At CloudFlare we are constantly working on ways to make the Internet better. An important part of this is enabling our customers to serve their sites encrypted over SSL/TLS. There are some interesting technical challenges in serving sites over TLS at CloudFlare’s scale. The computational cost of the cryptography required on our servers is one of those challenges. [Elliptic curve cryptography (ECC)](http://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography) is one of the more promising technologies in this area. ECC-enabled TLS is faster and more scalable on our servers and provides the same or better security than the default cryptography in use on the web.

In this blog post we will explore how one elliptic curve algorithm, the elliptic curve digital signature algorithm (ECDSA), can be used to improve performance on the Internet. The tl;dr is: CloudFlare now supports custom ECDSA certificates for our customers and that’s good for everybody using the Internet.

**Websites and Certificates**

When you visit a site that starts with https:// instead of http://, your browser connects to that site over an encrypted connection. The browser also validates that the site is who it claims to be using public key cryptography and a digital certificate.

In public key cryptography each person has a pair of keys: a public key and a private key. These are typically numbers that are chosen to have a specific mathematical relationship. In RSA, the public key is a large number that is a product of two primes, plus a smaller number. The private key is a related number. In ECC, the public key is an equation for an elliptic curve and a point that lies on that curve. The private key is a number. See our previous blog post on [elliptic curve cryptography](http://blog.cloudflare.com/a-relatively-easy-to-understand-primer-on-elliptic-curve-cryptography) for more details.

The private key can be used to create a digital signature for any piece of data using a digital signature algorithm. This typically involves taking a cryptographic hash of the data and operating on it mathematically using the private key. Anyone with the public key can check that this signature was created using the private key and the appropriate signature validation algorithm. A digital signature is a powerful tool because it allows you to publicly vouch for any message.

A website certificate usually contains two things:

* Identity information: Typically who owns the certificate and which domains the certificate is valid for.
* A public key: The public half of a key pair, the site owner controls and keeps secret the associated private key.

The certificate is digitally signed by a trusted certificate authority who validates the identity of the site owner.

Since the introduction of SSL by Netscape in 1994, certificates for web sites have typically used a public/private key pair based on the RSA algorithm. As the SSL specification evolved into TLS, support for different public key algorithms were added. One of the supported algorithms is ECDSA which is based on elliptic curves.

Despite the number of options available in TLS, almost all certificates used on the web today are RSA-based. Web sites have been slow to adopt new algorithms because they want to maintain support for legacy browsers that don’t support the new algorithms. Even as late as 2012, out of 13 million TLS certificates found in a [scan of the internet](https://scans.io/study/umich-https), fewer than 50 use an ECDSA key pair.

**The Popular Choice**

Although ECDSA has not taken off on the web, it has become the digital signature scheme of choice for new cryptographic non-web applications.

Bitcoin is a good example of a system that relies on ECDSA for security. Every Bitcoin address is a cryptographic hash of an ECDSA public key. The ownership of the account is determined by who controls the ECDSA private key. To transfer an amount of Bitcoin to another person, you create a message that says something along the lines of “I give this Bitcoin to address X”, sign it with your private key and submit it to the Bitcoin system. The linchpin of the security and consistency of the Bitcoin system is the security of ECDSA private keys.

Elliptic curves and ECDSA in particular are also used in messaging and systems security. In Apple’s recent [white paper on iOS security](http://images.apple.com/ipad/business/docs/iOS_Security_Feb14.pdf), they relayed how they use ECDSA extensively in the Apple ecosystem. Messages through iMessage are signed with ECDSA and iCloud keychain syncing relies on ECDSA. More and more technologies are using ECDSA for security, including end-to-end encrypted messaging services [TextSecure](https://github.com/WhisperSystems/TextSecure/) and [CryptoCat](https://crypto.cat).

**ECDSA vs RSA**

Why is ECDSA the algorithm of choice for new protocols when RSA is available and has been the gold standard for asymmetric cryptography since 1977? It boils down to the fact that we are better at breaking RSA than we are at breaking ECC.

As we described in a [previous blog post](http://blog.cloudflare.com/why-are-some-keys-small), the security of a key depends on its size and its algorithm. Some algorithms are easier to break than others and require larger keys for the same level of security. Breaking an RSA key requires you to factor a large number. We are pretty good at factoring large numbers and [getting better all the time](http://bristolcrypto.blogspot.com/2013/02/discrete-logarithms.html). Breaking an ECDSA key requires you to solve the Elliptic Curve Discrete Logarithm Problem (ECDLP). The mathematical community has not made any major progress in improving algorithms to solve this problem since is was independently introduced by Koblitz and Miller in 1985.

This means that with ECDSA you can get the same level of security as RSA but with smaller keys. Smaller keys are better than larger keys for several reasons. Smaller keys have faster algorithms for generating signatures because the math involves smaller numbers. Smaller public keys mean smaller certificates and less data to pass around to establish a TLS connection. This means quicker connections and faster loading times on websites.

