boxcryptor

Promises of a Safer Web

WebCrypto in 2016

Agenda

1) What is WebCrypto?

2) Why we need it

3) Built-in security measures

4) Support in 2016 – officially and actually

5) What if it doesn‘t work? Fun with polyfills

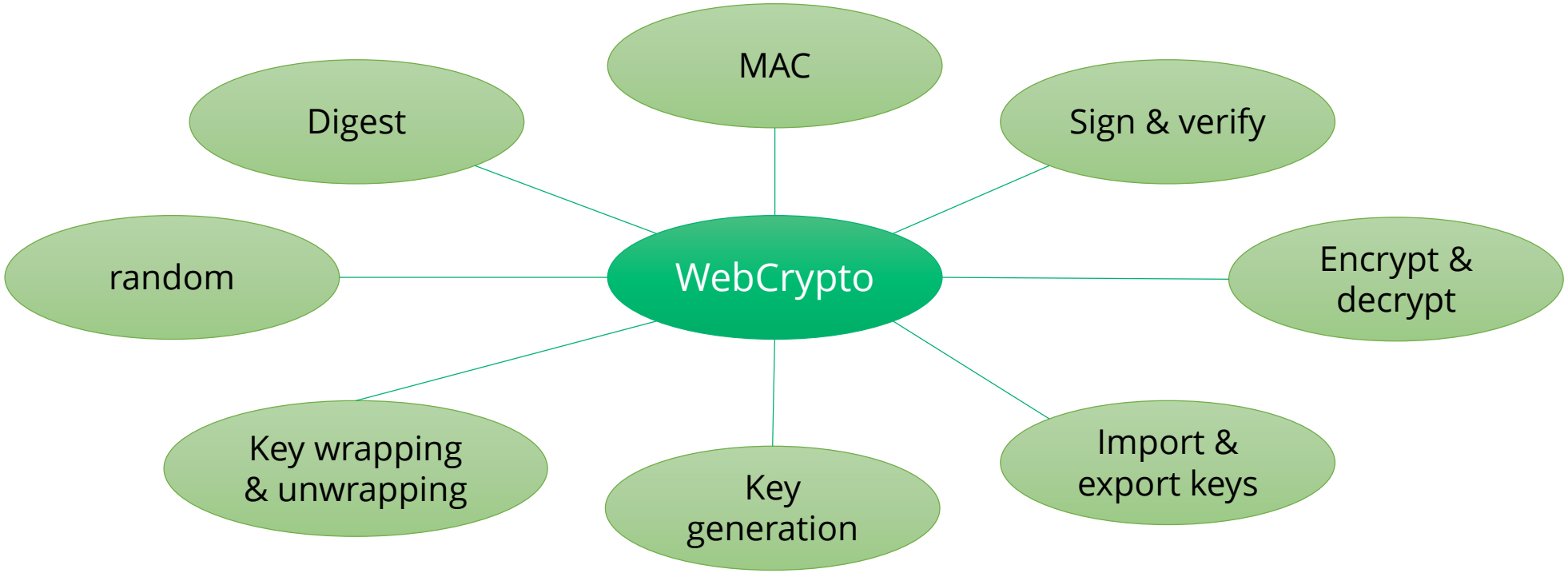
6) Tales from the front: Importing RSA keys and „bending bricks“

7) How to use WebCrypto – and how not to.

8) Lookout and question time!

1) What is WebCrypto?

An API available in all modern browsers, offering cryptographic primitives and secure storage of keys.

