Modern SSL/TLS Best Practices for Fast, Secure Websites

What every IT professional should know about SSL security

1 888 99 FLARE | enterprise@cloudflare.com | www.cloudflare.com

HTTPS Isn't What It Used To Be

It's faster, more secure, and used by more websites than ever before. As a busy web developer or network architect, it can be hard to keep up with the latest advancements in Internet technologies. In this whitepaper, we'll try to make it a little bit easier by exploring SSL best practices for securing modern websites:

- Support HSTS headers
- Use SHA-2 certificate signatures
- Enable forward secrecy with Ephemeral Diffie-Hellman handshakes
- Upgrade to TLS 1.2

But, modern SSL doesn't just better secure your site—it's designed for performance, too. We'll also talk about some common SSL performance optimizations that can dramatically speed up your website:

- Support OCSP stapling
- Leverage elliptic curve cryptography
- Allow TLS session resumption
- Enable HTTP/2 or SPDY

A Brief SSL Refresher

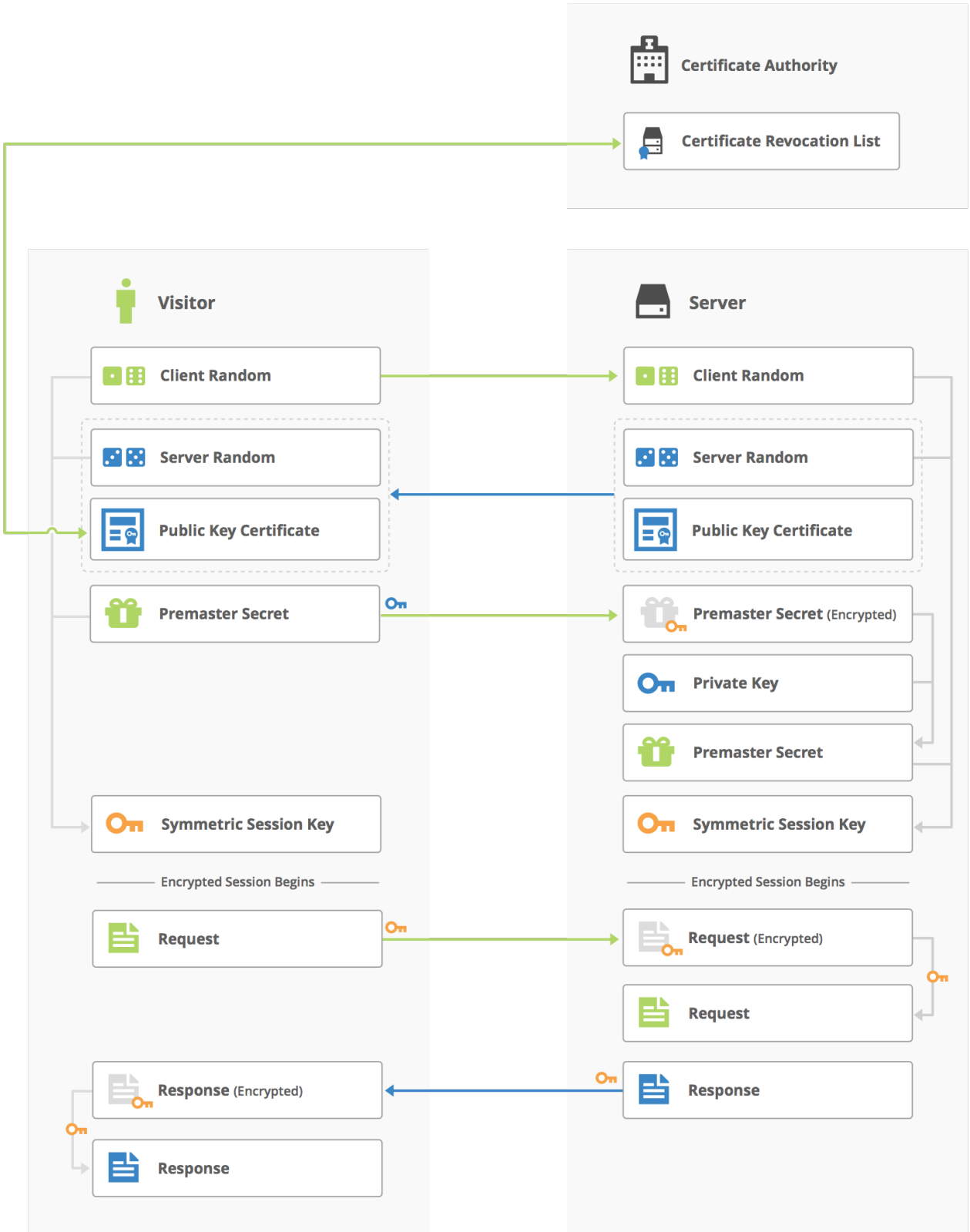
Before we jump into best practices, let's start with a little refresher on SSL and TLS. First up: terminology. Technically, Secure Sockets Layer (SSL) has been superseded by the Transport Layer Security (TLS) protocol, but most people call both of them "SSL." Unless you see a version number after it, that's the convention we'll be using.

