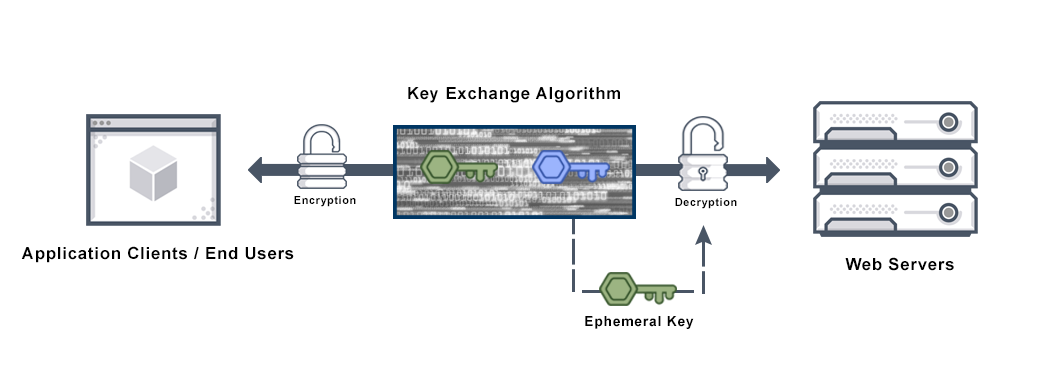
**Perfect Forward Secrecy Definition**

**Perfect Forward Secrecy (PFS),** also called forward secrecy (FS), refers to an encryption system that changes the keys used to encrypt and decrypt information frequently and automatically. This ongoing process ensures that even if the most recent key is hacked, a minimal amount of sensitive data is exposed.

Web pages, calling apps, and messaging apps all use encryption tools with perfect forward secrecy that switch their keys as often as each call or message in a conversation, or every reload of an encrypted web page. This way, the loss or theft of one decryption key does not compromise any additional sensitive information—including additional keys.

Determine whether forward secrecy is present by inspecting the decrypted, plain-text version of the data exchange from the key agreement phase of session initiation. An application or website’s encryption system provides perfect forward secrecy if it does not reveal the encryption key throughout the session.



**FAQs**

**What is Perfect Forward Secrecy?**

Perfect forward secrecy helps protect session keys against being compromised even when the server’s private key may be vulnerable. A feature of specific key agreement protocols, an encryption system with forward secrecy generates a unique session key for every user initiated session. In this way, should any single session key be compromised, the rest of the data on the system remains protected. Only the data guarded by the compromised key is vulnerable.

Before perfect forward secrecy, the Heartbleed bug affected OpenSSL, one of the common SSL/TLS protocols. With forward secrecy in place, even man-in-the-middle attacks and similar attempts fail to retrieve and decrypt sessions and communications despite compromise of passwords or secret long-term keys.

**Compare Backwards vs Forwards Secrecy**

Perfect forward secrecy guards against future compromises of past sessions, which could cause the loss of sensitive data such as passwords or additional secret keys.

Backward secrecy helps “self-heal” compromises of past sessions and the loss of sensitive data from them. This is confusing because both do focus on data from past sessions, but forward secrecy is preventative while backward secrecy is mitigating. For example, the Signal protocol uses the self-healing Double Ratchet Algorithm to achieve backward secrecy.

**How Does Perfect Forward Secrecy Work?**

Encryption perfect forward secrecy enables entirely private, short-term key exchanges between a client and the server.

Normally, web servers secure communication sessions with special encryption keys. Whenever a client wants to talk to the server, the client generates a pre-master secret and uses the server’s special key to encrypt it. Both users then continue the rest of the chat, encrypting it with this pre-master secret.

Only people who know the original key of the server can decrypt what client and server discuss. The network team, for example, supports the server and must monitor communications to assist in its task of tracking down bugs.

Without perfect forward secrecy, an attacker can spy on the server’s communications unobserved. This is because the server uses the same key to encrypt each pre-master secret with each client.

