**Enabling Perfect Forward Secrecy**

To encrypt communications between you and your end users, you purchase a SSL Certificate, install it on your server, and then configure your website to use the certificate to protect these communications. The SSL connection begins when the end user’s browser reaches out to shake hands with your website.

During this handshake, information regarding the ability of the browser and server are exchanged, validation occurs, and a session key that meets both the browser’s and server’s criteria is created. Once the session key is created, the rest of the conversation between the end user and your site is encrypted and thus secured. Historically, the most common method for negotiating the session key involved the RSA public-key cryptosystem. The RSA approach uses the server’s public key to protect the session key parameters created by the browser once they are sent the server. The server is able to decrypt this handshake with its corresponding private key.

When you use the RSA key exchange mechanism, it creates a link between the server’s key pair and the session key created for each unique secure session. Thus, if an attacker is ever able to get hold of the server’s private key, they can decrypt your SSL session and any saved SSL sessions.

In contrast, when you enable Perfect Forward Secrecy (PFS), there is no link between your server’s private key and each session key. If an attacker ever gets access to your server’s private key, the attacker cannot use the private key to decrypt any of your archived sessions, which is why it is called “Perfect Forward Secrecy”.

To see if your server supports Perfect Forward Secrecy, use [Discovery](https://docs.digicert.com/certificate-tools/discovery-user-guide/) to test it.

**Deploying Perfect Forward Secrecy**

Instead of using the RSA method for exchanging session keys, you should use the Elliptic Curve Diffie-Hellman (ECDHE) key exchange. Note that you can still use the RSA public-key cryptosystem as the encryption algorithm, just not as the key exchange algorithm. ECDHE is much faster than ordinary DH (Diffie-Hellman), but both create session keys that only the entities involved in the SSL connection can access. Because the session keys are not linked to the server’s key pair, the server’s private key alone cannot be used to decrypt any SSL session.

To enable Perfect Forward Secrecy, you must do the following:

1. Reorder your cipher suites to place the ECDHE (Elliptic Curve Diffie-Hellman) suites at the top of list, followed by the DHE (Diffie-Hellman) suites.
2. Configure servers to enable other non-DH-key-exchange cipher suites from the list of cipher suites offered by the SSL Client.

**Configuring Perfect Forward Secrecy**

* [Configuring Apache for Perfect Forward Secrecy](https://www.digicert.com/kb/ssl-support/ssl-enabling-perfect-forward-secrecy.htm#apache_forward_secrecy)
* [Configuring Nginx for Perfect Forward Secrecy](https://www.digicert.com/kb/ssl-support/ssl-enabling-perfect-forward-secrecy.htm#nginx_forward_secrecy)

Configuring Apache for Forward Secrecy

Before you configure your Apache server for Forward Secrecy, your web server and SSL/TLS library should support Elliptic Curve cryptography (ECC).

Minimum Required Versions

* OpenSSL 1.0.1c+
* Apache 2.4x

**Note:**    Because of the Heartbleed bug and OpenSSL vulnerabilities, you should update to the most recent versions (i.e. OpenSSL version 1.0.1h).

How to Configure Apache for Forward Secrecy

To configure Apache for Forward Secrecy, you configure the server to actively choose cipher suites and then activate the right OpenSSL cipher suite configuration string.

1. Locate your SSL Protocol Configuration on your Apache server.

For example,

* 1. Type the following command:

**grep -i -r "SSLEngine" /etc/apache**

In this example, **/etc/apache** is the base directory for the Apache installation.

* 1. The command will out put the available Virtual Hosts.
  2. Open the Virtual Host for which you are enabling Forward Secrecy.

1. Add the following lines to your configuration:
   1. **SSLProtocol all -SSLv2 -SSLv3**
   2. **SSLHonorCipherOrder on**
   3. For **SSLCipherSuite**, use one of the following configurations:
      * **Configure with RC4**

Use this configuration if you have a preference for GCM (Galois Counter Mode) suites (these suites are resistant to timing attacks) and RC4 (RC4 is resistant to BEAST). To improve performance, use the faster ECDHE suites whenever possible.