Second, it's important to have a basic understanding of the SSL/TLS handshake before diving into this whitepaper. This handshake occurs every time a browser connects to a server over HTTPS, and it's a way for the client and server to agree on what kind of encryption they'll use for the session. There are a few different types of TLS handshakes, but the RSA handshake diagram on the next page shows the relevant parts.

A TLS handshake begins with a browser sending a large random number and a list of supported cipher suites to your web server. Your web server selects its preferred cipher suite from the list, which determines the type of handshake and encryption mechanism to use for this TLS session. It sends that, along with another large random number and your server's public SSL certificate back to the browser. The browser needs to know if your SSL certificate has been revoked, so it checks your Certificate Authority's Certificate Revocation List (unless you support OCSP Stapling).

If the certificate is still valid, the browser creates a "premaster secret" that will be used to create a symmetric session key. It encrypts the premaster secret with your server's public key (which is contained its SSL certificate) and sends it back to the server. After the server decrypts the premaster secret with its private key, both the browser and server can derive the symmetric key that will be used to encrypt the rest of the TLS session.

Everything we'll be talking about in this whitepaper either makes one of these steps more secure or faster.



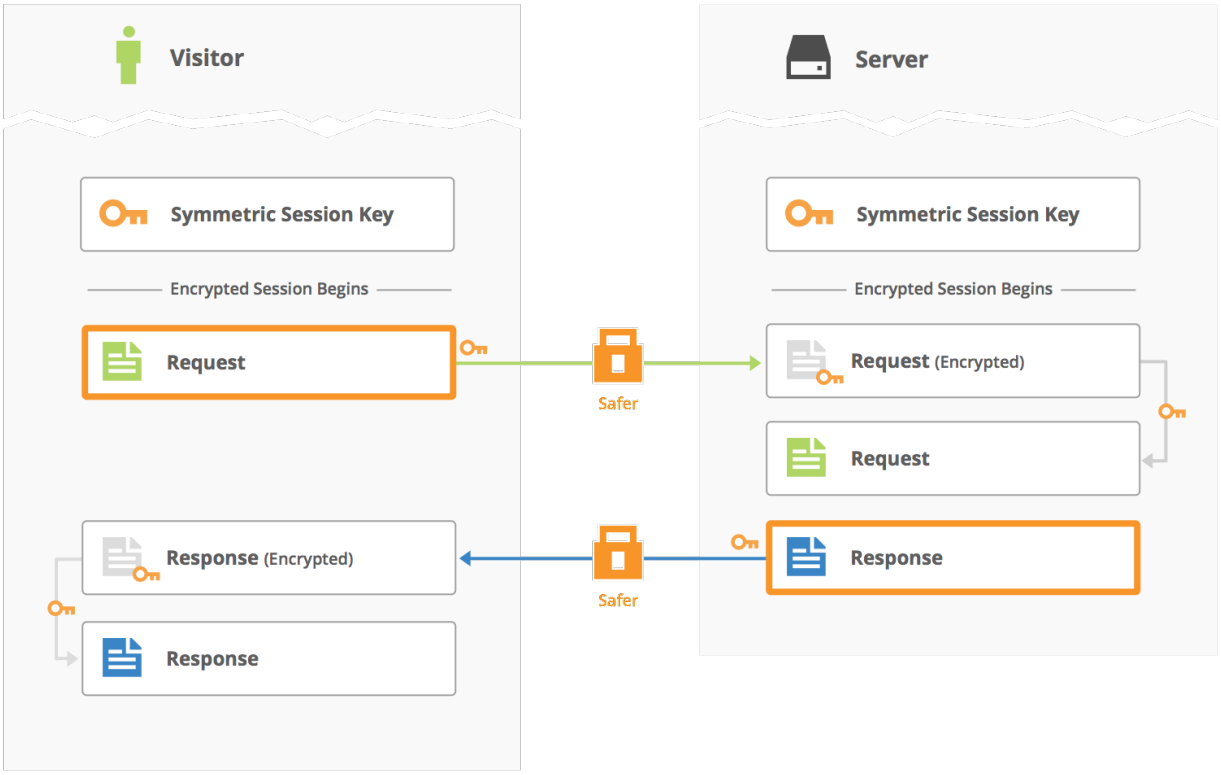
The RSA TLS handshake

Leverage Modern SSL Security Features

There's a lot of steps in the TLS handshake, which means there's a lot of opportunity for you to improve the security of your site. We'll start by talking about HSTS to force HTTPS connections, then we'll explain the current state of SHA-1 versus SHA-2, how to protect your users' session data in the future, and the importance of upgrading to the latest version of TLS.

Support HSTS Headers

Supporting the HTTP Strict Transport Security (HSTS) protocol is one of the easiest ways to better secure your website, API, or mobile application. HSTS is an extension to the HTTP protocol that forces clients to use secure connections for every request to your web server. By supplying a Strict-Transport-Security header, your web server can tell browsers that they should only connect to your website over HTTPS for a specified time frame.



