Zentrum für sichere Informationstechnologie – Austria
Secure Information Technology Center – Austria

A-1030 Wien, Seidlgasse 22 / 9
Tel.: (+43 1) 503 19 63–0
Fax: (+43 1) 503 19 63–66

A-8010 Graz, Inffeldgasse 16a
Tel.: (+43 316) 873-5514
Fax: (+43 316) 873-5520

http://www.a-sit.at
E-Mail: office@a-sit.at

DVR: 1035461 ZVR: 948166612 UID: ATU60778947

KRYPTOGRAPHIE PLUGIN FÜR CROSS-PLATTFORM APPLIKATIONEN

Abschlussbericht, Version 13.07.2015

Sandra Kreuzhuber – sandra.kreuzhuber@iaik.tugraz.at

Zusammenfassung: In diesem Projekt wurde ein Android Plugin für das Cross-Plattform Framework Apache Cordova implementiert. Dieses Plugin bietet kryptographische Funktionen über eine zur Web Crypto API idente Schnittstelle an. Schlüsselmaterial wird dabei im Schlüsselspeicher des mobilen Gerätes abgelegt. Da der Schlüsselspeicher bei vielen Android Geräten unter Verwendung eines Secure Element realisiert wurde, ergibt sich somit ein besserer Schutz für Schlüsselmaterial gegen Softwareangriffe.

Inhaltsverzeichnis

1. Einleitung

Die Entwicklung mobiler Applikationen unterscheidet sich stark zwischen den unterschiedlichen Plattformen. Diese Unterschiede erfordern eine Neuentwicklung der Applikation für jede zu unterstützende Plattform. Um mit minimalem Aufwand möglichst viele Plattformen abzudecken, gibt es mehrere sogenannte Cross-Plattform Frameworks. Apache Cordova[1] beispielsweise verfolgt einen „write once run anywhere"-Ansatz, wobei die Applikationslogik der mobilen Applikation in HTML und JavaScript implementiert wird. Eine Cordova Applikation wird somit prinzipiell als Web Anwendung implementiert und in einer Web View ausgeführt. Eine Web View stellt ein in eine mobile Applikation integriertes Browser Fenster dar. Damit stehen einer Cordova Applikation alle im Browser zur Verfügung stehenden Funktionen zur Verfügung. Um jedoch auch auf Funktionen der darunterliegenden Plattform zugreifen zu können, bietet Cordova *Plugins* an. Über Plugins kann beispielsweise der Zugriff auf das Dateisystem, Kontakte, NFC, die Kamera u.v.m. realisiert werden.

Für die Entwicklung sicherheitskritischer Cross-Plattform Anwendungen werden kryptographische Funktionen zum Schutz von Daten am Gerät oder zur Sicherung von Nachrichten benötigt. Da verfügbare JavaScript Kryptographie Bibliotheken jedoch nur einen sehr eingeschränkten Funktionsumfang bieten, wurde in diesem Projekt ein Cordova Kryptographie Plugin umgesetzt. Dabei werden die auf der jeweiligen Plattform verfügbaren kryptographischen Funktionen über

[1] http://cordova.apache.org/

eine an die Web Crypto API[2] angelehnte JavaScript Schnittstelle angesprochen. Die Web Crypto API ist bislang der einzige Ansatz, Zugriff auf kryptographische Funktionen im Browser zu vereinheitlichen. Aktuell wird die Web Crypto API zumindest teilweise von allen gängigen Browsern unterstützt.

Mehrere mobile Plattformen bieten zusätzlich Möglichkeiten zur gesicherten Ablage von Schlüsselmaterial. Bei iOS und bei mehreren Android Geräten erfolgt die Schlüsselablage durch Verwendung eines Secure Elements und bietet daher Schutz vor Softwareangriffen auf das Schlüsselmaterial. Bei Verwendung der Web Crypto API hingegen werden kryptographische Schlüssel in einer Datenbank im Browser abgelegt.

Neben der Implementierung von kryptographischen Funktionen soll Cordova Applikationen die Verwendung des Hardwaregestützten Schlüsselspeichers ermöglicht werden. Damit stehen geheime Schlüssel der Applikation oder dem Browser nicht im Rohformat zur Verfügung, sondern können nur über ein „Keyhandle“ angesprochen werden. Die Speicherung der Schlüssel erfolgt im Schlüsselspeicher des mobilen Geräts. Da weder JavaScript Bibliotheken noch die Web Crypto API auf die rohen Schlüssel zugreifen kann, ergibt sich die Notwendigkeit einer Implementierung der kryptographischen Funktionen in Form eines Cordova Plugins.

