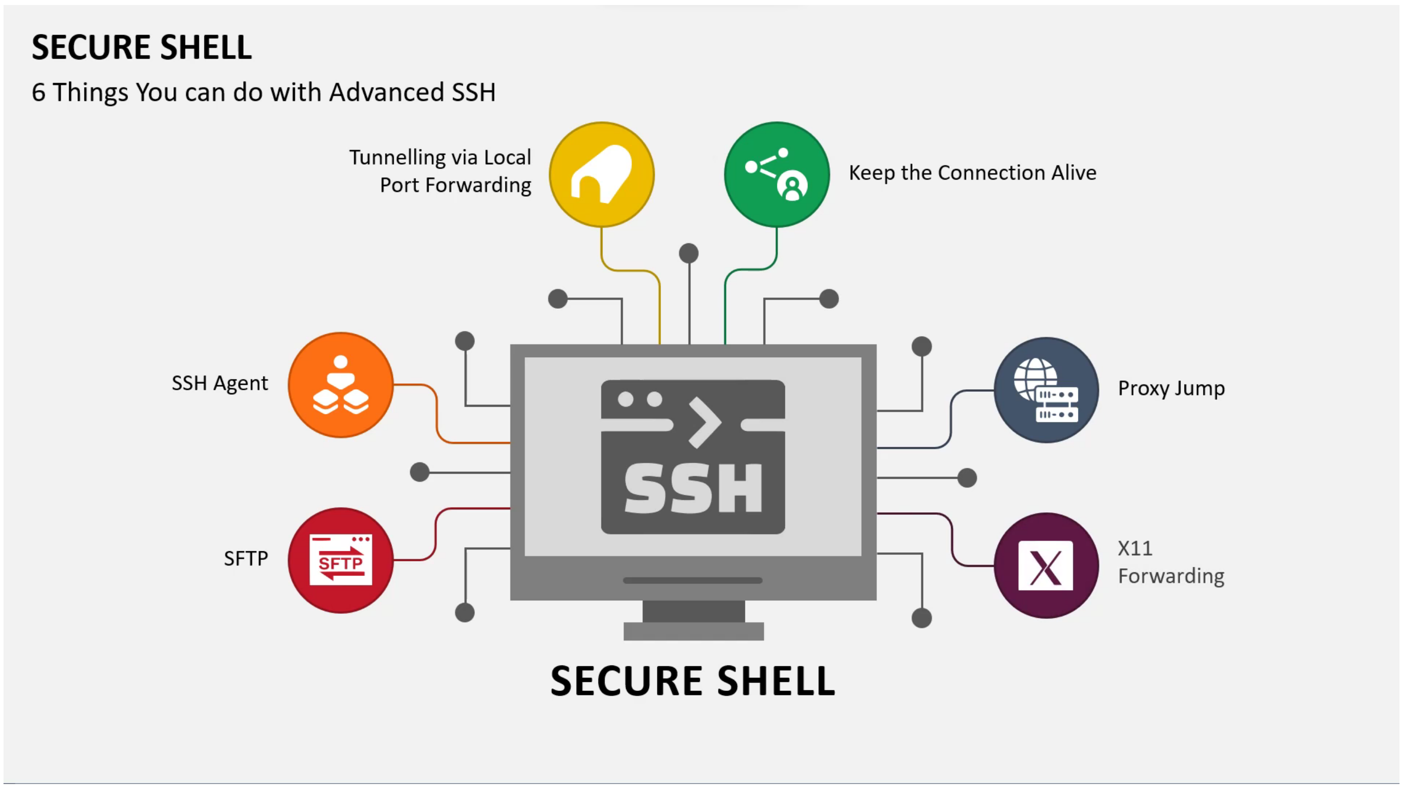
- SSH – Generic term used for SSH protocol
- ssh – Client command for running remote command
- sshd – Server program
- SSH-1 – Version 1 of the protocol
- SSh-2 – Version 2 of the protocol
- OpenSSH – Product from open BSD project