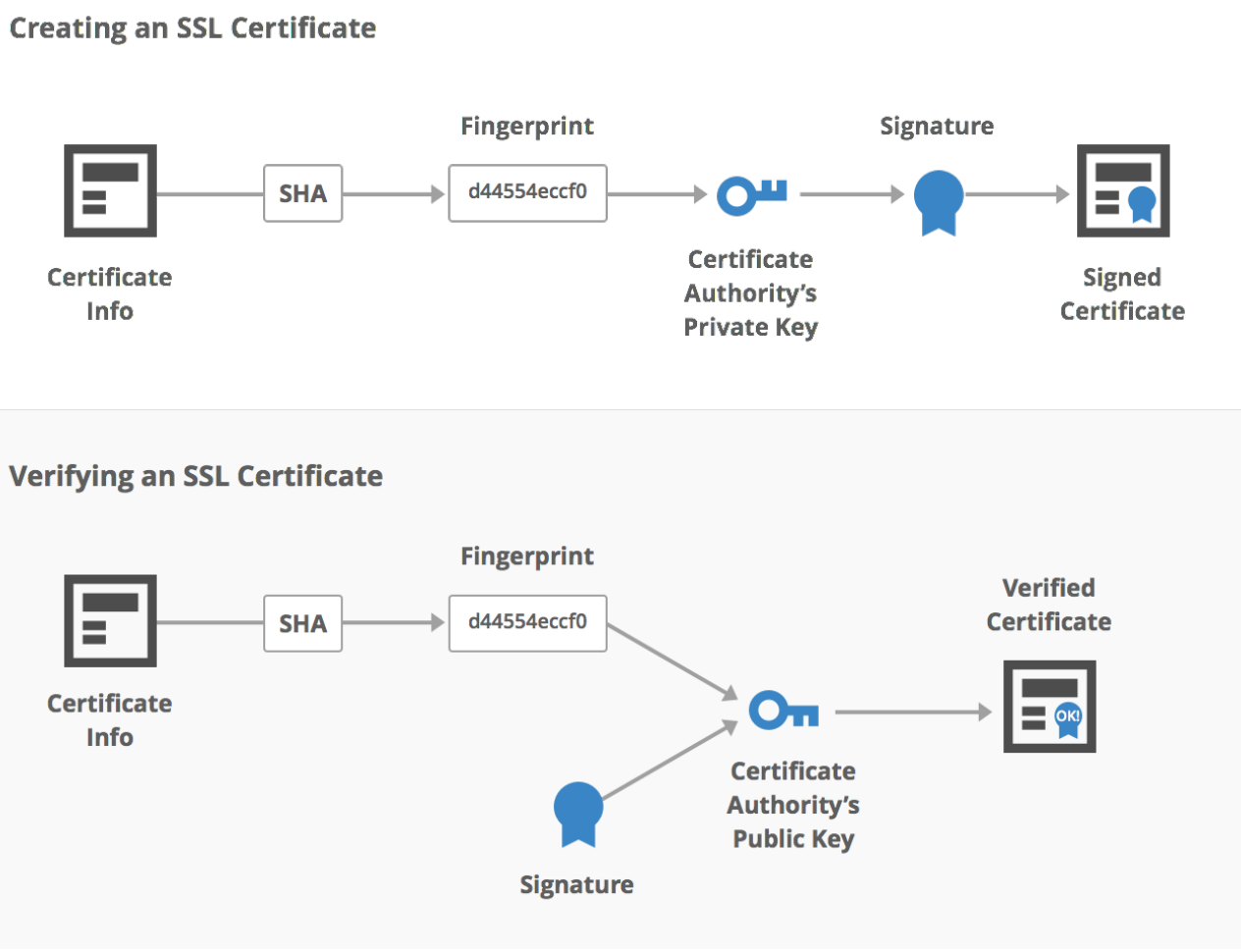
More secure requests and responses with HSTS

Conformant browsers will then automatically turn all http:// requests to https:// before sending them to your server. HSTS also tells browsers to show an error page when the security of the connection is in question (for example, when the server's TLS certificate is not trusted). And unlike errors that users instinctively dismiss, these errors cannot be bypassed.

This link rewriting protects against certain types of downgrade attacks like "SSL-stripping" where a man-in-the-middle silently changes an HTTPS request into an HTTP request so they can snoop on users' traffic.