According to the [ECRYPT II recommendations](https://blog.cloudflare.com/ecdsa-the-digital-signature-algorithm-of-a-better-internet/www.keylength.com/en/3/) on key length, a 256-bit elliptic curve key provides as much protection as a 3,248-bit asymmetric key. Typical RSA keys in website certificates are 2048-bits. If we compare the portion of the TLS handshake that happens on the server for 256-bit ECDSA keys against the cryptographically much weaker 2048-bit RSA keys we get the following:

sign/s

256 bit ecdsa (nistp256) 9516.8

rsa 2048 bits 1001.8

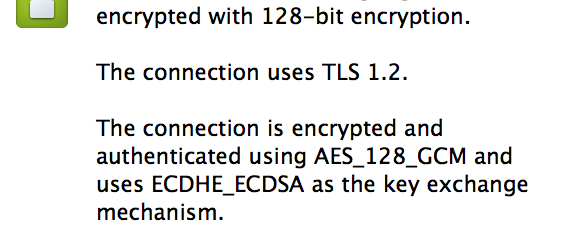
(openssl 1.0.2 beta on x86\_64 with enable-ec\_nistp\_64\_gcc\_128)

That table shows the number of ECDSA and RSA signatures possible per second. On our servers, using an ECDSA certificate reduces the cost of the private key operation by a factor of 9.5x, saving a lot of CPU cycles.

**Hello Future**

I mentioned earlier that fewer than fifty ECDSA certificate are being used on the web. You can now count <https://blog.cloudflare.com> among them. If you don't see a lock icon, click [here](https://blog.cloudflare.com/ecdsa-the-digital-signature-algorithm-of-a-better-internet) for the HTTPS version of the site. Once you are viewing this site over HTTPS, take a look at the TLS information bar (click on the lock icon in your address bar). You should see the key exchange mechanism listed as ECDHE\_ECDSA, which means the certificate is using ECDSA. If the HTTPS version site does not load, your browser probably does not support ECDSA.

This is an image taken from the Chrome browser under the green lock icon for this page under the connection tab:



This blog post is our first experiment using an SSL certificate based on elliptic curves. Our blog is being served by the standard CloudFlare service (yes, we [eat our own dog food](http://en.wikipedia.org/wiki/Eating_your_own_dog_food)), and is the first site on CloudFlare that uses an ECDSA certificate. We are happy to annouce that we now support custom ECDSA certificates for all CloudFlare business customers.

In the near future we will enable code that will allow sites to have a fallback certificate so that visitors with old browsers without ECDSA support can still view their site over HTTPS. Because ECDSA is so much more efficient for our servers, supporting these certificates is an essential step for enabling [SSL for free in 2014](http://blog.cloudflare.com/2013-refactoring-2014-stepping-on-the-gas).

**Danger Zone?**

We can be relatively confident about the mathematical security of ECDSA (save for some questions about the [choice of curve](http://safecurves.cr.yp.to)). The history of cryptography shows us that good cryptography has been repeatedly defeated not because of bad math, but because of bad implementations of good math.

One interesting quirk of the ECDSA algorithm is that every signature requires some random or unpredictable data as input. If the source of randomness is predictable to an attacker, then they can figure out the private key. Hackers have exploited this vulnerability in several high-profile incidents.

In 2010, a flaw in the way random numbers were used in ECDSA on Sony’s Playstation 3 resulted in a [private key being leaked](http://www.exophase.com/20540/hackers-describe-ps3-security-as-epic-fail-gain-unrestricted-access/). More recently, some Android devices were found to be incorrectly generating random values, resulting in a [massive theft](https://bitcoin.org/en/alert/2013-08-11-android) of Bitcoins from devices running Bitcoin software.

There are other more esoteric attacks against specific ECDSA implementations. Last week, [a paper was published](http://eprint.iacr.org/2014/161.pdf) by researchers from Australia and the UK describing an attack on OpenSSL’s implementation of ECDSA for curve secp256k1 (the one used by the Bitcoin protocol). Luckily, this attack is not a threat against busy remote servers.

The danger of key leakage via poor random data or [side channel attacks](http://en.wikipedia.org/wiki/Side_channel_attack) is a concern but is manageable with proper preparation. At CloudFlare we ensure that the system random number generator [has enough entropy](http://blog.cloudflare.com/ensuring-randomness-with-linuxs-random-number-generator). Even if there is a problem with the system PRNG, OpenSSL 1.0.2 has included a fix to reduce the chance of compromise. Cryptography is hard to implement correctly, especially in the context of a complex protocol like TLS as evidenced in some famous [recent](https://www.imperialviolet.org/2014/02/22/applebug.html) [bug fixes](http://arstechnica.com/security/2014/03/critical-crypto-bug-leaves-linux-hundreds-of-apps-open-to-eavesdropping/). That said, the benefits seem to outweigh the risks in this case.

**Conclusion**

On a personal note, Dr. Vanstone was one of my professors at the University of Waterloo. He was passionate about mathematics and cryptography and he was one of the reasons I decided to pursue security engineering as a career. The book he co-authored, [The Handbook of Applied Cryptography](http://cacr.uwaterloo.ca/hac/), is still one of the classics in the field.

From his [memorial page](http://www.bulletin.uwaterloo.ca/2014/mar/06th.html) at the Waterloo Daily Bulletin: "I had studied it enough to believe in it," Vanstone told Silicon Valley North in 2003. "It was the rest of the world that didn't believe in it." He will be missed.

Elliptic curve cryptography is a powerful technology that can enable faster and more secure cryptography across the Internet. The time has come for ECDSA to be widely deployed on the web, just as Dr. Vanstone hoped. We are taking the first steps towards that goal by enabling customers to use ECDSA certificates on their CloudFlare-enabled sites.