Im nächsten Abschnitt wird sowohl die Web Crypto API als auch das Cross-Plattform Framework Apache Cordova näher vorgestellt. Anschließend erfolgt eine kurze Beschreibung der implementierten Funktionen.

2. Hintergrund

2.1. Web Crypto API

Ziel der W3C ist es, mit der Web Crypto API die Verwendung von kryptographischen Funktionen im Browser zu standardisieren. Die Web Crypto API bietet eine Schnittstelle für grundlegende kryptographische Funktionen. Die verfügbaren Funktionen sind im Namespace *window.crypto.subtle* verfügbar. Die W3C wählte den Namespace „subtle“ um aufzuzeigen, dass Entwicklerinnen und Entwickler zur Verwendung der Web Crypto API über kryptographisches Vorwissen verfügen sollen. Zusätzlich zur Web Crypto API existiert ein Entwurf für eine Highlevel Crypto API[3], die Entwicklerinnen und Entwickler lediglich die Funktionen *seal/open* und *encryptAndSign/decryptAndVerify* zur Verfügung stellt und dabei auf vordefinierte Standardwerte in Bezug auf Schlüssellängen, Algorithmen und Modi zurückgreift. Dabei können die beiden Anwendungsszenarien a) Absicherung abgelegter Daten und b) Absicherung von Nachrichtenaustausch realisiert werden.

Vor Umsetzung der Web Crypto API stand JavaScript Entwicklerinnen und Entwicklern kein kryptographisch starker Zufallszahlengenerator zur Verfügung. Die Web Crypto API definiert daher die Funktion *getRandomValues(…)* und empfiehlt Browser Herstellern dabei auf Betriebssystemfunktionen zurückzugreifen.

Kryptographische Schlüssel werden in der Web Crypto API als CryptoKey Objekte dargestellt. CryptoKey Objekte können über die in der API definierten Funktionen *generateKey*(…) und *importKey(…)* erzeugt werden. CryptoKey Objekte stellen ein KeyHandle dar. Das rohe Schlüsselmaterial ist vor Zugriffen von Web Anwendungen geschützt. Die Web Crypto API bietet per se keine persistente Speicherung von Schlüsselmaterial. Sobald das CryptoKey Objekt zerstört wurde, kann nicht mehr auf das Schlüsselmaterial zugegriffen werden. Entwicklerinnen und Entwickler können CryptoKey Objekte jedoch persistent in der vom Browser angebotenen IndexedDB Datenbank ablegen[4]. Die Web Crypto API unterstützt die Formate „jwk“, „spki“, „pkcs8“

[2] http://www.w3.org/TR/WebCryptoAPI
[3] https://dvcs.w3.org/hg/webcrypto-highlevel/raw-file/tip/Overview.html
[4] Obwohl das Schlüsselmaterial des CryptoKey Objektes im Browser unter Verwendung von JavaScript nicht ausgelesen werden kann, befindet sich das Schlüsselmaterial bei Speicherung des CryptoKeys in die

und „raw“ für den Import und Export von Schlüsselmaterial. Beim Import oder der Generierung von Schlüsselmaterial werden sogenannte Key Usages definiert („encrypt“, „decrypt“, „sign“, „deriveBits“ etc.) und bei jeder nachfolgenden Operation mit diesem CryptoKey Objekt überprüft. Des Weiteren spezifiziert die Entwicklerin bzw. der Entwickler, ob das rohe Schlüsselmaterial des CryptoKey Objektes extrahierbar sein soll, d.h. der Schlüssel über die *exportKey(…)* Funktion exportiert werden kann. Implementierungen der Web Crypto API haben sicher zu stellen, dass CryptoKey Objekte der Same Origin Policy unterliegen und damit Web Anwendungen keinen Zugriff auf nicht von dieser generierte oder importierte Schlüssel hat.