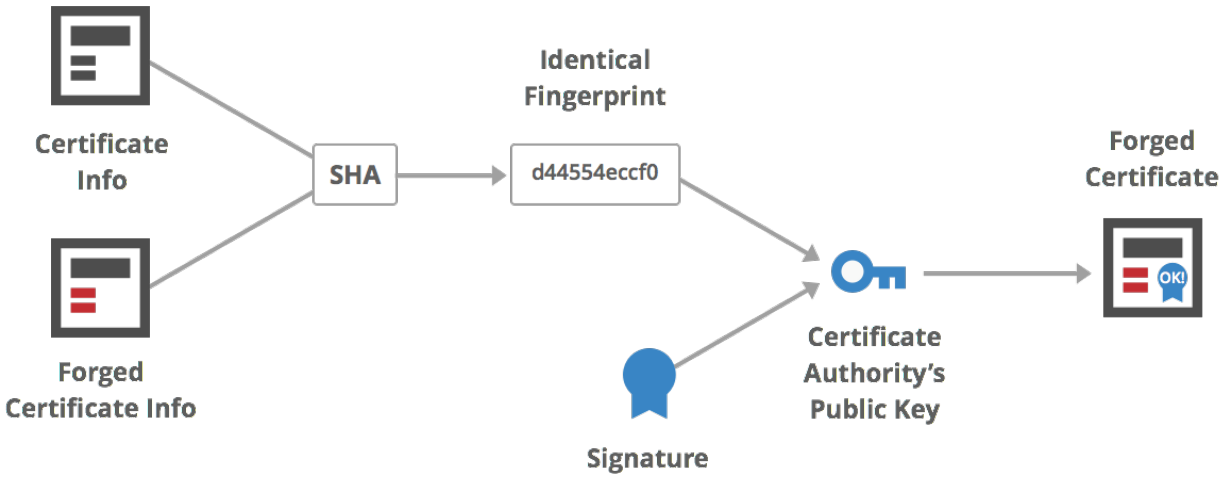
Support SHA-2 Certificate Signatures

SHA-2 is the next evolution of the Secure Hash Algorithm (SHA), replacing the potentially insecure SHA-1 algorithm. Hash algorithms are one-way functions that create a unique digital fingerprint of a message, and they've been a crucial component of Internet security since its inception.

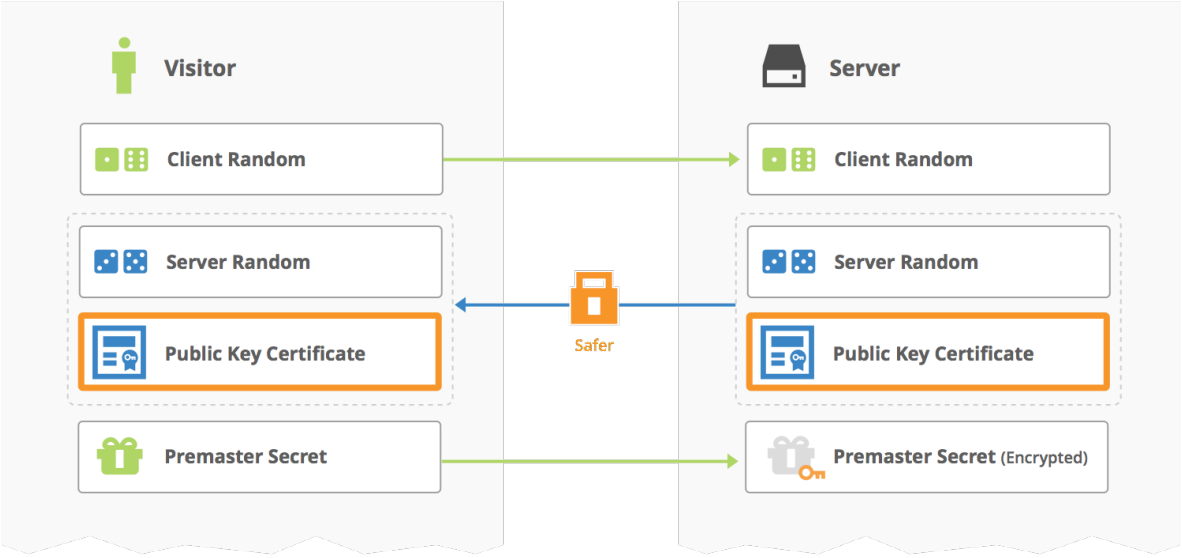


When a Certificate Authority (CA) issues a TLS certificate for your website, it takes all the information in that certificate (the domain, validity period, public key, serial number, etc.), hashes it into a digital fingerprint, and creates a signature using that fingerprint and their private signing key. Before a browser can trust your server's certificate, it needs to hash the certificate information and verify that the fingerprint matches your certificate's signature using the CA's public signing key.

If an attacker can create the same digital fingerprint using different certificate information, they can produce a forged certificate that still validates against the CA's signature. Then, they could serve up that forged certificate as a man-in-the-middle, and an end user wouldn't be able to tell they're actually sending all their sensitive information to some random person on the Internet instead of your "secure" web server.



All it really takes to produce this kind of hash collision is a lot of computing power. As computers have gotten faster and cheaper, it's become increasingly feasible for an attacker to forge a TLS certificate that uses SHA-1 for its signature. The solution is SHA-2, which makes it much harder to create a collision.