If the server secures communications with perfect forward secrecy, every time a new client starts a conversation with the server, the two generate a unique pre-master secret that is totally private. It is also ephemeral, and only lasts for that one communication. The client never sees the long-term key, and a hacker is limited to only what is shared during that one conversation.

Consider this hypothetical example of a basic instant messaging protocol using perfect forward secrecy:

* **Step One:** X and Y each generate a pair of asymmetric, long-term, public keys and private keys. They use an already-authenticated channel to verify the public-key fingerprints, or verify them in person. The verification process establishes to a high degree of certainty that the public key’s claimed owner is also its actual owner.
* **Step Two:** X and Y securely agree on an ephemeral key for the session using a key exchange algorithm such as Diffie-Hellman. They authenticate each other with the keys from Step One during this process.
* **Step Three:** X uses the session key negotiated in Step Two to encrypt a message with a symmetric cipher and sends that encrypted version to Y.
* **Step Four:** Y decrypts the message with the key from Step Two.

Step One never repeats. Instead, the process repeats starting from Step Two for each new message sent. Depending on the conversation, X and Y’s roles as sender or recipient may switch. It is this generation of new session keys for each message that achieves forward secrecy.

Even if Step Two is compromised at some point, that key is only good for one message. A compromise of Step One would also leave messages intact—although it might enable an attacker to impersonate X or Y moving forward, leaving future messages vulnerable.

**Benefits of Perfect Forward Secrecy**

There are many benefits to perfect forward secrecy. Brute force attacks can eventually penetrate even very secure encryption, given enough time and computing power to try combinations of security keys. Without forward secrecy, encryption keys are used for sessions—entire batches of transactions.

Brute force hacking demands extensive time and resources, but that level of return of sensitive data makes it worthwhile. Perfect forward secrecy guarantees brute force attacks won’t be as worthwhile.

Generating a unique session key for each transaction limits hackers to obtaining data from one exchange per successful attack. A server protected by perfect forward secrecy is simply a less appealing target for a hacker, because it demands more effort and time. There’s also no future value in such an attack, because the server with PFS generates a new set of Diffie-Hellman parameters per session.

**Does Blockchain Have Forward Secrecy?**

Perfect forward secrecy protects future compromises of the passwords or secret keys from past sessions. With forward secrecy in place, previously recorded and encrypted sessions and communications cannot be retrieved and decrypted by an attacker who compromises long-term secrets keys in the future.

This is critical for a blockchain use case. A leaked key has the potential to compromise a significant amount of assets in a blockchain scenario since all data is stored forever.

**VPN Perfect Forward Secrecy**

VPN perfect forward secrecy simply refers to the use of perfect forward secrecy by VPNs. PFS makes VPN connections more secure, though it can reduce speed slightly in some cases.

**Perfect Forward Secrecy Protocols**

Several major protocol implementations provide perfect forward secrecy, at least as an optional feature, including SSH, IPsec (RFC 2412), and the IM library and cryptography protocol, Off-the-Record Messaging.

In Transport Layer Security (TLS) 1.3, the ephemeral Diffie–Hellman key exchange supports perfect forward secrecy. OpenSSL provides forward secrecy with elliptic curve Diffie–Hellman key exchange.

The Signal Protocol supports forward security with the Double Ratchet Algorithm. However, WPA does not provide perfect forward secrecy.

**How to Enable Perfect Forward Secrecy**

Perfect forward secrecy works on sites that use either SSL or TLS sessions. Both cryptographic protocols allow secure connections to be created, but neither determines the encryption cipher to be used or mandates the actual key exchange.

Instead, to enable perfect forward secrecy, the user and server machines must agree upon the encryption type. Therefore, when configuring forward secrecy, set your servers up to make compliant cipher suites available:

* Ephemeral Elliptic Curve Diffie-Hellman (ECDHE)
* Ephemeral Diffie-Hellman (DHE)

The key exchange must be ephemeral, meaning the server and client will generate a unique set of Diffie-Hellman parameters and use the keys just once per session. The exchange-related encryption is deleted from the server after the transaction ends, which ensures that any given session key is almost useless to hackers.

