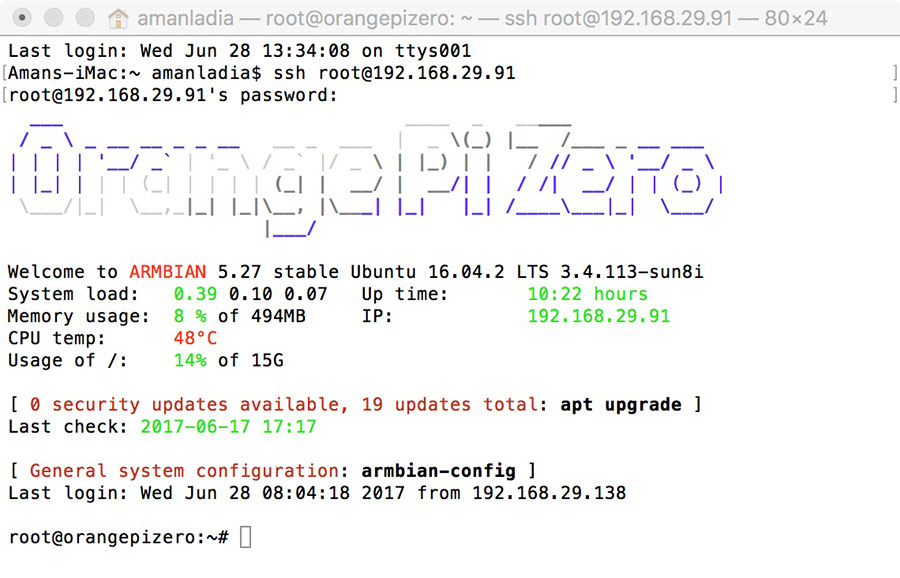


* **86**SHARES
* [Share](https://www.facebook.com/sharer/sharer.php?u=https://www.hostinger.in/tutorials/ssh-tutorial-how-does-ssh-work&t=SSH%20Tutorial%20for%20Beginners%20-%20How%20Does%20SSH%20Work&redirect_uri=https://www.hostinger.in/tutorials?sharing-thankyou=yes)
* [Tweet](https://www.hostinger.in/tutorials/ssh-tutorial-how-does-ssh-work)
* [+1](https://plus.google.com/share?url=https://www.hostinger.in/tutorials/ssh-tutorial-how-does-ssh-work)
* [Reddit](http://reddit.com/submit?url=https://www.hostinger.in/tutorials/ssh-tutorial-how-does-ssh-work&title=SSH%20Tutorial%20for%20Beginners%20-%20How%20Does%20SSH%20Work)
* [Share](http://www.linkedin.com/shareArticle?mini=true&ro=true&trk=EasySocialShareButtons&title=SSH%20Tutorial%20for%20Beginners%20-%20How%20Does%20SSH%20Work&url=https://www.hostinger.in/tutorials/ssh-tutorial-how-does-ssh-work)

**What is SSH**

SSH, or Secure Shell, is a remote administration protocol that allows users to control and modify their remote servers over the Internet. The service was created as a secure replacement for the unencrypted Telnet and uses cryptographic techniques to ensure that all communication to and from the remote server happens in an encrypted manner. It provides a mechanism for authenticating a remote user, transferring inputs from the client to the host, and relaying the output back to the client.

The Figure Below shows a typical SSH Window. Any Linux or macOS user can SSH into their remote server directly from the terminal window. Windows users can take advantage of[SSH clients like Putty](https://www.hostinger.com/tutorials/how-to-connect-to-your-account-using-putty).  You can execute shell commands in the same manner as you would if you were physically operating the remote computer.

[](https://www.hostinger.com/tutorials/wp-content/uploads/sites/2/2017/07/shell-snapshot.jpg)  
This SSH tutorial will cover the basics of how does ssh work, along with the underlying technologies used by the protocol to offer a secured method of remote access. It will cover the different layers and types of encryption used, along with the purpose of each layer.

