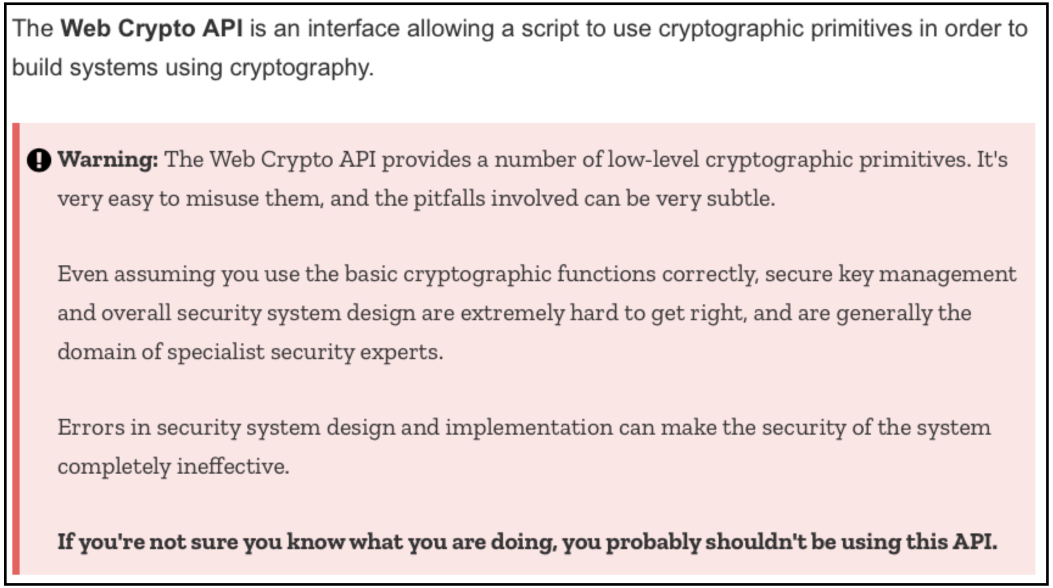
Web Crypto API

I learned enough Web Crypto to be dangerous

https://dev.to/subterrane/i-learned-enough-web-crypto-to-be-dangerous-5b5j

John Feiner

Web Crypto API



The **Web Crypto API** is an interface allowing a script to use cryptographic primitives in order to build systems using cryptography.

Warning: The Web Crypto API provides a number of low-level cryptographic primitives. It's very easy to misuse them, and the pitfalls involved can be very subtle.

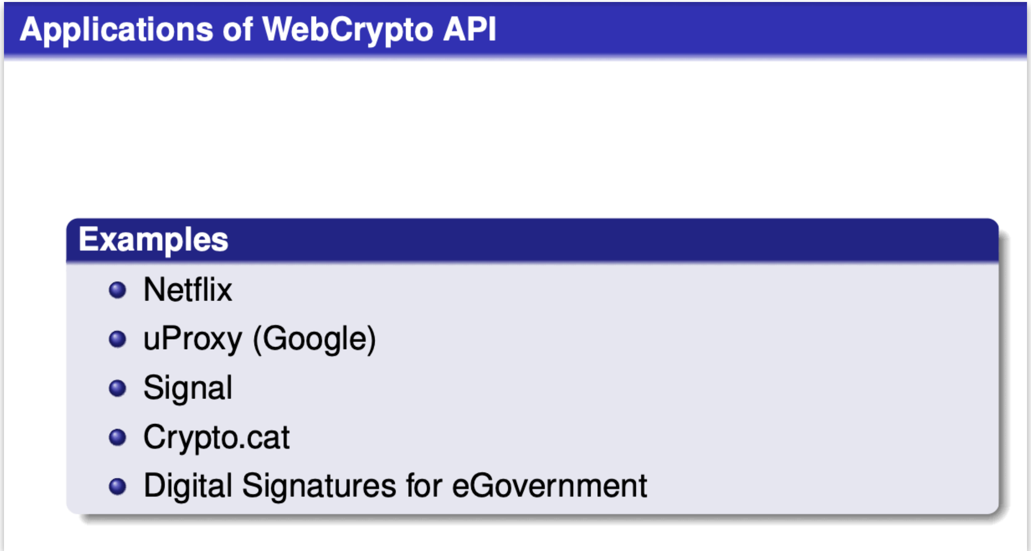
Even assuming you use the basic cryptographic functions correctly, secure key management and overall security system design are extremely hard to get right, and are generally the domain of specialist security experts.

