**What is OSPF authentication?**

OSPF authentication is a security measure that you can use to protect OSPF routing information from being modified or tampered with. When OSPF authentication is enabled, each router must authenticate itself before exchanging routing information with other routers. This authentication can be done using a simple clear-text password, or it can be done using more sophisticated methods such as MD5 cryptographic checksums.

**Which type of authentication is used by the OSPF protocol?**

The following authentication types are used and supported by OSPF:

Type 0 – Null authentication (Means no authentication, Default settings) Null authentication means that there is no authentication, which is the default on Cisco routers.

Type 1 – Clear-text passwords In this method of authentication, passwords are exchanged in clear text on the network

Type 2 – MD5 cryptographic checksums : The cryptographic method uses the open standard MD5 (Message Digest type 5) encryption.

**How does OSPF authentication work?**

The Open Shortest Path First (OSPF) is a link-state routing protocol for Internet Protocol (IP) networks. It uses the “shortest path first” (SPF) technique to calculate the best path through a network. OSPF is a widely used interior gateway protocol (IGP).

One of the key features of OSPF is that it supports authentication. This means each router can verify the identity of the other routers it communicates with. Two types of authentication can be used with OSPF: simple password authentication and MD5 authentication.

OSPF plain text authentication (Type 1) is the most basic form. With this method, each router has a clear-text password configured that it uses to authenticate with other routers. The problem with this authentication method is that the password is shown in the configuration and in OSPF messages. This is not a secure way to configure devices.

OSPF MD5 authentication (Type-2) is more secure than simple text authentication. This approach computes a hash value from the contents of an OSPF packet and a password using the MD5 algorithm (or key). This hash value is delivered alongside a key ID and a non-decreasing sequence number in the packet.

**Enabling OSPF Authentication:**

OSPF authentication can be enabling in two ways:

1) Per interface: Authentication is enabling per interface using the "ip ospf athentication" command.

2) Area authentication: Authentication for area can enable using "area authentication" command.

Configuring Authentication Key:

In either case password must be configure at interface using "ip ospf authentication-key" or "ip ospf message-digest-key" command

**Configuration Example:**

**A)Area based authentication Example:**

To enable OSPF MD5 authentication:

Router#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#interface fa0/0

Router(config-if)#ip ospf message-digest-key 1 md5 cisco@123

Router(config-if)#exit

Router(config)#router ospf 100

Router(config-router)#area 2 authentication message-digest

Router(config-router)#exit

To enable clear text authentication

Router#configure terminal

Enter configuration commands, one per line. End with CNTL/Z.

Router(config)#interface f0/0

Router(config-if)#ip ospf authentication-key cisco@123

Router(config-if)#exit

Router(config)#router ospf 100

Router(config-router)#area 2 authentication

Router(config-router)#exit

**Interface based authentication Example:**

To enable OSPF MD5 authentication:

Router(config)#int fa0/0

Router(config-if)#ip ospf authentication message-digest

Router(config-if)#ip ospf message-digest-key 1 md5 cisco

Router(config-if)#exit

Router(config)#

To enable clear text authentication

Router(config)#int fa0/0

Router(config-if)#ip ospf authentication

Router(config-if)#ip ospf authentication-key cisco

Router(config-if)#exit

Router(config)#

# How NTP works for authentication

 It's important for all network-attached computers to have the same time for multiple reasons. All log messages, file accesses and logins are marked with a timestamp. Computer real-time hardware clocks, or RTCs, are built into every computer, but are only used to get the time when the computer boots up. The reason for this is that real-time clocks are inaccurate, and can lose five seconds per month, or more. Once the computer has booted up, it uses the TSC, or time stamp counter, built into the CPU, to maintain time, which is accurate, but not persistent between reboots. Network time protocol allows our computers to access a time server on the network, and with that information, reset its local clock. These time servers are synchronized with atomic clocks around the world, an NTP server can synchronize with another NTP server. Each level away from the atomic clock is called a stratum. Stratum zero are the atomic clocks themselves, stratum one is the NTP server that synchronizes to it, stratum two is an NTP server that synchronizes with a stratum one server, and so on. At stratum 16, the time is considered unsynchronized. Network time synchronization is important for authentication. When a user tries authenticating, the system records the timestamp. It then measures the amount of time used to authenticate. If the time was too great, authentication fails. For remote authentication sources, things are more complex. Timestamps are used in authentication protocols, such as kerberos, to prevent replay attacks, where an attacker can reuse an authentication token. Most implementations of kerberos authentication do this. If the token is more than five minutes old authentication will fail, so it's important to have the correct time on both a client and the server. Newer versions of MIT kerberos calculate the offsets between the clocks during authentication, so the time difference is not as important. Other authentication methods, such as Google Authenticator and RSA SecurID also require synchronized clocks, and have a much tighter threshold of 30 to 60 seconds. Without synchronized time between a client's server, authentication fails. NTP has roles, depending on the function a system provides. There are primary and secondary NTP servers, NTP peers, and NTP clients. A primary NTP server gets its time from a source such as an atomic clock, it then either provides this information to a secondary server or clients. A secondary NTP server gets its time from a primary server, and provides this information to clients either directly, or by a broadcast. An NTP peer is a host that provides time to an NTP server, and gets time from that server. Peered hosts synchronize their time with each other. Peered hosts usually work at the same stratum level. Lastly, an NTP client only retrieves time information from an NTP primary or secondary server. A client can receive this information either by pulling an NTP server, or listening for NTP broadcasts.