**How Does SSH Work**

If you’re using Linux or Mac, then using SSH is very simple. If you use Windows, you will need to utilize an SSH client to open SSH connections. The most popular SSH client is PuTTY, which you [can learn more about here](https://www.hostinger.com/tutorials/how-to-connect-to-your-account-using-putty).

For Mac and Linux users, head over to your **terminal** program and then follow the procedure below:

The SSH command consists of 3 distinct parts:

ssh {user}@{host}

The SSH key command instructs your system that you want to open an encrypted Secure Shell Connection. **{user}** represents the account you want to access. For example, you may want to access the **root** user, which is basically synonymous for system administrator with complete rights to modify anything on the system. **{host}** refers to the computer you want to access. This can be an IP Address **(e.g. 244.235.23.19)** or a domain name (e.g. www.xyzdomain.com).

When you hit enter, you will be prompted to enter the password for the requested account. When you type it in, nothing will appear on the screen, but your password is, in fact being transmitted. Once you’re done typing, hit enter once again. If your password is correct, you will be greeted with a remote terminal window.

If you want to learn about some more SSH commands, [find them out here](https://www.hostinger.com/tutorials/ssh/basic-ssh-commands).

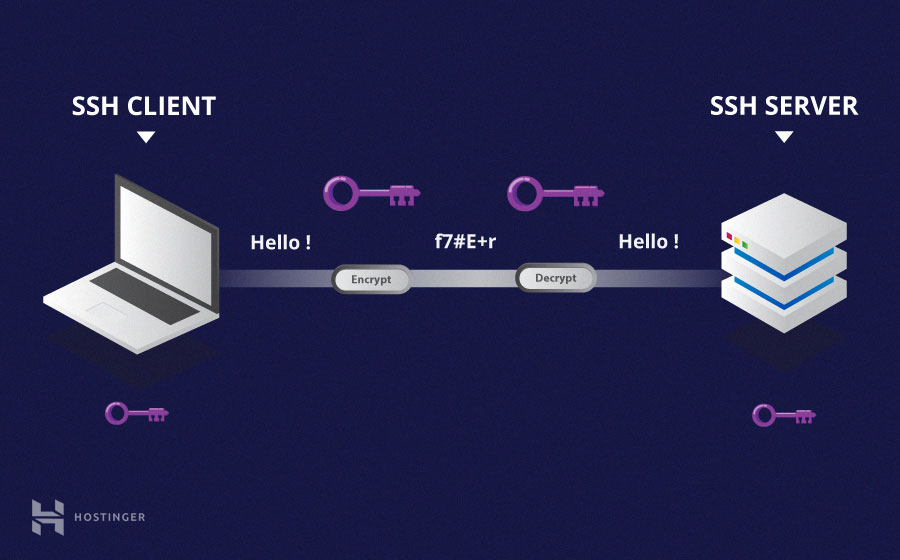
**Understanding Different Encryption Techniques**

The significant advantage offered by SSH over its predecessors is the use of encryption to ensure secure transfer of information between the host and the client. **Host** refers to the remote server you are trying to access, while the **client** is the computer you are using to access the host. There are three different encryption technologies used by SSH:

1. Symmetrical encryption
2. Asymmetrical encryption
3. Hashing.

**Symmetric Encryption**

Symmetric encryption is a form of encryption where a **secret key** is used for both encryption and decryption of a message by both the client and the host. Effectively, any one possessing the key can decrypt the message being transferred.

[](https://www.hostinger.com/tutorials/wp-content/uploads/sites/2/2017/07/symmetric-encryption-ssh-tutorial.jpg)

Symmetrical encryption is often called **shared key** or **shared secret** encryption. There is usually only one key that is used, or sometimes a pair keys where one key can easily be calculated using the other key.

Symmetric keys are used to encrypt the entire communication during a SSH Session. Both the client and the server derive the secret key using an agreed method, and the resultant key is never disclosed to any third party. The process of creating a symmetric key is carried out by a **key exchange algorithm***.* What makes this algorithm particularly secure is the fact that the key is never transmitted between the client and the host. Instead, the two computers share public pieces of data and then manipulate it to independently calculate the secret key. Even if another machine captures the publically shared data, it won’t be able to calculate the key because the key exchange algorithm is not known.

