6 October 2014 Study · Web & Cloud · Privacy & Crypto

Create and decode CMS in Javascript

The full text is currently only available in the Deutsch version. A translation will follow.

Archived. This content is no longer updated. It remains available for reference.

Source https://technology2.a-sit.at/en/publications/create-and-decode-cms-in-javascript/

This project implements a demo that provides encryption and decryption in a browser using CMS. As the report of this project is available in German only, this article covers the most relevant results.

In order to process sensitive data in a browser-based application, several cryptographic functions have to be implemented by developers. Currently basic functions as RSA or AES encryption are covered by publicly available Javascript libraries, but they lack the support for advanced mechanisms as Cryptographic Message Syntax or XML signatures.

This project implements a demo for using CMS functionality in Javascript. The following steps are currently supported:

- Creation of CMS-Container
- Creation of the random AES key and AES encryption of data in the browser
- Creation of the ASN.1 structure of the CMS container
- Parsing a CMS-Container
- Parse the ASN.1 structure of the CMS container
- Extraction of the encrypted Content Encryption Key
- Extraction of the encrypted data, the algorithm in use etc.
- Decryption of the encrypted data with the encrypted Content Encryption Key

In order to encrypt and decrypt the Content Encryption Key we provide a simple Java application that can be used on the server side. To encrypt or decrypt the Content Encryption Key, the Base64 encoded key has to be sent from the Javascript application to the Java application.

Our code is based on the following publicly available Javascript cryptography libraries:

- **CryptoJS** provides AES and DES encryption and support for the computation of HMAC-Codes. Further CryptoJS provides PBKDF2 for key derivation and offers multiple Hash function. In this project we use CryptoJS for encrypting and decrypting the data to be stored in the CMS container.
- **jsrsasign** is used for creating the ASN.1 structure of the CMS container. In addition jsrsasign supports parsing of X.509 certificates and thereby simplifies the creation of the ASN.1 structure *IssuerAndSerialNumber*.
- **asn1js** offers an ASN.1 parser for decoding DER oder BER encoded ASN.1 data structures. This project uses asn1js for parsing CMS containers.
- **jQuery** is used for DOM manipulation within the demo application. The jQuery Plugin **jQuery Base64** is used for encoding and decoding Base64 data.

Limitations

The developed proof-of-concept application is not a complete implementation of the CMS standard. Below you can find several limitations of our implementation:

- We currently only support the data type EnvelopedData combined with KeyTransRecipientInfos.
- Further only one RecipientInfo for each CMS container is supported. Our implementation can be easily extended to support multiple RecipientInfo data structures.
- We use AES in CBC mode to encrypt the data. Our implementation can be easily extended to support further encryption algorithms and modes.
- The Content Encryption Key is encrypted in a separate Java application. Therefore the encrypted key and the decrypted result have to be exchanged manually between the Java application and the web application. In order to use this component on the server-side (e.g. when using mechanisms for central key storage) a transport mechanism and methods for matching the right key with regard to the Recipient Info data structure have to be implemented.
- Currently we use Math.rand() as used in CryptoJS for generating random encryption keys. However, Math.random() cannot be considered as a cryptographic strong Random Number Generator. If the implemented components are to be used for protecting critical data in a real world scenario, alternatives for Math.random() have to be integrated. Possible alternatives are provided by W3C Crypto API, Clipperz or SJCL.

Links

It has to be noted, that using cryptography in the web browser raises some security concerns. For more information we recommend the following articles:

- http://matasano.com/articles/javascript-cryptography/ (EN)
- http://tonyarcieri.com/whats-wrong-with-webcrypto (EN)