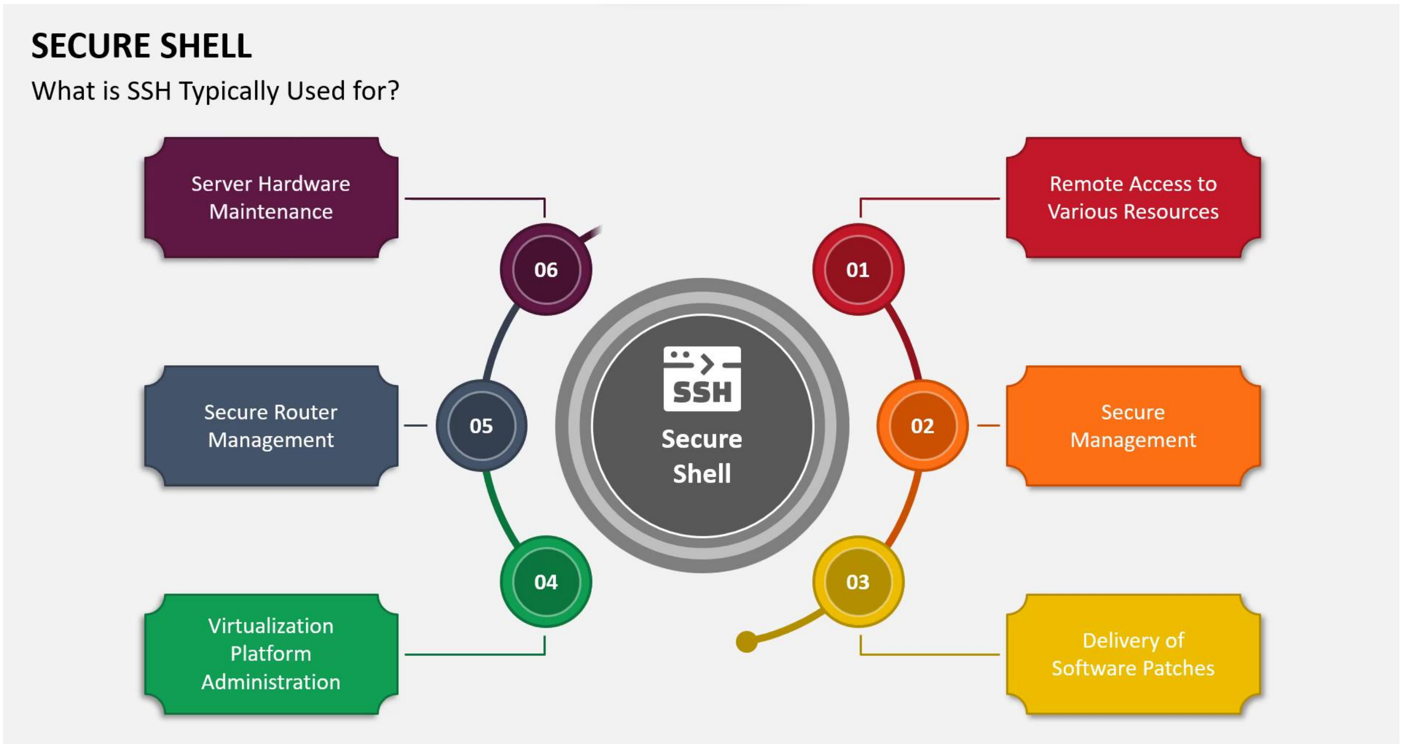


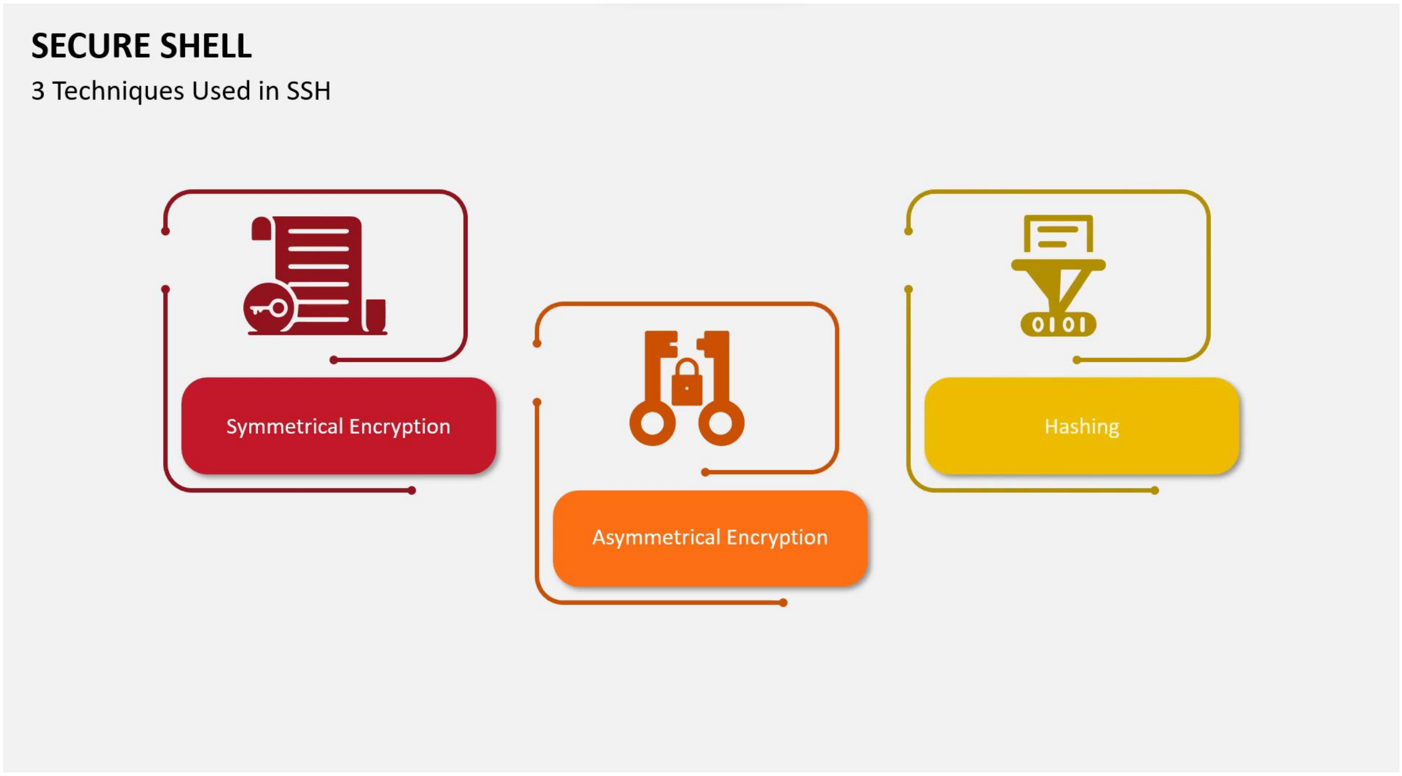
1. Client request to access the server, server will check its files for the public key of the client.
2. Now, server will generate a random string and encrypt it using clients public key (and this can only be decrypted by client's private key) and send it to client.
3. Client will decrypt the message and acknowledge it (i.e. send back to server).
   Now server will know that – yes this was the message which was send to the client and thus, the user is authentic and now, the client is verified.
4. After that SSH creates a tunnel between the client and server and we can pass our data (in encrypted form over this tunnel).
5. The encryption used by SSH is intended to provide confidentiality and integrity of data over an unsecured network (like internet).

- How this system can be used?

⇨ We have various types of SSH clients on our various types of machines
  - For MAC/Linux – terminal (it acts as SSH Client)
  - Windows – PuTTY


SECURE SHELL
6 Things You can do with Advanced SSH
Tunnelling via Local Port Forwarding
Keep the Connection Alive
SSH Agent
Proxy Jump
SFTP
X11 Forwarding
SSH
SECURE SHELL


SECURE SHELL
What is SSH Typically Used for?
Server Hardware Maintenance
06
Secure Router Management
05
Virtualization Platform Administration
04
SSH
Secure Shell
Remote Access to Various Resources
01
Secure Management
02
Delivery of Software Patches
03



- **Symmetric Encryption in SSH:**

1. **Session Key Exchange:**
   - During the initial handshake, the client and server don't directly exchange the actual symmetric key they'll use for encryption. This would be risky as the key itself would be exposed during transmission.
   - Instead, they use their asymmetric key pairs (public and private keys) to securely negotiate a shared secret, which is then used to derive the symmetric session key.

2. **Key Derivation Function (KDF):**
    - The shared secret obtained through public key cryptography is not directly used as the symmetric key. A KDF, a secure cryptographic function, stretches the shared secret into a stronger and longer symmetric key suitable for data encryption.
3. **Encryption and Decryption:**
    - The agreed-upon symmetric key is used to encrypt and decrypt all data exchanged during the SSH session, including commands, responses, and file transfers. This ensures confidentiality, meaning only the authorized parties (client and server) can understand the data.
4. **Different Encryption Modes:**
    - SSH supports various symmetric encryption algorithms and modes, such as AES-256 in Cipher Block Chaining (CBC) mode or CTR (Counter) mode. The specific choice depends on the desired balance between security, performance, and compatibility.
5. **Key Ephemerality:**
    - The symmetric session key is generated for each individual SSH session and discarded after the connection is closed. This mitigates the risk of compromising the key compromising future sessions.