Errors in security system design and implementation can make the security of the system completely ineffective.

If you're not sure you know what you are doing, you probably shouldn't be using this API.

https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API

Examples



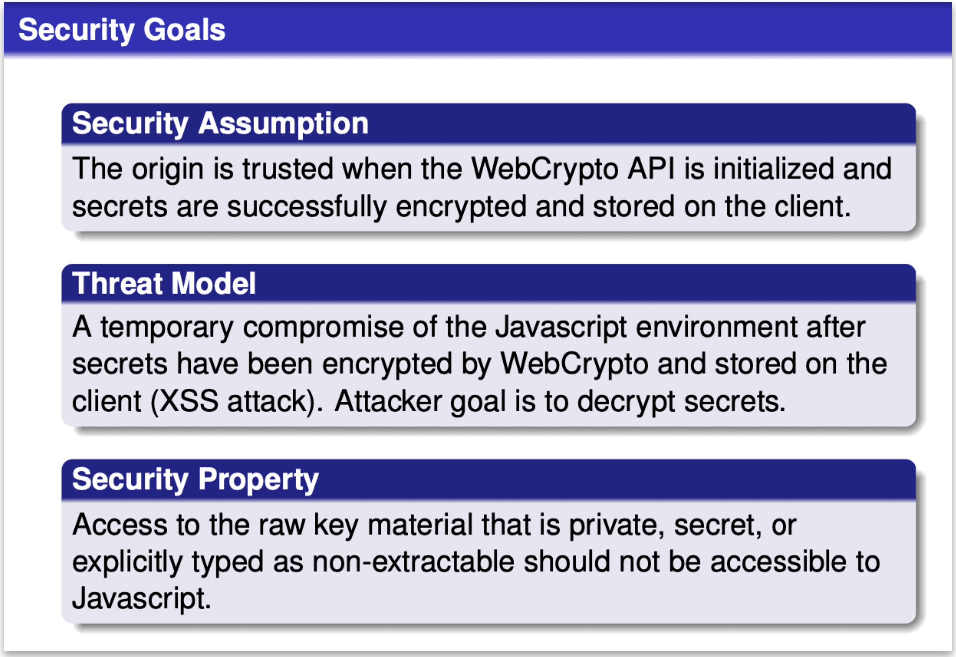
Applications of WebCrypto API

Examples

- Netflix
- uProxy (Google)
- Signal
- Crypto.cat
- Digital Signatures for eGovernment

https://csrc.nist.gov/csrc/media/events/ssr-2016-security-standardisation-research/documents/presentation-mon-halpin.pdf

Goals



Security Goals

Security Assumption

The origin is trusted when the WebCrypto API is initialized and secrets are successfully encrypted and stored on the client.

Threat Model

A temporary compromise of the Javascript environment after secrets have been encrypted by WebCrypto and stored on the client (XSS attack). Attacker goal is to decrypt secrets.

Security Property

Access to the raw key material that is private, secret, or explicitly typed as non-extractable should not be accessible to Javascript.

https://csrc.nist.gov/csrc/media/events/ssr-2016-security-standardisation-research/documents/presentation-mon-halpin.pdf

Web Cryptography API

Use cases

…

2.2. Protected Document Exchange

…

2.4. Document Signing

…

Technology:

Async functions

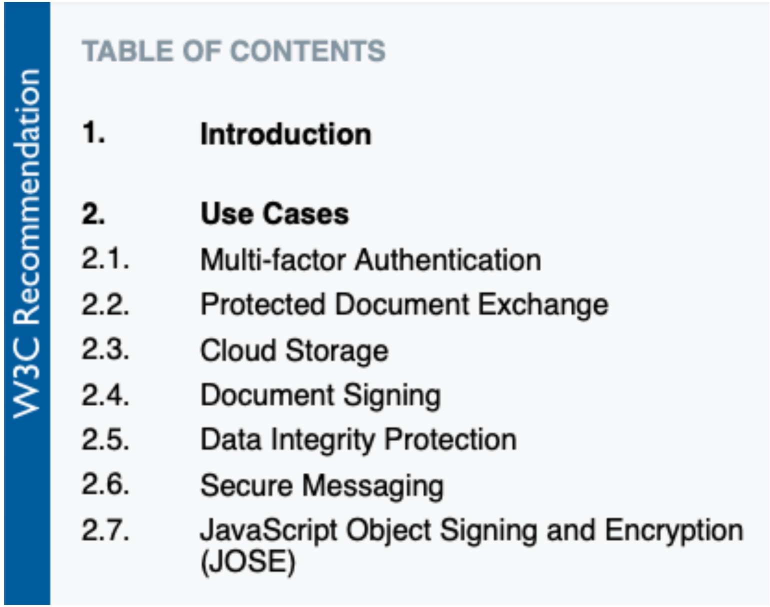
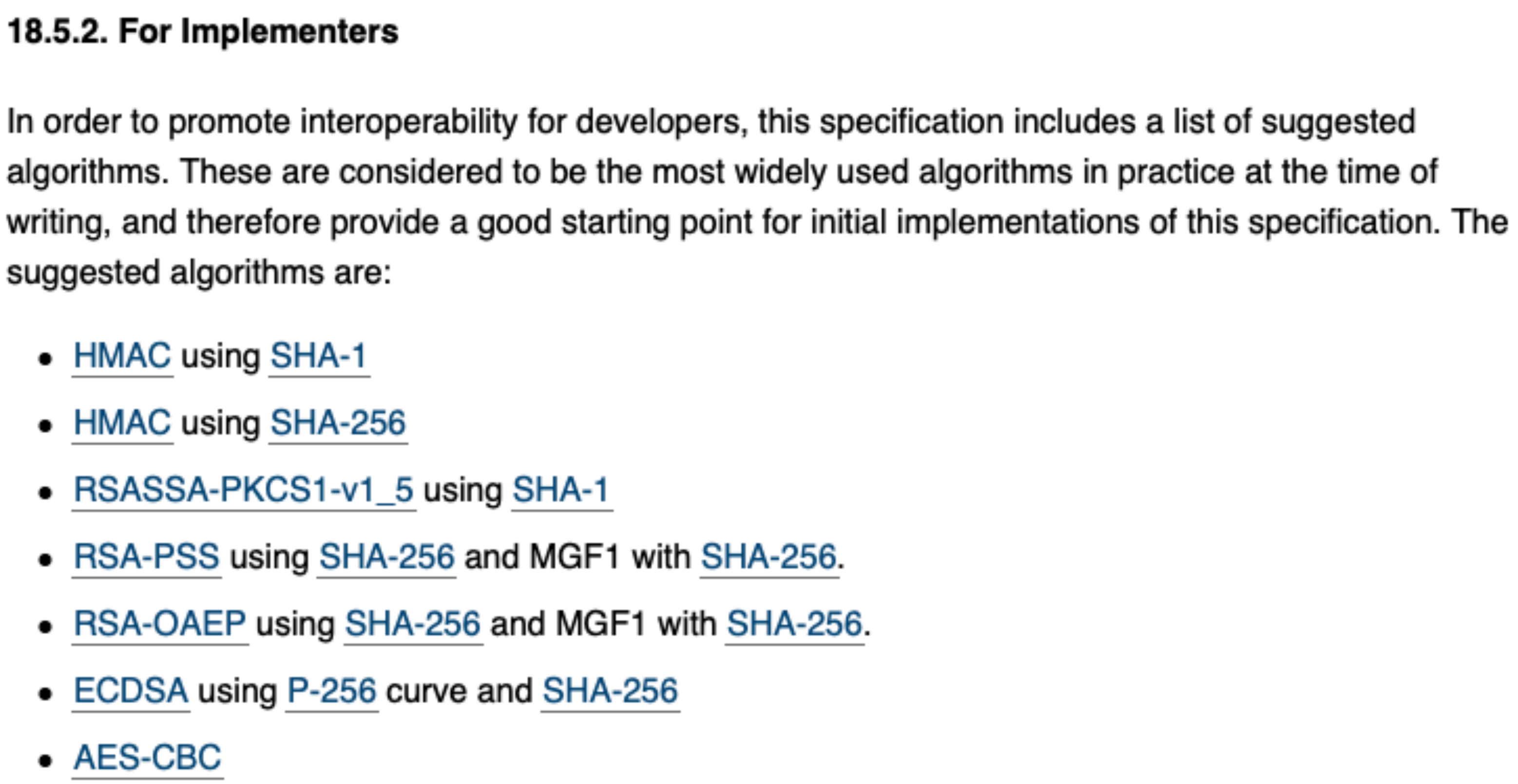