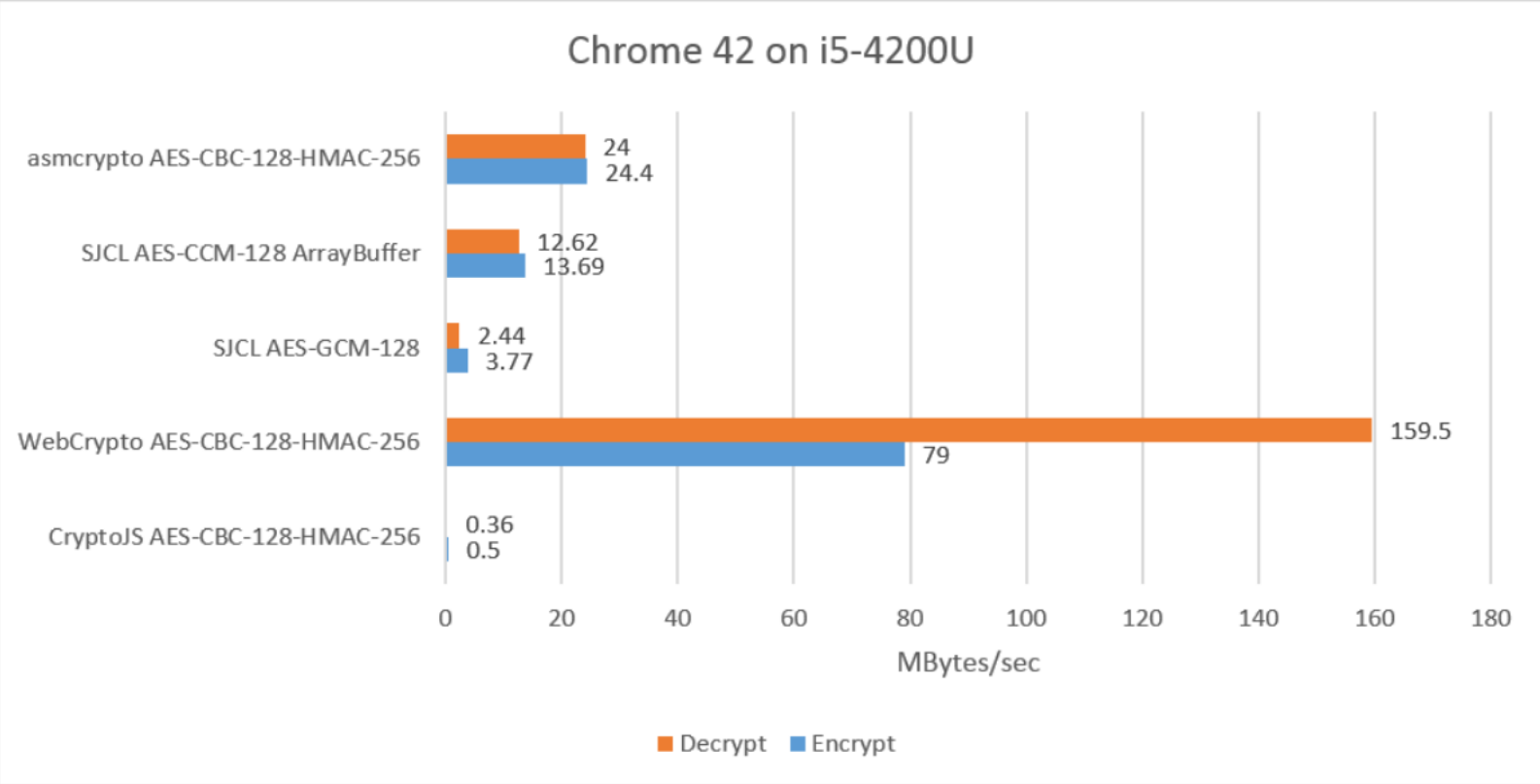


What does it look like?

```
window.crypto.subtle.generateKey(
    {
        name: "AES-CBC",
        length: 256, //can be  128, 192, or 256
    },
    false, //whether the key is extractable (i.e. can be used in exportKey)
    ["encrypt", "decrypt"] //can be "encrypt", "decrypt", "wrapKey", or "unwrapKey"
)
.then(function(key){
    //returns a key object
    console.log(key);
})
.catch(function(err){
    console.error(err);
});
```

2) Why we need it

WebCrypto is fast



2) Why we need it

Many JS libraries are insecure

(just look how they generate RSA keys – assumed they even provide the required key length)

Some things are simply impossible without WebCrypto:

- Generating secure random numbers
- Securing keys on the client

3) Security measures of WebCrypto

Protection against overwriting (in Chrome)

- window.crypto.subtle is read-only
- This makes polyfills more difficult, though…