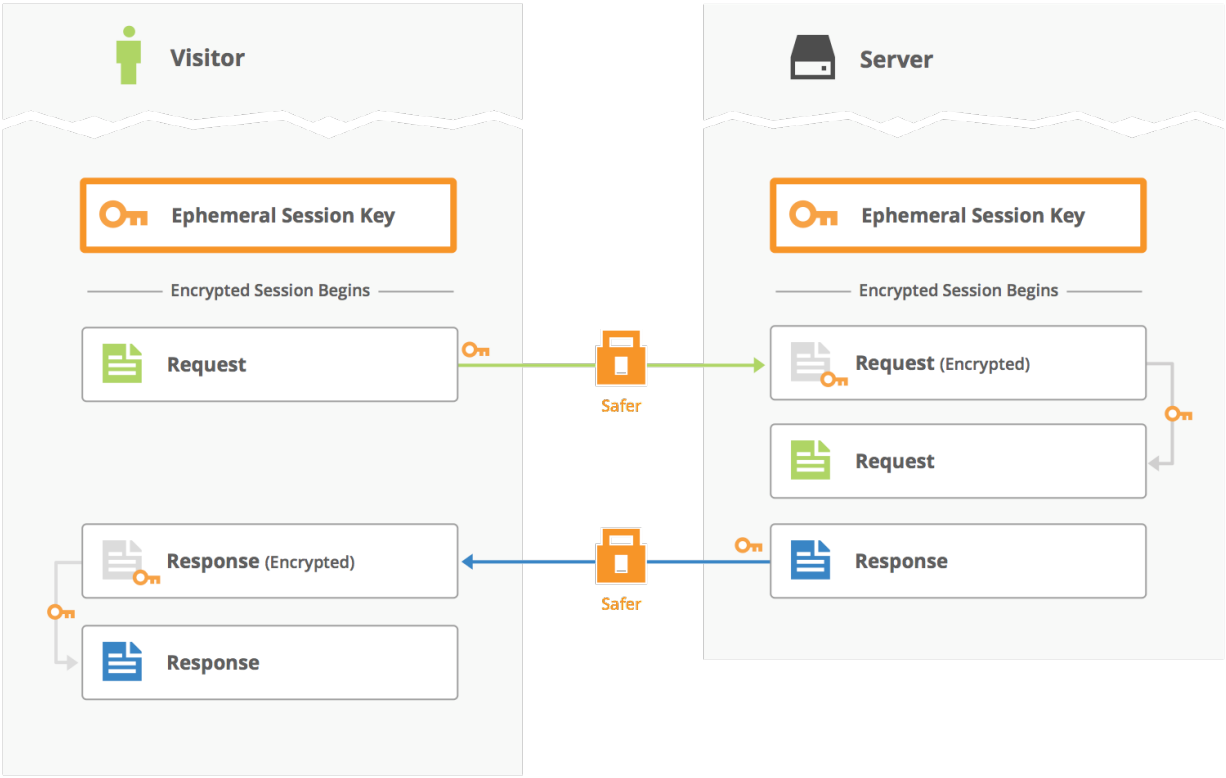
More secure SSL certificates with SHA-2 signatures

Modern websites should use a TLS certificate signed with SHA-2 instead of SHA-1. If you're still using a SHA-1 certificate, major browser vendors have already begun showing warning pages telling your website visitors that they're visiting an insecure site. By the end of 2016, browsers will stop letting users visit your website altogether. To upgrade, you need to purchase a new SHA-2 certificate from your certificate authority and install it on your web server.

If you still need to continue supporting users that cannot use SHA-2, consider using a provider (software or CDN) that can select the highest-security certificate the user's browser can support.

Support Forward Secrecy with Ephemeral Diffie-Hellman

If your server always derives its symmetric TLS session keys with the same private key, your private key becomes a weak link in the chain. For example, if an attacker recorded a bunch of encrypted traffic between your server and your users then managed to steal the private key from your server, they would be able to decrypt all of that traffic after the fact.



More secure requests and responses with Ephemeral Diffie-Hellman

Ephemeral Diffie-Hellman (EDH or DHE) keeps your users' encrypted session safe even if your private keys are stolen later on. EDH is a key-exchange mechanism that creates a unique symmetric key for each session, which means that even if your server's private key is stolen years from now, an attacker couldn't use it to decrypt recorded sessions.

Support TLS 1.2 Only

Cryptography doesn't stand still, so secure web servers should always try to support the latest and greatest advancements in Internet security. Ideally, your website should not support anything below the most current version of TLS, version 1.2. Earlier versions of the TLS and SSL protocols include outdated ciphers suites or insecure implementations that leave your encrypted traffic vulnerable to attackers.

TLS connections rely on both the client's and the server's capabilities. You may be worried that some of your audience may be left out in the cold if you don't support older SSL implementations, but in reality, the majority of browsers have supported TLS 1.2 for quite some time (with the notable exception of Android):

- **Chrome** since version 30 (Sep 2013)
- **Firefox** since version 27 (Feb 2014)
- **Opera** since version 17 (Oct 2013)
- **Safari** since version 7 (Oct 2013)
- **Safari Mobile** since iOS 5 (Oct 2011)
- **Internet Explorer** since version 11 (Oct 2013)
- **Windows Phone** since version 8.1 (Apr 2014)
- **Android Browser** since Lollipop (Sep 2015)

Upgrading to TLS 1.2 will be obligatory for PCI-compliant organizations by the end of June 2016, so there's really no reason not to enforce a TLS 1.2-only policy for your website right now.