TABLE OF CONTENTS

https://www.w3.org/TR/WebCryptoAPI/

W3C Recommendation



18.5.2. For Implementers

In order to promote interoperability for developers, this specification includes a list of suggested algorithms. These are considered to be the most widely used algorithms in practice at the time of writing, and therefore provide a good starting point for initial implementations of this specification. The suggested algorithms are:

- HMAC using SHA-1
- HMAC using SHA-256
- RSASSA-PKCS1-v1_5 using SHA-1
- RSA-PSS using SHA-256 and MGF1 with SHA-256.
- RSA-OAEP using SHA-256 and MGF1 with SHA-256.
- ECDSA using P-256 curve and SHA-256
- AES-CBC

https://w3c.github.io/webcrypto/#algorithm-recommendations-implementers

Web Cryptography API

Just FYI:

Depends on: DOM, HTML, WebIDL specification

Notes for CryptoKey objects:

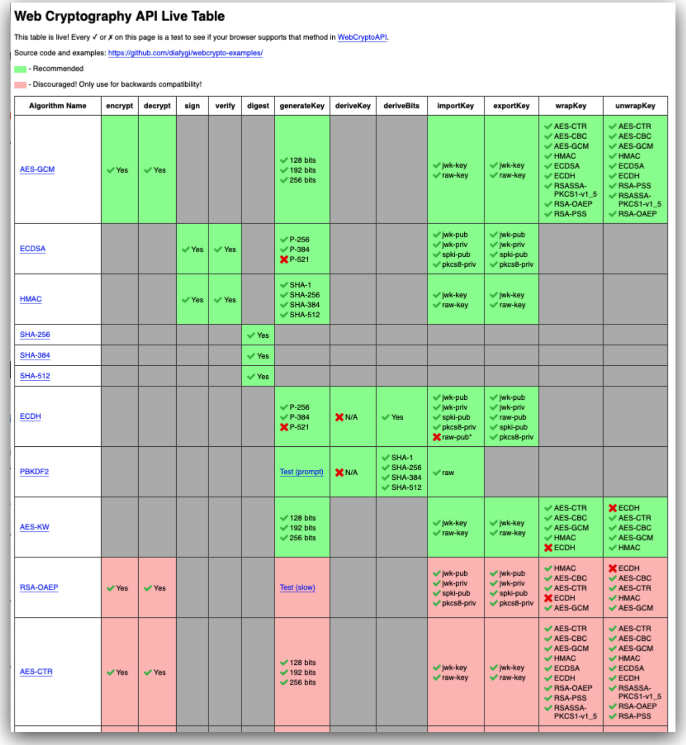
Key Storage <= use, for example, Index DB API

Examples

Check: Browser Support

Check: Recommended Algorithms

Then generate key(s)...



Web Cryptography API Live Table

This table is live! Every ✓ or ✗ on this page is a test to see if your browser supports that method in WebCryptoAPI.