- **Asymmetric Encryption in SSH:**

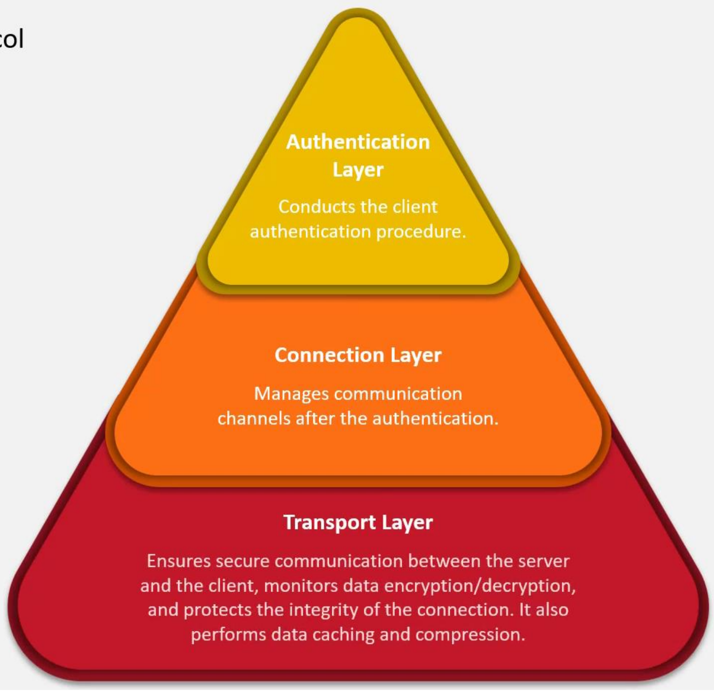
1. **Server Authentication:**
    - When you initiate an SSH connection, the server sends its public key fingerprint. You verify this fingerprint against a trusted source (e.g., server administrator) to ensure you're connecting to the intended server and not an imposter.
2. **Secure Key Exchange:**
    - The client and server use their own asymmetric key pairs (public and private keys) to negotiate a temporary, shared secret for deriving the symmetric session key used for data encryption. This ensures the key exchange is secure even without directly transmitting the actual key itself.
3. **Public Key Authentication:**
    - Instead of using a password, you can configure SSH to use your private key for authentication. The server challenges the client with a random number, and the client signs it with its private key. The server verifies the signature using the client's public key, granting access if valid. This method eliminates the risk of password sniffing and offers stronger authentication.
4. **Host-Based Authentication:**
    - The server's public key can be placed on the client's machine. During connection, the server signs a challenge with its private key, and the client verifies the signature using the stored public key. This can be an additional layer of security to ensure you're connecting to the authorized server.

- **Hasing in SSH:**

1. **Password Authentication:**
    - When using password authentication, your password is never sent directly to the server. Instead, the client calculates a cryptographic hash (e.g., SHA-256) of your password and sends it to the server.
    - The server stores its own hashed version of the password (typically during account creation).
    - If the received hash matches the stored one, authentication is successful. This prevents attackers from sniffing and stealing actual passwords.
2. **Message Authentication Codes (MACs):**
    - After establishing the encrypted connection, SSH uses MACs to ensure the integrity of all data transfer.
    - The client and server calculate a hash of each message (command, response, file) using a shared secret key and a specific MAC algorithm (e.g., HMAC-SHA1).
    - Both sides attach the calculated MAC to the message and send it across the encrypted channel.
    - Upon receiving the message, the recipient recalculates the MAC using the same algorithm and secret key.
    - If the calculated and received MACs match, the message integrity is confirmed. Any alteration during transmission would result in mismatched MACs, alerting both parties to a potential security breach.

SECURE SHELL

Three Layers of SSH Protocol



Authentication Layer

Conducts the client authentication procedure.

Connection Layer

Manages communication channels after the authentication.

Transport Layer

Ensures secure communication between the server and the client, monitors data encryption/decryption, and protects the integrity of the connection. It also performs data caching and compression.

Application Layer	**ssh – connection** Session multiplexing, X11 and port forwarding, remote command execution, SOCKS proxy, etc.
	ssh – userauth User authentication using public key, password, host based, etc.
	ssh – transport Initial key exchange and server authentication, setup encryption