If possible, select Elliptic Curve DHE suites. These are faster than the standard DHE counterparts.

To determine whether perfect forward secrecy is enabled, refer to the security details of a site. If it is using “ECDHE” or “DHE” then it is currently using forward secrecy.

Most modern servers are already configured for perfect forward secrecy, but if your server is not, complete the process in four steps:

* Locate the SSL protocol configuration.
* Add the protocol to your configuration.
* Set the SSL cipher. Ensure you enforce the ordering of your ciphers by using ‘SSLHonorCipherOrder on’ in Apache and ‘ssl\\_prefer\\_server\\_ciphers on;’ in nginx.
* Restart.

Importantly, it is easy to configure perfect forward secrecy incorrectly. A common error is to simply enable support for DHE or ECDHE without actually enforcing the ordering of the ciphers. Simply enabling them doesn’t mean the server is using perfect forward secrecy.

In addition, prioritize perfect forward secrecy over other security methods to ensure it works properly. In some cases you may need to disable other types of security so weaker forms of encryption don’t take priority in error, allowing for FREAK attacks and other SSL/TLS vulnerabilities.

It is also important to disable long duration session tickets or session IDs. These hold onto session information on the user’s side for extended periods of time—sometimes until the system is rebooted.

**When to Use Perfect Forward Secrecy**

In November 2014, Sony Pictures experienced a major security breach in which hackers stole their servers’ private keys and SSH keys. This makes the case for perfect forward secrecy in that the attackers could use these stolen keys to decrypt confidential data Sony may have collected in the past.

Any current sites should support PFS. Perfect forward secrecy is valuable against attackers who may be able to achieve READ access, but not WRITE access. In other words, an attacker who can undertake cryptanalysis of the underlying ciphers being used and modify the way the session key generator functions may be responsible for failed forward secrecy. For example, large quantum computers are up to breaking these ciphers in a reasonable amount of time.

However, in most cases perfect forward secrecy separates the confidentiality of past conversations and any compromise of a long-term secret key successfully. This begs the question: why don’t all websites support PFS? There are several reasons organizations have for failing to implement perfect forward secrecy.

Lack of infrastructure support and lack of browser support are among the reasons. PFS demands specific combinations of SSL settings, particularly the ephemeral Diffie-Hellman cipher suite, which it deploys for the key exchange. Most current web servers and OpenSSL support PFS and the Diffie-Hellman ciphers. However, servers and infrastructure that don’t can still benefit from PFS thanks to newer, software-based load balancers.

Performance impact of perfect forward secrecy is another issue, in that enabling PFS can reduce the SSL performance of sites by [more than 90 percent](https://blog.avinetworks.com/perfect-forward-secrecy-with-performance-you-need). PFS is also approximately three to four times more computationally expensive for traditional RSA keys. However, PFS adds little to no overhead for newer [elliptic curve cryptography](https://avinetworks.com/glossary/elliptic-curve-cryptography/) certificates, and so long as your SSL decryption infrastructure has capacity or can scale capacity on demand, performance won’t be a problem even if you are still using RSA 2k certificates.

Implementation complexity has been a barrier to achieving perfect forward secrecy in the past, but implementation is easier than ever with modern load balancers and other packaged solutions. To implement PFS ideally, use an ECC certificate to negotiate fast SSL/TLS encryption with PFS, and simultaneously leverage an RSA certificate as a backup for compatibility with older browsers.

Waiting for a compelling event in the form of a data breach or attack is a mistake.

**Does Avi Networks Support Perfect Forward Secrecy?**

Why don’t all websites support perfect forward secrecy? Because many [load balancers](https://avinetworks.com/software-load-balancer/) are not equipped to drive PFS efficiently. In some cases, enabling forward secrecy can reduce SSL performance on a site by more than 90%.

The Avi Networks innovation is unlimited SSL performance scaling and massively improved PFS performance. Learn more about the [Avi Networks platform](https://avinetworks.com/why-avi/application-delivery-platform/) here.