Source code and examples: https://github.com/diafygi/webcrypto-examples/

🟩 - Recommended

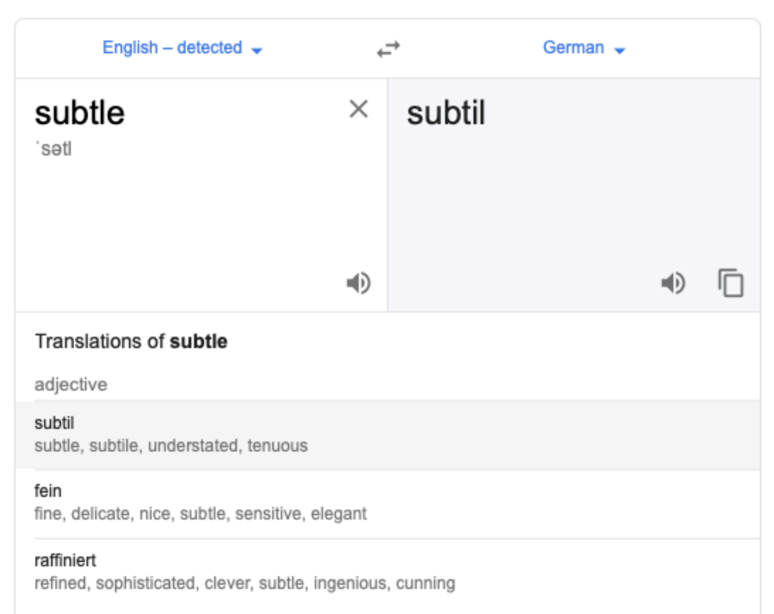
🟥 - Discouraged! Only use for backwards compatibility!

Algorithm Name	encrypt	decrypt	sign	verify	digest	generateKey	deriveKey	deriveBits	importKey	exportKey	wrapKey	unwrapKey
AES-GCM	✓ Yes	✓ Yes				✓ 128 bits ✓ 192 bits ✓ 256 bits			✓ jwk-key ✓ raw-key	✓ jwk-key ✓ raw-key	✓ AES-CTR ✓ AES-CBC ✓ AES-GCM ✓ HMAC ✓ ECDSA ✓ ECDH ✓ RSASSA-PKCS1-v1_5 ✓ RSA-OAEP ✓ RSA-PSS	✓ AES-CTR ✓ AES-CBC ✓ AES-GCM ✓ HMAC ✓ ECDSA ✓ ECDH ✓ RSA-PSS ✓ RSASSA-PKCS1-v1_5 ✓ RSA-OAEP
ECDSA			✓ Yes	✓ Yes		✓ P-256 ✓ P-384 ✗ P-521			✓ jwk-pub ✓ jwk-priv ✓ spki-pub ✓ pkcs8-priv	✓ jwk-pub ✓ jwk-priv ✓ spki-pub ✓ pkcs8-priv		
HMAC			✓ Yes	✓ Yes		✓ SHA-1 ✓ SHA-256 ✓ SHA-384 ✓ SHA-512			✓ jwk-key ✓ raw-key	✓ jwk-key ✓ raw-key		
SHA-256					✓ Yes							
SHA-384					✓ Yes							
SHA-512					✓ Yes							
ECDH						✓ P-256 ✓ P-384 ✗ P-521	✗ N/A	✓ Yes	✓ jwk-pub ✓ jwk-priv ✓ spki-pub ✓ pkcs8-priv ✗ raw-pub*	✓ jwk-pub ✓ jwk-priv ✓ raw-pub ✓ spki-pub ✓ pkcs8-priv		
PBKDF2						Test (prompt)	✗ N/A	✓ SHA-1 ✓ SHA-256 ✓ SHA-384 ✓ SHA-512	✓ raw			
AES-KW						✓ 128 bits ✓ 192 bits ✓ 256 bits			✓ jwk-key ✓ raw-key	✓ jwk-key ✓ raw-key	✓ AES-CTR ✓ AES-CBC ✓ AES-GCM ✓ HMAC ✗ ECDH	✗ ECDH ✓ AES-CTR ✓ AES-CBC ✓ AES-GCM ✓ HMAC
RSA-OAEP	✓ Yes	✓ Yes				Test (slow)			✓ jwk-pub ✓ jwk-priv ✓ spki-pub ✓ pkcs8-priv	✓ jwk-pub ✓ jwk-priv ✓ spki-pub ✓ pkcs8-priv	✓ HMAC ✓ AES-CBC ✓ AES-CTR ✗ ECDH ✓ AES-GCM	✗ ECDH ✓ AES-CBC ✓ AES-CTR ✓ HMAC ✓ AES-GCM
AES-CTR	✓ Yes	✓ Yes				✓ 128 bits ✓ 192 bits ✓ 256 bits			✓ jwk-key ✓ raw-key	✓ jwk-key ✓ raw-key	✓ AES-CTR ✓ AES-CBC ✓ AES-GCM ✓ HMAC ✓ ECDSA ✓ ECDH ✓ RSA-OAEP ✓ RSA-PSS ✓ RSASSA-PKCS1-v1_5	✓ AES-CTR ✓ AES-CBC ✓ AES-GCM ✓ HMAC ✓ ECDSA ✓ ECDH ✓ RSASSA-PKCS1-v1_5 ✓ RSA-OAEP ✓ RSA-PSS