## What is the Secure Shell (SSH) protocol?

The Secure Shell (SSH) protocol is a method for securely sending commands to a computer over an unsecured network. SSH uses cryptography to authenticate and encrypt connections between devices. SSH also allows for [tunneling](https://www.cloudflare.com/learning/network-layer/what-is-tunneling/), or port forwarding, which is when data [packets](https://www.cloudflare.com/learning/network-layer/what-is-a-packet/) are able to cross networks that they would not otherwise be able to cross. SSH is often used for controlling servers remotely, for managing infrastructure, and for transferring files.

## What does SSH do?

**Remote encrypted connections:** SSH sets up a connection between a user's device and a faraway machine, often a server. It uses encryption to scramble the data that traverses the connection. An intercepting party would only find something like static — random data that means nothing unless it is decrypted. (SSH uses encryption methods that make decryption prohibitively difficult for outsiders.)

**The ability to tunnel:** In networking, tunneling is a method for moving packets across a network using a protocol or path they would not ordinarily be able to use. Tunneling works by wrapping data packets\* with additional information — called headers — to change their destination. SSH tunnels use a technique called port forwarding to send packets from one machine to another. Port forwarding is explained in more detail below.

## How does SSH work?

#### TCP/IP

SSH runs on top of the [TCP/IP](https://www.cloudflare.com/learning/ddos/glossary/tcp-ip/) protocol suite. TCP/IP pairs those two protocols in order to format, route, and deliver packets. IP indicates, among other information, which [IP address](https://www.cloudflare.com/learning/dns/glossary/what-is-my-ip-address/) a packet should go to (think of a mailing address), while TCP indicates which port a packet should go to at each IP address (think of the floor of a building or an apartment number).

#### Public key cryptography

SSH is "secure" because it incorporates encryption and [authentication](https://www.cloudflare.com/learning/access-management/what-is-authentication/) via a process called public key cryptography. [Public key cryptography](https://www.cloudflare.com/learning/ssl/how-does-public-key-encryption-work/) is a way to encrypt data, or sign data, with two different keys. One of the keys, the public key, is available for anyone to use. The other key, the private key, is kept secret by its owner. Because the two keys correspond to each other, establishing the key owner's identity requires possession of the private key that goes with the public key.

These "asymmetric" keys also make it possible for the two sides of the connection to negotiate identical, shared [symmetric keys](https://www.cloudflare.com/learning/ssl/what-is-asymmetric-encryption/) for further encryption over the channel. Once this negotiation is complete, the two sides use the symmetric keys to encrypt the data they exchange.

In an SSH connection, both sides have a public/private key pair, and each side authenticates the other using these keys. This differentiates SSH from [HTTPS](https://www.cloudflare.com/learning/ssl/what-is-https/), which in most implementations only verifies the identity of the web server in a client-server connection.

#### Authentication

While public key cryptography authenticates the connected devices in SSH, a properly secured computer will still require authentication from the person using SSH. Often this takes the form of entering a username and password.

Once authentication is complete, the person can execute commands on the remote machine as if they were doing so on their own local machine.

#### SSH tunneling, or 'port forwarding'

Port forwarding is like forwarding a message between two people. Bob may send a message to Alice, who in turn passes it to Dave. Similarly, port forwarding sends data packets directed at an IP address and port on one machine to an IP address and port on a different machine.

For example, imagine an administrator wants to make a change on a server inside a private network they manage, and they want to do so from a remote location. However, for security reasons, that server only receives data packets from other computers within the private network. The administrator could instead connect to a second server within the network — one that is open to receiving Internet traffic — and then use SSH port forwarding to connect to the first server. From the first server's perspective, the administrator's data packets are coming from inside the private network.