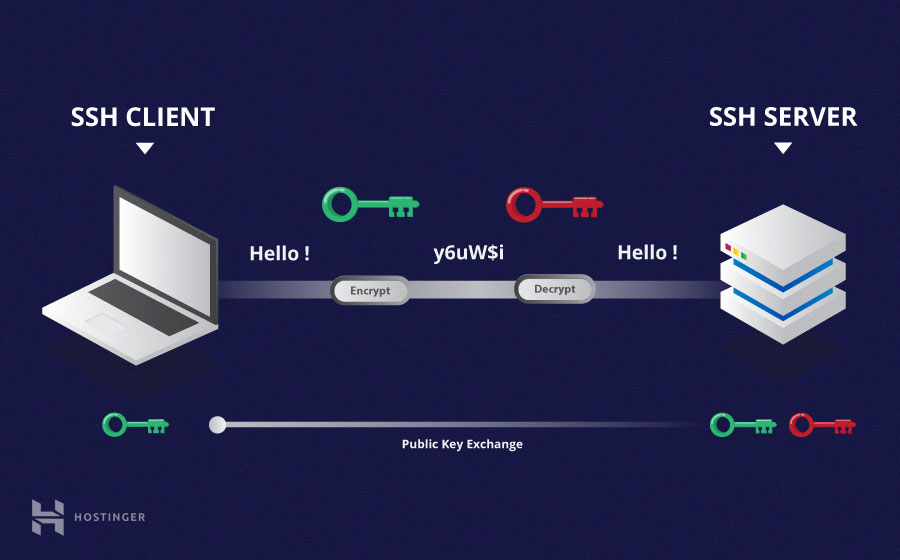
It must be noted, however, that the secret token is specific to each SSH session, and is generated prior to client authentication. Once the key has been generated, all packets moving between the two machines must be encrypted by the private key. This includes the password typed into the console by the user, so credentials are always protected from network packet sniffers.

A variety of symmetrical encryption ciphers exist, including, but not limited to, AES (Advanced Encryption Standard), CAST128, Blowfish etc. Before establishing a secured connection, the client and a host decide upon which cipher to use, by publishing a list of supported cyphers in order of preference. The most preferred cypher from the clients supported cyphers that is present on the host’s list is used as the bidirectional cypher.

For example, if two Ubuntu 14.04 LTS machines are communicating with each other over SSH, they will use **aes128-ctr** as their default cipher.

**Asymmetric Encryption**

Unlike symmetrical encryption, asymmetrical encryption uses two separate keys for encryption and decryption. These two keys are known as the **public key** and the **private key**. Together, both these keys form a **public-private key pair**.

[](https://www.hostinger.com/tutorials/wp-content/uploads/sites/2/2017/07/asymmetric-encryption.jpg)

The public key, as the name suggest is openly distributed and shared with all parties. While it is closely linked with the private key in terms of functionality, the private key cannot be mathematically computed from the public key. The relation between the two keys is highly complex: a message that is encrypted by a machine’s public key, can only be decrypted by the same machine’s private key. This one-way relation means that the public key cannot decrypt its own messages, nor can it decrypt anything encrypted by the private key.