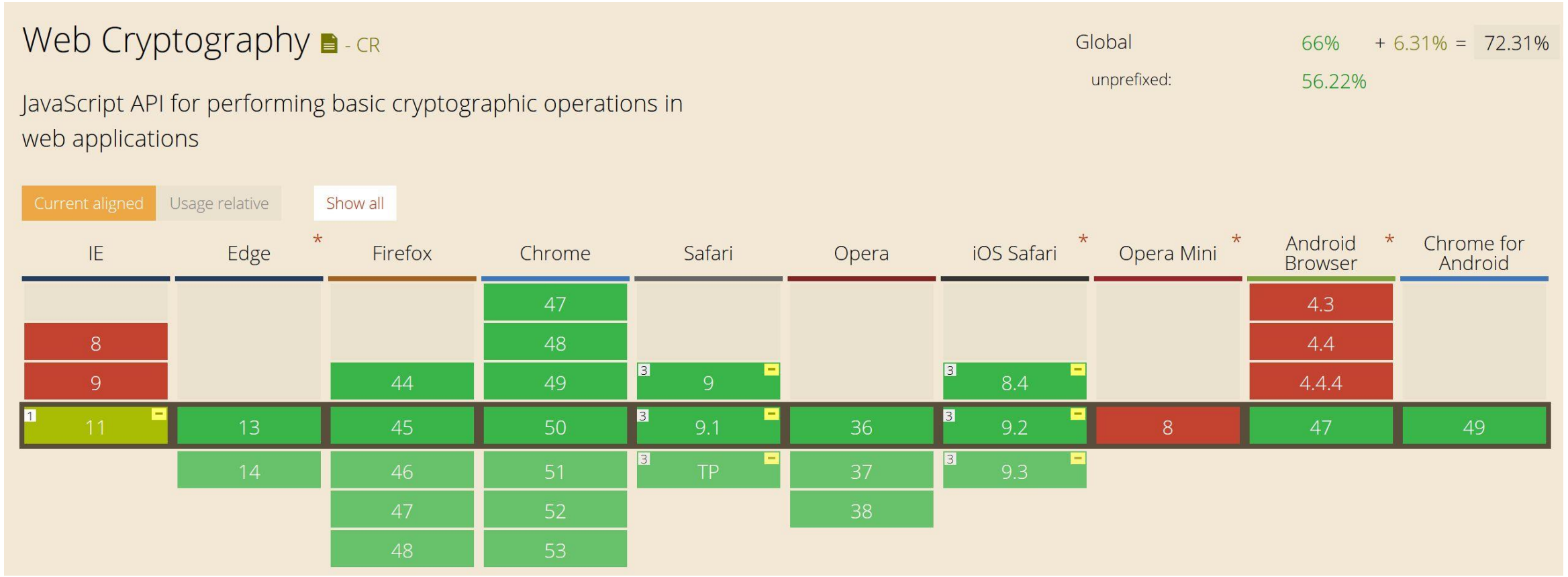
Non-extractable keys

- This partly solves the „JavaScript is always insecure“-issue
- You can create a key, store it at the client, and use it. But if you don‘t specify it, you can never read the key.

  → This means an attacker cannot steal your keys.
- Use IndexedDB to store the keys (they are not serializable).

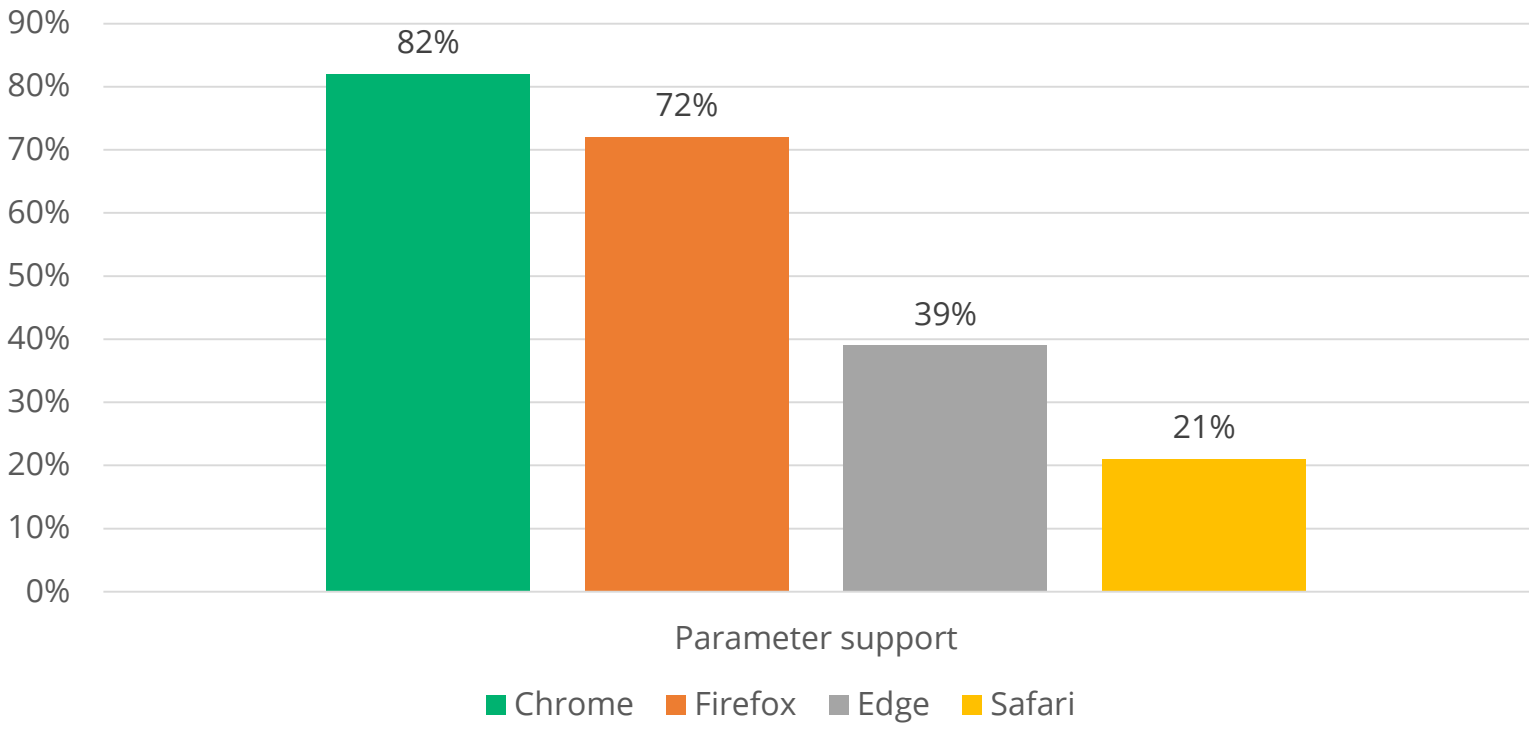
SSL only (exception: localhost and extensions)

