

# ECDSA for WebRTC: Better Security, Better Privacy and Better Performance



By Sam Dutton

Sam is a Developer Advocate

From Chrome 52, WebRTC uses a much more efficient and secure algorithm for certificate (RTCCertificate) generation: ECDSA. In addition, RTCCertificates can now be stored with IndexedDB.

RTCCertificates are the self-signed certificates used in the DTLS handshake when setting up a WebRTC peer connection. (DTLS is an implementation of the cryptographic protocol TLS for datagram protocols such as UDP, which is used by WebRTC.)

Until recently, WebRTC used RSA-1024 keys for certificates. There are several disadvantages with these keys:

- Generating RSA-1024 keys can add up to around 1000ms in call setup time.
- 1024-bit RSA keys do not provide adequate cryptographic strength.

Because certificate generation with RSA-1024 is slow, some mobile apps have resorted to preparing certificates in advance or reusing them.

The key strength issue could be resolved by going to 2048-bit RSA keys or more, but that would delay call setup by several additional seconds. Instead of changing the RSA key size, Chrome 52 implements ECDSA keys (Elliptic Curve Digital Signature Algorithm) for use in certificates. These are as strong as 3072-bit RSA keys, but several thousand times faster: call setup overhead with ECDSA is just a few milliseconds.

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. 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 it was independently introduced by Koblitz and Miller in 1985.

— Nick Sullivan, CloudFlare

All in all, ECDSA keys mean better security, better privacy and better performance — especially on mobile. For these reasons, ECDSA has been mandated in the WebRTC Security.

## Architecture draft.

From Chrome 47 you can opt in to ECDSA:

```
// or webkitRTCPeerConnection
RTCPeerConnection.generateCertificate({
  name: "ECDSA",
  namedCurve: "P-256"
}).then(function(certificate) {
  var pc = new RTCPeerConnection({..., certificates: [certificate]});
});
```



From Chrome 52, though ECDSA is enabled by default, you can still choose to generate RSA certificates:

```
pc.generateCertificate({
  name: "RSASSA-PKCS1-v1_5",
  modulusLength: 2048,
  publicExponent: new Uint8Array([1, 0, 1]),
  hash: "SHA-256"
})
```



(See the [W3C draft](#) for more information about `generateCertificate()`.)


## Storing RTCCertificate in IndexedDB

Another improvement in Chrome 52: the RTCCertificates used by WebRTC can be saved and loaded from IndexedDB storage, avoiding the need to generate new certificates between sessions. This can be useful, for example, if you still need to use RSA and want to avoid the RSA generation overhead. With ECDSA, caching is not necessary since it is fast enough to generate a new certificate every time.

RTCCertificate IndexedDB storage has already shipped in Firefox and is in Opera 39.

## Find out more

- [ECDSA: The digital signature algorithm of a better internet](#)
- [WebRTC certificate management](#)
- [RTCCertificate interface](#)
- [Discussion of RTCCertificates and storage](#)

- [Getting started with WebRTC](#) 

---

*Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.*

*Last updated July 2, 2018.*