**SSLCipherSuite "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA256 EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP !DSS"**

* + - **Configure without RC4**

Use this configuration if you have a preference for GCM (Galois Counter Mode) suites (these suites are resistant to timing attacks) and you prefer not to use RC4. To improve performance, use the faster ECDHE suites whenever possible.

**SSLCipherSuite "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA256 EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP !DSS !RC4"**

* + - **Configure with RC4 as a last resort to support wide range and older browsers**

Use this configuration if you have a preference for GCM (Galois Counter Mode) suites (these suites are resistant to timing attacks) and you want to use RC4 only as a last resort in order to support a wide range of browsers and/or older browsers. To improve performance, use the faster ECDHE suites whenever possible.

**SSLCipherSuite "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA256 EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP !DSS +RC4 RC4"**

1. Restart Apache.

For example, type the following command:

**apachectl -k restart**

1. To verify that you have enabled Forward Secrecy, use [Discovery](https://docs.digicert.com/certificate-tools/discovery-user-guide/) to test your configuration.
2. You have successfully configured Apache for Forward Secrecy.

Configuring Nginx for Forward Secrecy

Before you configure your Nginx server for Forward Secrecy, your web server and SSL/TLS library should support Elliptic Curve cryptography (ECC).

Minimum Required Versions

* OpenSSL 1.0.1c+
* Nginx 1.0.6+ and 1.1.0+

**Note:**    Because of the Heartbleed bug and OpenSSL vulnerabilities, you should update to the most recent versions (i.e. OpenSSL version 1.0.1h).

How to Configure Nginx for Forward Secrecy

To configure Nginx for Forward Secrecy, you configure the server to actively choose cipher suites and then activate the right OpenSSL cipher suite configuration string.

1. Locate your SSL Protocol Configuration on your Nginx server.

For example,

* 1. Type the following command:

**grep -r ssl\_protocol /etc/nginx**

In this example, **/etc/nginx** is the base directory for the Nginx installation.

* 1. The command will out put the available Server Blocks.
  2. Open the Server Block for which you are enabling Forward Secrecy.

1. Add the following lines to your configuration:
   1. **ssl\_protocols TLSv1.2 TLSv1.1 TLSv1;**
   2. **ssl\_prefer\_server\_ciphers on;**
   3. For **ssl\_ciphers**, use one of the following configurations:
      * **Configure with RC4**

Use this configuration if you have a preference for GCM (Galois Counter Mode) suites (these suites are resistant to timing attacks) and RC4 (RC4 is resistant to BEAST). To improve performance, use the faster ECDHE suites whenever possible.

**ssl\_ciphers "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA256 EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP !DSS";**

* + - **Configure without RC4**

Use this configuration if you have a preference for GCM (Galois Counter Mode) suites (these suites are resistant to timing attacks) and you prefer not to use RC4. To improve performance, use the faster ECDHE suites whenever possible.

**ssl\_ciphers "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA256 EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP !DSS !RC4";**

* + - **Configure with RC4 as a last resort to support wide range and older browsers**

Use this configuration if you have a preference for GCM (Galois Counter Mode) suites (these suites are resistant to timing attacks) and you want to use RC4 only as a last resort in order to support a wide range of browsers and/or older browsers. To improve performance, use the faster ECDHE suites whenever possible.

**ssl\_ciphers "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384 EECDH+ECDSA+SHA256 EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP !DSS +RC4 RC4";**

1. Restart Nginx.

For example, type the following command:

**sudo service nginx restart**

1. To verify that you have enabled Forward Secrecy, use [Discovery](https://docs.digicert.com/certificate-tools/discovery-user-guide/) to test your configuration.
2. You have successfully configured Nginx for Forward Secrecy.