4) Support in 2016 - officially



4) Support in 2016 - actually

Live view: https://diafygi.github.io/webcrypto-examples/



5) What if it doesn‘t work?

Good Browsers throw „good“ exceptions

Actual browsers throw exceptions of mixed usabillity

- Chrome: „192-bit AES keys are not supported“
- Firefox: „SyntaxError: An invalid or illegal string was specified”
- Edge: "{name: "ECDSA", namedCurve: "P-256"}“
- Safari: „TypeError: undefined is not an object (evaluating 'window.crypto.subtle.importKey')”

This actually improved compared to 2015

5) What if it doesn‘t work II - Polyfills

Kind of required if everybody should be able to use your app

Incomplete

- We didn‘t find a polyfill that is complete and secure (and there cannot be one, either: RSA Ops are too expensive)
- The best ones are missing basic operations, too (e.g. AES-CBC)

→ use many or extend yourself

Slow

PBKDF2-SHA512 test in Chrome: 100.000 iterations
JS-implementation: 10 seconds
WebCrypto: 0 seconds

5) What if it doesn‘t work II - Polyfills

There‘s another issue: Just because the browser lets you create a key doesn‘t mean you can use it for other WebCrypto operations

...or for JS operations, because the key is secured -.-

- What if WebCrypto created a key and the polyfill has to use it?

→ The polyfill has to calculate two keys – always.

1. Create WebCrypto key
2. Append JS-key equivalent (so the WebCrypto key is still useable as parameter)
3. If the polyfill is required, check for appended JS-key

6) More tales from the front: RSA key import

There are a lot of ways to encode RSA keys

PKCS#1 RSAPublicKey, PKCS#1 RSAPrivateKey

PKCS#8 (private key only)

X.509 (public key only)

XML

SPKI

JWK

Just raw bytes

We found JWK to be the (currently) mostly supported way to import (general) keys.

→ unpack DER packages, fill JWK, deal with Browser-quirks (e.g. „0100 is not a number because it starts with a zero“).

So you want to encrypt a file… what do you do?

a) You load your files in blocks into memory and encrypt them

b) You load your complete, huge file into memory and encrypt it in one run

WebCrypto expects you to do b)…

There is no „progress“

- Users are very patient when the app freezes anyways, right?
- WebCrypto might still be slow on mobile devices
- We actually needed to write a polyfill for working WebCrypto-PBKDF2 because of this…

7) How should you use WebCrypto?

Generally, WebCrypto is misunderstood regarding it‘s purpose.

→ It‘s <u>NOT</u> meant to protect the user from the server!

Valid use-cases:

- Protect the server from the user! (e.g. Mega upload, legal issues)
- Protect user from passive attackers (e.g. by pre-hashing the password)
- Enable zero-knowledge (e.g. Whisply, outsource the crypto to the client)

7) Ways to use WebCrypto in an insecure way

Be lazy! Way easier if every key is extractable by default
(i.e.: Be restrictive, only allow required uses/options)

Ignore best practices. Password == key?

Trust all WebCrypto functions (If it's offered it must be secure, right?)

- Old algorithms are required for legacy-reasons
- An proposal to mark them with a console warning has been declined

In short: Use WebCrypto without a basic background in cryptography. (That's why the API is available under „subtle").
I would argue that a basic course such as Stanford's free „Cryptography I" online course is sufficient.

8) To be continued...

We could see clear improvement compared to 2015

- More API coverage
- It‘s getting more stable (and documentation / tutorials are converging)
- At the end of the year this will likely behave like a „normal“ API ☺

Other things will likely stay... suboptimal

- No progress in the spec
- No iterable input feeding in the spec

We expect a lot of incorrect uses of this API. And a few clever ones, too ☺

Any questions?

boxcryptor

Thank you!

Christian Olbrich
Secomba GmbH
Werner-von-Siemens-Str. 6
86159 Augsburg
co@secomba.com