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

https://github.com/diafygi/webcrypto-examples

Web Crypto API – Tutorial



English – detected ⇄ German

subtle
ˈsətl

subtil

Translations of **subtle**

adjective

subtil
subtle, subtile, understated, tenuous

fein
fine, delicate, nice, subtle, sensitive, elegant

raffiniert
refined, sophisticated, clever, subtle, ingenious, cunning

Translation provided by Google

Crypto.subtle

Generate HMAC (key object)

```
window.crypto.subtle.generateKey(
    {
        name: "HMAC",
        hash: {name: "SHA-256"},
    },
    false, //whether the key is extractable (i.e. can be used in exportKey)
    ["sign", "verify"] //can be any combination of "sign" and "verify"
)
.then(function(key){
    //returns a key object
    console.log(key);
})
.catch(function(err){
    console.error(err);
});
```

Security best practice:

o) Limit Algo

o) for specific Usage

o) disable extraction

o) do not reveal details

https://github.com/diafygi/webcrypto-examples

Sign / Verify Data

E.g. **Sign** using HMAC Algorithm

```
window.crypto.subtle.sign(
    {
        name: "HMAC",
    },
    key, //from generateKey or importKey above
    data //ArrayBuffer of data you want to sign
)
.then(function(signature){
    //returns an ArrayBuffer containing the signature
    console.log(new Uint8Array(signature));
})
.catch(function(err){
    console.error(err);
});
```

https://github.com/diafygi/webcrypto-examples/#hmac

Sign / Verify Data

E.g. **Verify** using HMAC Algorithm

```
window.crypto.subtle.verify(
    {
        name: "HMAC",
    },
    key, //from generateKey or importKey above
    signature, //ArrayBuffer of the signature
    data //ArrayBuffer of the data
)
.then(function(isvalid){
    //returns a boolean on
    // whether the signature is true or not
    console.log(isvalid);
})
.catch(function(err){
    console.error(err);
});
```

https://github.com/diafygi/webcrypto-examples/#hmac

AES-GCM Encrypt / Decrypt Data

```
window.crypto.subtle.encrypt(
    {
        name: "AES-GCM",

        //Don't re-use initialization vectors!
        //Always generate a new iv every time your encrypt!
        //Recommended to use 12 bytes length
        iv: window.crypto.getRandomValues(new Uint8Array(12)),

        //Additional authentication data (optional)
        additionalData: ArrayBuffer,

        //Tag length (optional)
        tagLength: 128, //can be 32, 64, 96, 104, 112, 120 or 128 (default)
    },...
```