2.2. Apache Cordova

Apache Cordova ermöglicht die Entwicklung von nativen mobilen Applikationen mit Webtechnologien. Dabei werden HTML, JavaScript und CSS Dateien in eine native Applikation verpackt. Cordova Applikationen können wie herkömmliche Applikationen über den Applikationsstore des jeweiligen Herstellers installiert werden. Jede Cordova Applikation wird prinzipiell als Webanwendung implementiert und in einer Web View dargestellt. Eine Web View ist ein in eine Applikation eingebettetes Browser Fenster. Cordova Applikationen können zusätzlich zu den im Browser verfügbaren Funktionen auch auf APIs der darunterliegenden Plattform über sogenannte Plugins zugreifen. Plugins werden für jede Plattform separat in der nativen Entwicklungssprache entwickelt. Daraus resultiert, dass nur der in Webtechnologien implementierte Teil einer Cordova Applikation auf jeder Plattform ohne Modifikationen ausgeführt werden kann. Sobald ein Zugriff auf Plattformfunktionen erfolgt, muss ein Plugin für jede zu unterstützende Plattform umgesetzt werden. Für viele Funktionen wie z.B. Zugriff auf NFC, die Kamera, das Dateisystem[5] etc. stehen bereits Plugins zur Verfügung und können in neue Applikationen integriert werden.

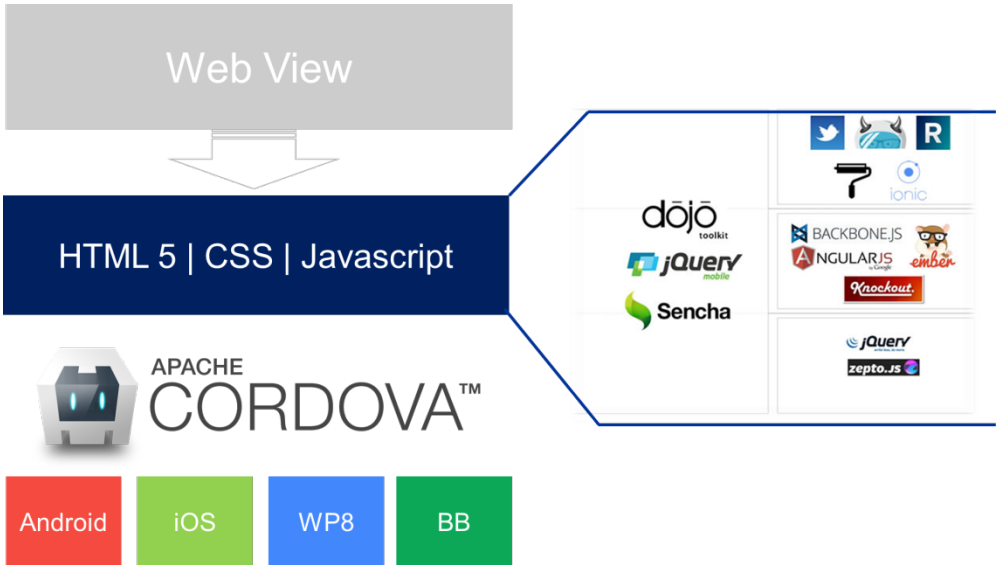

Abbildung 1: Apache Cordova Übersicht

Abbildung 1 bietet einen Überblick über die eingesetzten Komponenten bei mit Apache Cordova umgesetzten Cross-Plattform Anwendungen. Dabei kann ein Cordova Plugin als Javascript Bibliothek, die die im Browser verfügbaren Funktionen ergänzt, angesehen werden. Die Javascript Schnittstelle des Plugins wird wie eine herkömmliche JavaScript Bibliothek in die Applikation inkludiert. Die Bibliothek übernimmt die Kommunikation mit dem nativen Code des Plugins und retourniert das Resultat an die Anwendung.

IndexedDB im Klartext in der dazugehörigen Datei. Das bedeutet, dass bei persistenter Schlüsselspeicherung, Angreifer, die Zugriff auf das Dateisystem besitzen, das geheime Schlüsselmaterial auslesen können. Durch den Import des ausgelesenen Bytestrings als CryptoKey Objekt können zuvor damit verschlüsselte Daten entschlüsselt werden.