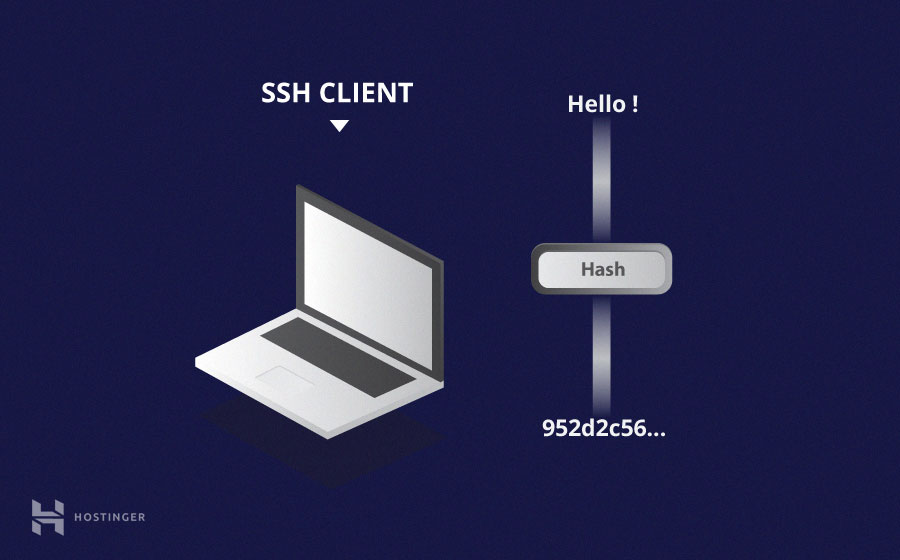
The private key must remain private i.e. for the connection to be secured, no third party must ever know it. The strength of the entire connection lies in the fact that the private key is never revealed, as it is the only component capable of decrypting messages that were encrypted using its own public key. Therefore, any party with the capability to decrypt publically signed messages must possess the corresponding private key.

Unlike the general perception, asymmetrical encryption is not used to encrypt the entire SSH session. Instead, it is only used during the key exchange algorithm of symmetric encryption. Before initiating a secured connection, both parties generate temporary public-private key pairs, and share their respective private keys to produce the shared secret key.

Once a secured symmetric communication has been established, the server uses the clients public key to generate and challenge and transmit it to the client for authentication. If the client can successfully decrypt the message, it means that it holds the private key required for the connection. The SSH session then begins.

**Hashing**

One-way hashing is another form of cryptography used in Secure Shell Connections. One-way-hash functions differ from the above two forms of encryption in the sense that they are never meant to be decrypted. They generate a unique value of a fixed length for each input that shows no clear trend which can exploited. This makes them practically impossible to reverse.

[](https://www.hostinger.com/tutorials/wp-content/uploads/sites/2/2017/07/ssh-tutorial-hash.jpg)

It is easy to generate a cryptographic hash from a given input, but impossible to generate the input from the hash. This means that if a client holds the correct input, they can generate the crypto-graphic hash and compare its value to verify whether they possess the correct input.

SSH uses hashes to verify the authenticity of messages. This is done using HMACs, or **H**ash-based **M**essage **A**uthentication **C**odes. This ensures that the command received is not tampered with in any way.

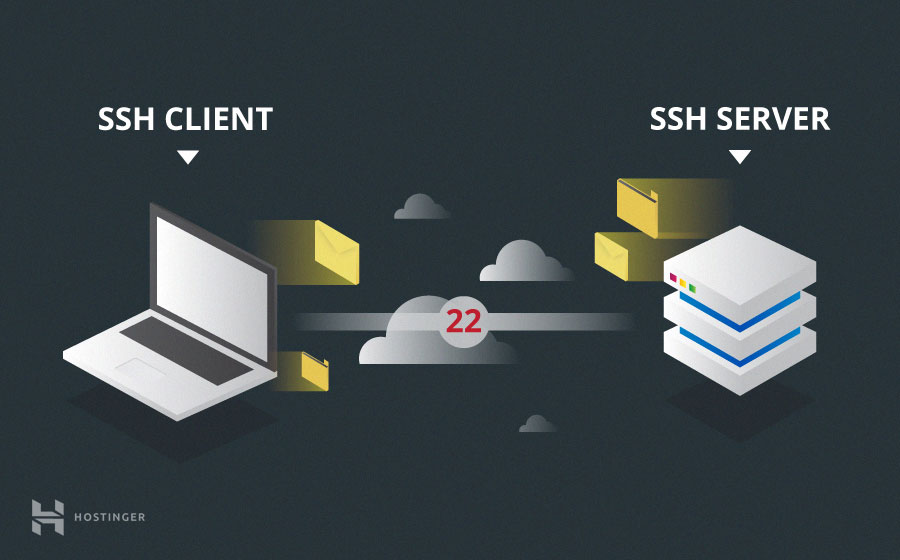
While the symmetrical encryption algorithm is being selected, a suitable message authentication algorithm is also selected. This works in a similar way to how the cipher is selected, as explained in the symmetric encryption section.