Keeping your server's SSL implementation up to date does have a tangible impact on your website's security. SSL 3.0 was vulnerable to the POODLE attack and also suffered from a broken RC4 encryption implementation. TLS 1.0 fixed this, but then new vulnerabilities were found: implicit initialization vectors and incorrect handling of padding errors resulted in insecure cipher block chaining (CBC) modes of encryption. The cycle repeated with TLS 1.1, which is susceptible to the Lucky 13 attack. TLS 1.2 allows the use of AES-GCM cipher suites, which don't have the same vulnerabilities as CBC mode encryption.

Leverage Modern SSL Performance Optimizations

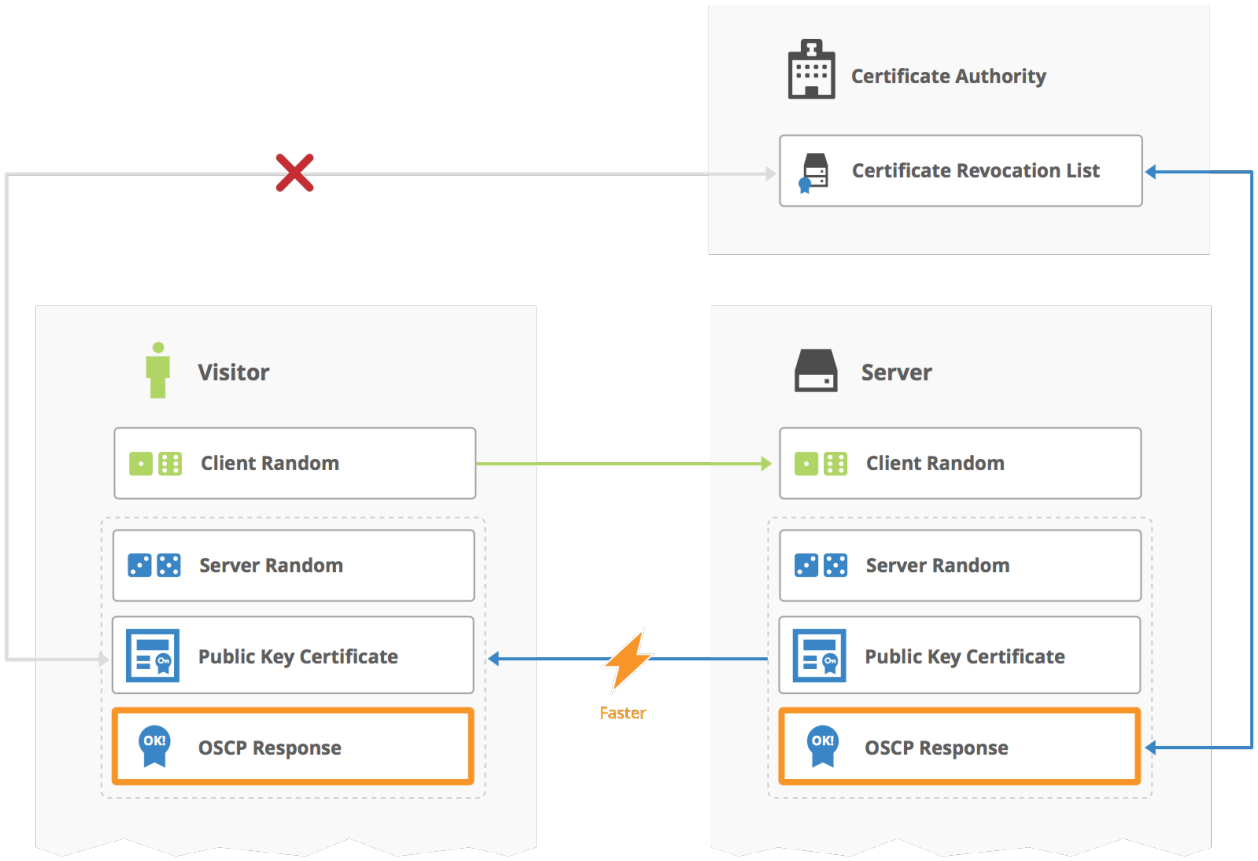
Indeed, SSL used to be significantly slower than unencrypted HTTP, which is why a lot of web developers and network administrators erred towards http:// when they could. However, the Internet community has made great strides towards making SSL more efficient. By leveraging both the latest TLS performance features as well as SPDY or HTTP/2, modern SSL websites are often faster that their unencrypted counterparts.

Support OCSP Stapling

Certificate Authorities and TLS are not infallible. Sometimes a CA issues a certificate that they weren't supposed to, sometimes a company changes its security policies in a way that voids their certificate, and sometimes the keys to certificates are outright stolen by malicious users. Whatever the reason, CAs and browser vendors recognized they needed a way to "revoke" certificates that could potentially be compromised.

For your website's users to actually trust the TLS certificate your server presents to their browser, they need to query the Certificate Revocation List (CRL) maintained by your CA to see if the certificate has been revoked. Unfortunately, this extra revocation check can slow down your users' page load times, as their browsers have to wait for the CA to return the status of your TLS certificate. This check typically requires an additional DNS lookup and downloading a large list of revoked certificates, resulting in a significant performance penalty.