[5] http://plugins.cordova.io

Die folgende Grafik stellt den Datenfluss im entwickelten Kryptographie Plugin dar. Dabei erfolgt zu Beginn der Aufruf einer vom Plugin zur Verfügung gestellten JavaScript Funktion. Dieser Funktionsaufruf erfolgt ident zum Aufruf der im Browser implementierten Web Crypto API. Innerhalb der JavaScript Funktion werden die Parameter überprüft und gegebenfalls neu kodiert. Anschließend werden die Parameter zusammen mit einem Identifikator des gewünschten Plugins an die *cordova.exec(…)* Funktion übergeben. Diese Funktion startet die Ausführung im nativen Teil des Plugins. Nach Abarbeitung des Befehls im Plugin (in diesem Fall nach Verschlüsselung der übergebenen Daten) wird das Ergebnis an eine Callback-Funktion im JavaScript Teil des Plugins übergeben. Alle Aufrufe von Cordova Plugins erfolgen asynchron über Callback-Funktionen. Um mit der Web Crypto API konform zu bleiben, werden vom JavaScript Teil des Plugins sogenannte *Promise*[6] Objekte an die Applikation retourniert. Sobald die Callback-Funktion im JavaScript Teil des Plugins aufgerufen wird, erfolgt die Rückgabe des Resultats an die Applikation über *Promise.resolve(…).*

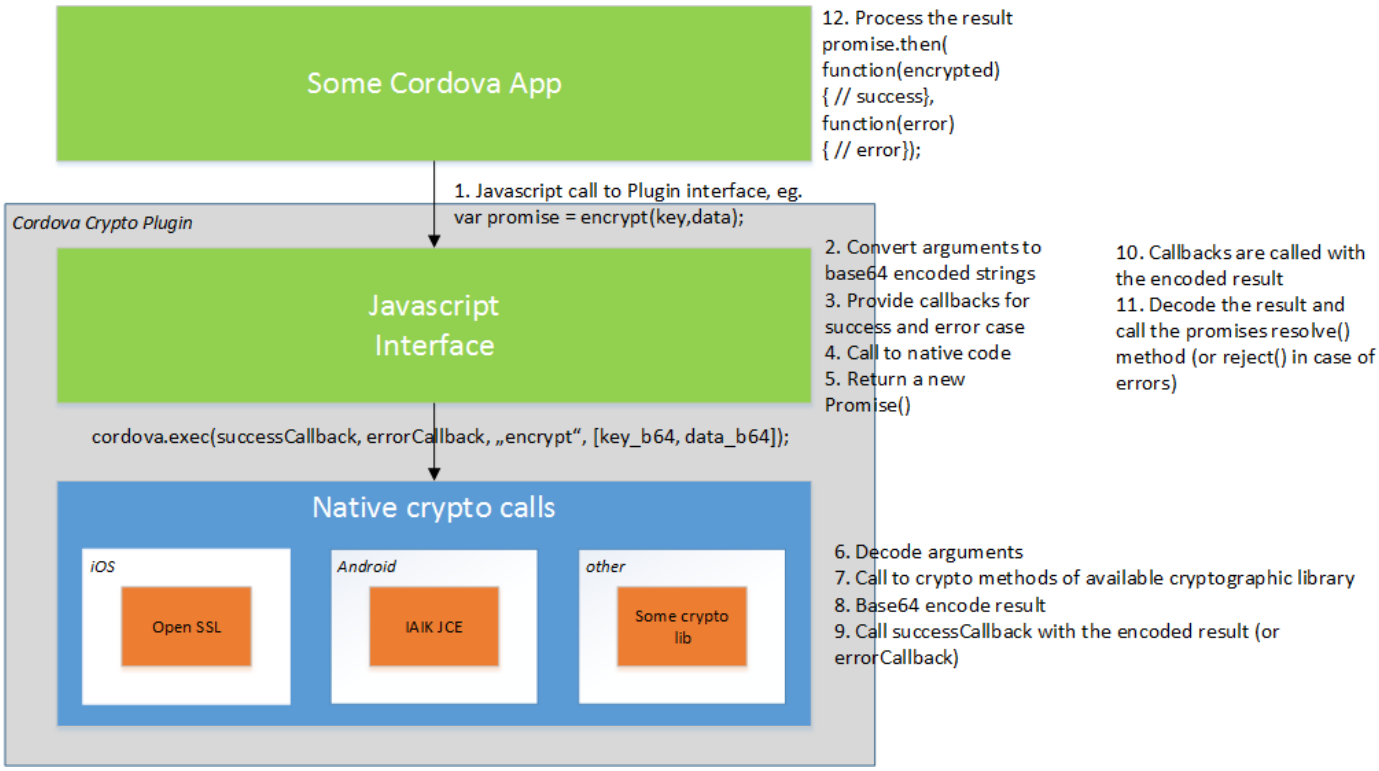

Abbildung 2: Prozessfluss Kryptographie Plugin

3. Umgesetzte Funktionen

Bei der Implementierung des Kryptographie Plugins wurde der Fokus auf Funktionen gelegt, die eine Ablage von Schlüsselmaterial erfordern. Die Berechnung von Hashwerten beispielsweise wurde nicht im Kryptographie Plugin umgesetzt, da dazu im Browser zur Verfügung stehende Funktionen verwendet werden können.