Each message that is transmitted must contain a MAC, which is calculated using the symmetric key, packet sequence number, and the message contents. It is sent outside the symmetrically encrypted data as the concluding section of the communication packet.

**How Does SSH Work with These Encryption Techniques**

The way SSH works is by making use of a client-server model to allow for authentication of two remote systems and encryption of the data that passes between them.

SSH operates on TCP port 22 by default (though this can be changed if needed). The host (server) listens on port 22 (or any other SSH assigned port) for incoming connections. It organizes the secure connection by authenticating the client and opening the correct shell environment if the verification is successful.

[](https://www.hostinger.com/tutorials/wp-content/uploads/sites/2/2017/07/ssh-client-and-server.jpg)

The client must begin the SSH connection by initiating the TCP handshake with the server, ensuring a secured symmetric connection, verifying whether the identity displayed by the server match previous records (typically recorded in an RSA key store file), and presenting the required user credentials to authenticate the connection.

There are two stages to establishing a connection: first both the systems must agree upon encryption standards to protect future communications, and second, the user must authenticate themselves. If the credentials match, then the user is granted access.

**Session Encryption Negotiation**

When a client tries to connect to the server via TCP, the server presents the encryption protocols and respective versions that it supports. If the client has a similar matching pair of protocol and version, an agreement is reached and the connection is started with the accepted protocol. The server also uses an asymmetric public key which the client can use to verify the authenticity of the host.

Once this is established, the two parties use what is known as a [Diffie-Hellman Key Exchange Algorithm](https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman_key_exchange) to create a symmetrical key. This algorithm allows both the client and the server to arrive at a shared encryption key which will be used henceforth to encrypt the entire communication session.

Here is how the algorithm works at a very basic level:

1. Both the client and the server agree on a very large prime number, which of course does not have any factor in common. This prime number value is also known as the **seed value**.
2. Next, the two parties agree on a common encryption mechanism to generate another set of values by manipulating the seed values in a specific algorithmic manner. These mechanisms, also known as encryption generators, perform large operations on the seed. An example of such a generator is AES (Advanced Encryption Standard).
3. Both the parties independently generate another prime number. This is used as a secret private key for the interaction.
4. This newly generated private key, with the shared number and encryption algorithm (e.g. AES), is used to compute a public key which is distributed to the other computer.
5. The parties then use their personal private key, the other machine’s shared public key and the original prime number to create a final shared key. This key is independently computed by both computers but will create the same encryption key on both sides.
6. Now that both sides have a shared key, they can symmetrically encrypt the entire SSH session. The same key can be used to encrypt and decrypt messages (read: section on symmetrical encryption).

Now that the secured symmetrically encrypted session has been established, the user must be authenticated.

**Authenticating the User**

The final stage before the user is granted access to the server is authenticating his/her credentials. For this, most SSH users use a password. The user is asked to enter the username, followed by the password. These credentials securely pass through the symmetrically encrypted tunnel, so there is no chance of them being captured by a third party.

Although passwords are encrypted, it is still not recommended to use passwords for secure connections. This is because many bots can simply brute force easy or default passwords and gain access to your account. Instead, the recommended alternative is [SSH Key Pairs](https://www.hostinger.com/tutorials/ssh/how-to-set-up-ssh-keys).

These are a set of asymmetric keys used to authenticate the user without the need of inputting any password.

**Conclusion**

Gaining an in-depth understanding of the underlying how SSH works can help users understand the security aspects of this technology. Most people consider this process to be extremely complex and un-understandable, but it is much simpler than most people think. If you’re wondering how long it takes for a computer to calculate a hash and authenticate a user, well, it happens in less than a second. In fact, the maximum amount of time is spent in transferring data across the Internet.

Hopefully, this SSH tutorial has helped you see the way different technologies can be clubbed together to create a robust system in which each mechanism has a very important role to play. Also, now you know why Telnet became a thing of the past as soon as SSH came up.