AES-GCM Encrypt / Decrypt Data

```
...},
    key, //from generateKey or importKey above
    data //ArrayBuffer of data you want to encrypt
)
.then(function(encrypted){
    //returns an ArrayBuffer containing the encrypted data
    console.log(new Uint8Array(encrypted));
})
.catch(function(err){
    console.error(err);
});
```

AES-GCM Encrypt / Decrypt Data

```
window.crypto.subtle.decrypt(
    {
        name: "AES-GCM",
        iv: ArrayBuffer(12), //The initialization vector you used to encrypt
        additionalData: ArrayBuffer, //The addtionalData you used to encrypt (if any)
        tagLength: 128, //The tagLength you used to encrypt (if any)
    },
    key, //from generateKey or importKey above
    data //ArrayBuffer of the data
)
.then(function(decrypted){
    //returns an ArrayBuffer containing the decrypted data
    console.log(new Uint8Array(decrypted));
})
.catch(function(err){
    console.error(err);
});
```

Task

Select **one** of following task

- **Sign the log/statistic data you send back to the server**, or
- Store (encrypted) info on a server you do NOT trust
- Exchange signed(!) data with a peer
- Store (encrypted) info locally
- Send data (decrypt) with key (you got by mail)
- …

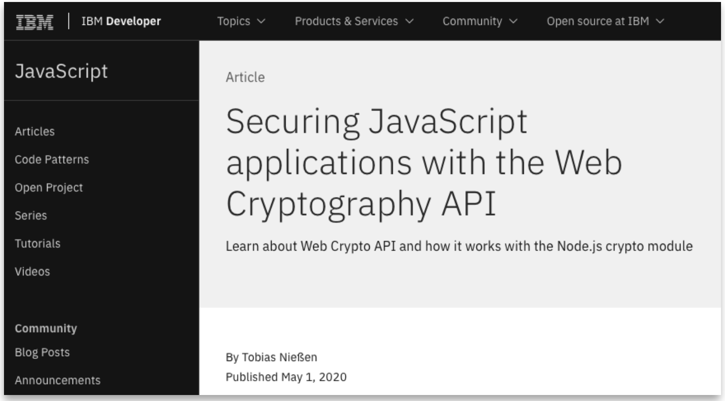
http://qnimate.com/post-series/web-cryptography-api-tutorial/

Security Hints / Outlook

(a) Many APIs are restricted to **Secure Contexts**

(b) WebCrypto and **Node.js**

(a)



Article

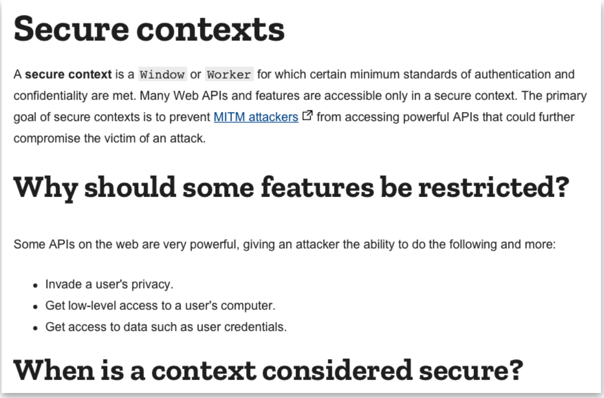
Securing JavaScript applications with the Web Cryptography API

Learn about Web Crypto API and how it works with the Node.js crypto module

By Tobias Nießen
Published May 1, 2020

https://developer.ibm.com/articles/secure-javascript-applications-with-web-crypto-api/

(b)



Secure contexts

A **secure context** is a `Window` or `Worker` for which certain minimum standards of authentication and confidentiality are met. Many Web APIs and features are accessible only in a secure context. The primary goal of secure contexts is to prevent MITM attackers from accessing powerful APIs that could further compromise the victim of an attack.

Why should some features be restricted?

Some APIs on the web are very powerful, giving an attacker the ability to do the following and more:

- Invade a user's privacy.
- Get low-level access to a user's computer.
- Get access to data such as user credentials.

When is a context considered secure?

https://developer.mozilla.org/en-US/docs/Web/Security/Secure_Contexts