Um die Verwendung der kryptographischen Funktionen mit der Web Crypto API zu vereinheitlichen, wurde eine Provider Struktur eingeführt. Dabei kann die jeweilige Implementierung der Web Crypto API anhand eines Provider Namens instanziert werden. Dadurch können parallel sowohl Funktionen der im Browser implementierten Web Crypto API als auch des Kryptographie Plugins innerhalb einer Applikation verwendet werden. Aktuell werden zwei Provider unterstützt: *cordova-iaik* und *w3c*. Die Instanzierung eines Providers erfolgt durch folgenden Aufruf:

[6] http://www.html5rocks.com/en/tutorials/es6/promises/

var cordovaCryptoApi = window.getCryptoProviderByName("cordova-iaik");

bzw.

var w3cCryptoApi = window.getCryptoProviderByName("w3c");

Die Web Crypto API gibt das Format von *CryptoKey* Objekten vor, die als Keyhandle für das eigentliche Schlüsselmaterial dienen. Dieses CryptoKey Objekt wurde um das Feld *Id* erweitert. Der darin gesetzte Identifikator wird benötigt, um die in JavaScript zur Verfügung stehenden Keyhandle mit dem in der KeyChain abgelegten Schlüsselmaterial zu assoziieren. Die W3C veröffentlichte 2012 einen Working Draft zu Key Discovery[7]. Damit sollen Schlüssel, die nicht über die Web Crypto API erzeugt oder importiert wurden, eingebunden werden. Die Erweiterung des CryptoKey um das *Id* Feld, ist konform zu der im Working Draft vorgeschlagenen Erweiterung um *name* und *id*.

Das Kryptographie Plugin steht aktuell nur für die Android Plattform zur Verfügung. Für die native Implementierung der kryptographischen Methoden wurden die vom IAIK angebotenen Kryptographie Bibliotheken IAIK-JCE[8] und ECCelerate[9] verwendet.

Zusätzlich wurde die Web Crypto API Schnittstelle um den SCrypt Algorithmus[10] zur Ableitung kryptographischer Schlüssel aus einem Passwort erweitert. Dabei wird aktuell eine von BouncyCastle[11] angebotene Implementierung des SCrypto Algorithmus verwendet.

Nachfolgende Tabelle, bietet eine Übersicht über die im Kryptographie Plugin implementierten Methoden.

---

[7] https://dvcs.w3.org/hg/webcrypto-keydiscovery/raw-file/440cee6b9ab3/Overview.html
[8] https://jce.iaik.tugraz.at/sic/Products/Core_Crypto_Toolkits/JCA_JCE
[9] https://jce.iaik.tugraz.at/sic/Products/Core-Crypto-Toolkits/ECCelerate
[10] https://tools.ietf.org/html/draft-josefsson-scrypt-kdf-02
[11] http://www.bouncycastle.org/docs/docs1.5on/org/bouncycastle/crypto/generators/SCrypt.html

Algorithmus	Encrypt	Decrypt	Sign	Verify	Digest	generateKey	deriveKey	deriveBits	importKey	exportKey	wrapKey	unwrapKey
AES-CTR	x[12]	x				x			x	x		
AES-CBC	x	x				x			x	x		
AES-GCM	x	x				x			x	x		
AES-CFB	x	x				x			x	x		
AES-CMAC			x	x		x			x	x		
AES-KW	Aktuell nicht unterstützt.											
HMAC			x	x		x			x	x		
SHA-1 SHA-256 SHA-384 SHA-512	Nicht unterstützt, da bei diesen Methoden keine sichere Speicherung von Schlüsselmaterial erfolgen muss. Empfohlen wird die Verwendung der Web Crypto API von Chrome auf Android.											
DH	Aufgrund fehlender RSA Implementierung aktuell nicht unterstützt.											
ECDH[13]						x	x	x	x	x		
ECDSA			x	x		x			x	x		
RSASSA-PKCS1-v1_5 RSA-PSS RSA-OAEP	Durch Studentenprojekte abgedeckt. Methoden wurden noch nicht im Kryptographie Plugin integriert.											
CONCAT	Aktuell nicht unterstützt.											
HKDF-CTR	Aktuell nicht unterstützt.											
PBKDF2	Aktuell nicht unterstützt.											
SCRYPT							x	x	x[14]			
getRandomValues	Unterstützt. Resultat wird in als Promise Objekt retourniert.											