OCSP Stapling is part of the Online Certificate Status Protocol (OCSP) that fixes some of the performance issues with real-time revocation checking. Instead of forcing users' browsers to query your CA directly, your own web server queries the CA at regular intervals to retrieve an OCSP response. This response is signed by your CA as proof that your TLS certificate is valid for a specified time period. Your server can then "staple" this response to the certificate in the initial TLS handshake, eliminating an unnecessary round-trip for your website visitors.

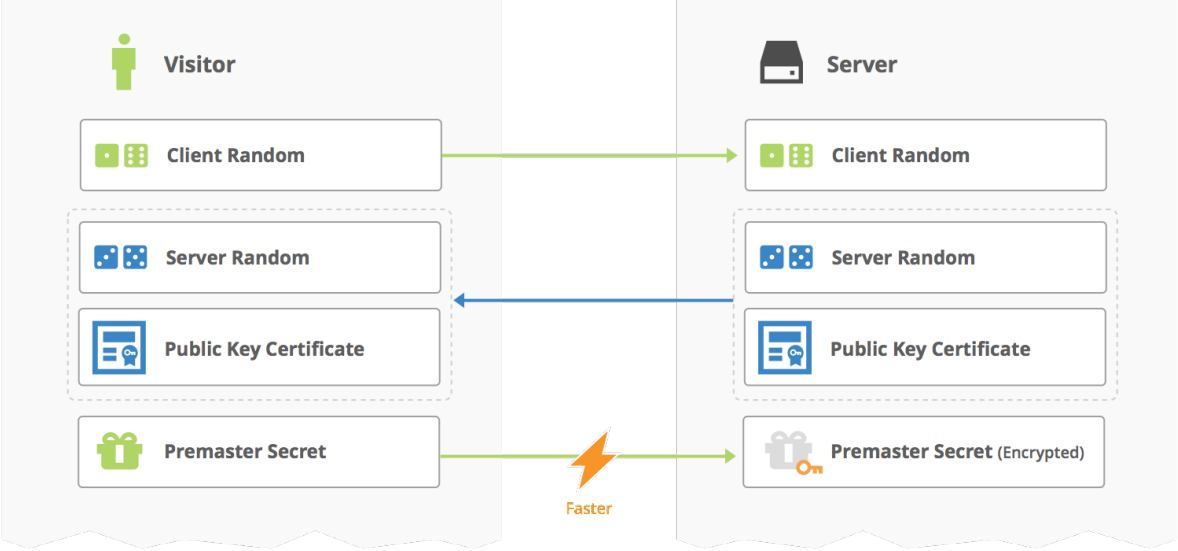


Faster SSL handshakes with OSCP stapling

Support Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) is a modern alternative to RSA. Instead of relying on prime numbers, ECC uses the mathematical properties of elliptic curves to create secure one-way functions, which are the foundation of public-key cryptography.

The advantage to using ECC is that they require shorter keys, so there's fewer bits to transfer to the end-user's computer. This speeds up the initial TLS handshake that precedes every SSL connection. In addition, ECC takes 10x fewer cycles than RSA to create a signature. Remember that every TLS handshake needs to be signed by your private key, so ECC can actually reduce the CPU load on your servers, too.



Faster SSL handshakes with elliptic curve cryptography

Support TLS Session Resumption With Session Tickets

There's no getting around the fact that TLS handshakes are expensive; however, TLS does include a way to avoid unnecessary handshakes. TLS session resumption lets a client reconnect with a server using the symmetric key from a previous session. This eliminates an extra round trip and the overhead of deriving a new symmetric key. There are two different TLS resumption mechanisms: tickets and IDs.

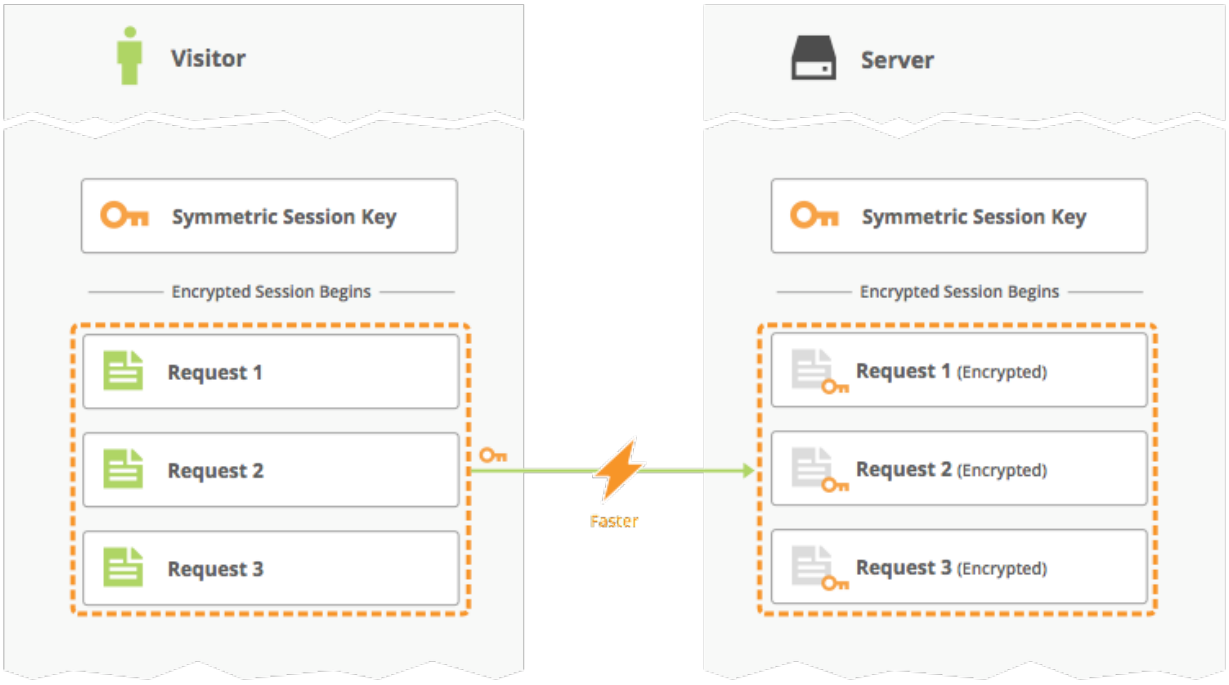
If the browser and server both support TLS session tickets, your server will send back a "ticket" containing the symmetric key that was negotiated as part of the handshake. This ticket is encrypted using a secret key that only the server knows, making it a secure way to store TLS session information. When the browser tries to reconnect to your server, it sends back the encrypted session ticket. Everybody can skip the rest of the TLS handshake, since both parties now have access to the symmetric session key.