[12] Parameter „length“ wird aktuell nicht unterstützt. 16 Bytes IV wird direkt an IAIK-JCE übergeben.
[13] Import und Export bei ECDH und ECDSA unterstützen nur das Schlüsselformat „jwk“. Pkcs8 und SPKI werden nicht unterstützt.
[14] Die SCrypt Implementierung erfordert ein Importieren des zu verwendenden Passworts über importKey. In der Web Crypto API ist jedoch auch eine im Browser integrierte Passworteingabe vorgesehen, die die Benutzerin bzw. den Benutzer bei Aufruf von generateKey nach dem Passwort fragt.

Um eine Interoperabilität des Kryptographie Plugins mit der im Browser zur Verfügung stehenden Web Crypto API zu gewährleisten, wurde eine Demo Applikation bereitgestellt. Diese Demo Applikation integriert vom Chromium Projekt zur Verfügung gestellte Testfälle[15] und vergleicht die von der jeweiligen Implementierung der Web Crypto API erhaltenen Resultate mit Referenzwerten.

4. Lizenz

Dieses Projekt ist unter EUPL Version 1.1 lizenziert. Beim Cordova Kryptographie Plugin kommen die IAIK-Toolkits IAIK-Java Cryptography Extension[16] und IAIK ECC[17] zum Einsatz. Diese Komponenten sind im kommerziellen Umfeld kostenpflichtig, für Forschung und Ausbildung sind kostenlose Lizenzen[18] verfügbar. Eine Evaluierungsversion der beiden Bibliotheken wurde im Ordner *cordova-crypto-plugin/src/android/libs* beigelegt.
Für die SCrypt Schlüsselableitungsfunktion wurde BouncyCastle verwendet. Die Java Bibliothek BouncyCastle ist unter der MIT Lizenz lizenziert und wurde im Ordner *cordova-crypto-plugin/src/android/libs* beigelegt.

5. Future Work

Aktuell wurde das Kryptographie Plugin lediglich für die Android Plattform entwickelt. Eine Ausdehnung auf iOS und Windows Phone fehlt jedoch. Des Weiteren wurden RSA Operationen nicht implementiert, da diese im Rahmen eines Studentenprojektes umgesetzt, jedoch noch nicht integriert wurden. Als Erweiterung wäre eine Integration von High Level Kryptographischen Methoden wie beispielsweise CMS sinnvoll.

6. Ressourcen

- Web Crypto API allgemein
  - W3C Candidate Recommendation Web Crypto API http://www.w3.org/TR/WebCryptoAPI/
  - Mozilla Dokumentation der Web Crypto API: https://developer.mozilla.org/en-US/docs/Web/API/SubtleCrypto
  - Hintergrundinformationen und Motivation für die Einführung der Web Crypto API: http://www.ibiblio.org/hhalpin/homepage/presentations/webcrypto/Overview.html
  - Aktuelle Umsetzung der Web Crypto API in gängigen Browsern: http://caniuse.com/#feat=cryptography
  - Firefox Status zur Umsetzung der Web Crypto API: https://docs.google.com/spreadsheet/ccc?key=0AiAcidBZRLxndE9LWEs2R1oxZ0xidUVoU3FQbFFobkE#gid=1
  - Chromium Support für die Web Crypto API: http://www.chromium.org/blink/webcrypto
  - Internet Explorer Support für die Web Crypto API: https://msdn.microsoft.com/en-us/library/dn302338(v=vs.85).aspx
- Beispiele zur Verwendung der Web Crypto API
  - http://blog.engelke.com/
  - https://github.com/diafygi/webcrypto-examples

[15] https://chromium.googlesource.com/chromium/blink/+/master/LayoutTests/crypto/
[16] https://jce.iaik.tugraz.at/sic/Products/Core_Crypto_Toolkits/JCA_JCE
[17] https://jce.iaik.tugraz.at/sic/Products/Core-Crypto-Toolkits/ECCelerate
[18] https://jce.iaik.tugraz.at/sic/Sales/Licences