Unfortunately, Safari and Internet Explorer do not support session tickets. They resume TLS sessions using a mechanism called session IDs. Session information is stored on the server instead of the client, and the client sends the ID of their session back to the server. Maintaining a cache of all TLS connection parameters can be a burden on your server, but if you want broad support for TLS session resumption, it's a good idea to support both session tickets and session IDs.

Support SPDY or HTTP/2

SPDY and HTTP/2 are Internet protocols that dramatically speed up load times of your website. SSL isn't technically required for either, but no major browser vendor supports either one over plain old HTTP.

HTTP/1.1 forces each request into its own TCP connection, and browsers can only open a limited number of TCP connections at a time. As a result, your website's HTML, CSS, JavaScript, and media assets are downloaded one after the other. HTTP/2 and SPDY add a new feature called multiplexing, which lets browsers use a single TCP connection for your entire website. Unlike HTTP/1.1, multiplexing lets browsers download all of your site's assets in parallel. Needless to say, this is much faster than HTTP/1.1.



Faster requests with HTTP/2 multiplexing

Around 80% of Internet traffic supports either SPDY or HTTP/2. You're not making the most of your TLS-enabled web server if it doesn't support this broad base of SPDY and HTTP/2 browsers. This is a good example of modern SSL actually making the web faster, despite the overhead of its initial handshake.

Conclusion: HTTPS All The Things

Considering the security and performance benefits of modern SSL, there's really no reason why your website shouldn't be using HTTPS for every request. This simplifies the life of web developers and system administrators, since they no longer need to write complicated URL rewrites to make some pages HTTP and others HTTPS.

Opting for TLS also builds trust between you and your website visitors. HTTPS ensures that your visitors are seeing exactly what they're supposed to be seeing on your website. Common HTTP-only scenarios like ISPs injecting advertisements into your web pages can't happen when you're serving up content over HTTPS.

To wrap up, there's four modern SSL best practices when it comes to securing your website:

- Support HSTS headers
- Use SHA-2 certificate signatures
- Enable forward secrecy with Ephemeral Diffie-Hellman handshakes
- Upgrade to TLS 1.2

And, there's four best practices to make your SSL-enabled website more performant:

- Support OCSP stapling
- Leverage elliptic curve cryptography
- Allow TLS session resumption
- Enable HTTP/2 or SPDY

CloudFlare offers a one-click solution for all of these technologies. By joining the CloudFlare network, you can benefit from HSTS headers, SHA-2 certificates, Ephemeral Diffie-Hellman, TLS 1.2, OCSP stapling, ECC, TLS session resumption, and HTTP/2 with no changes to your existing application.

For more information about how we can help secure and speed up your web properties, contact our team at:

+1 888 99 FLARE (US office)
+44 20 3713 4479 (UK office)
enterprise@cloudflare.com

About CloudFlare

CloudFlare, Inc. (www.cloudflare.com / @cloudflare) makes any Internet application lightning fast, protects them from attacks, ensures they are always online, and makes it simple to add SSL with a single click. Regardless of size or platform, CloudFlare supercharges Internet applications without the need to add hardware, install software, or change a line of code.

The CloudFlare community gets stronger as it grows: every new application makes the network smarter. More than 5 percent of global Internet requests flow through CloudFlare's network; every month more than 1.8 billion people experience a faster, safer, better Internet. CloudFlare was recognized by the World Economic Forum as a Technology Pioneer, named the Most Innovative Network & Internet Technology Company for two years running by the Wall Street Journal, and ranked among the world's 50 most innovative companies by Fast Company.

1 888 99 FLARE | enterprise@cloudflare.com | www.cloudflare.com