Required modifications to CDOC for elliptic curve support

Document no.: A-101-7
Version 1.1
27.09.2017

Introduction

This document describes changes for CDOC format, which enable confidentiality with the help of Elliptic Curve Cryptography.

Currently CDOC is based on [XMLENC] standard. It is set to

- data encryption using algorithm and mode (AES128-CBC) and
- transport key encryption using algorithm (RSA PKCS#1 v1.5).

The ECC support requires to use [XMLENC1] standard, which defines key exchange primitive ECDH. In the newer version of the standard it is possible to use also more secure encryption algorithms.

The shortcomings of the current CDOC version are described in [CDOC10]. Specification [CDOC20] describes possible improvements for these shortcomings, however, the required modifications are large therefore affect application structure significantly.

The changes described in the current document are small therefore can be faster supported. From the changes described in document [CDOC10] will be taken only encryption algorithms and related modes (instead of `AES128-CBC` use `AES256-GCM`)

Backward Compatibility

New and old CDOC version software implementation rules are the following.

- New software will support reading of files created with old version.
- New software creates only files, which old version cannot read, ECC as well as RSA key based. We assume, that most of the users open encrypted files with the help of base software, therefore there should not accur big problems after base software updating to read files.

In case if encrypted file opening is integrated in some other compiled software package, will have to create new version of this software package also.

In case if encrypted file creation is integrated in some other compiled software package, then without its updating encrypted files can be sent only to recipients, who have old ID card with RSA keys.

Data Encryption

For file encryption is created one random symmetric key. For encryption is used AES256-GCM algorithm [SP800-38D]. Encryption and decryption is perform the way as described in the section 5.2.4 of the document [XMLENC1].

Initialization vector is generated randomly.

The decryption must verify that checksum matches. In case checksum does not match an error is given.

Should be taken into account, that using AES256-GCM algorithm the largest file that can be

encrypted is 2^39-256 bits or almost 64 gigabytes [SP800-38D].

Transport key encryption for RSA certificate owners

For owners of RSA certificates transport key is encrypted with recipients public key, using PKCS#1 v1.5 padding scheme. The result is `EncryptedKey` element, where `EncryptionMethod` element denotes encryption algorithm `http://www.w3.org/2001/04/xmlenc#rsa-1_5`.

Against the padding scheme used in PKCS#1 v1.5 is known the so-called Bleichenbacher attack [Bleichenbacher98], whose applicability and optimization posibilities against Estonian ID card in 2012 was analyzed by Bardou et al [BFKSST12].

In this attack the attacker used ID card as an oracle, submitting his decryption queries and received response info about successful decryption (but not necessarily decrypted files).

As a result of acquiring large number of such queries the attacker could decrypt one file, whose query was not among in the originals. Important to note, that the secret key itself with the result of this attack did not leak.

Bardou et al. estimated that with such attack against ID card required 28300 queries, which took in total around 27 hours. Let's take into account, that for successful query sending the attacker needs the PIN1 code also. The same way the attacker having access to ID card and PIN1 code could just decrypt the needed file, which makes the decrypted attack against the ID card in essence worthless.

Transport key encryption for ECC certificate owners

For owners of ECC certificates the transport key is encrypted using the result of ECDH key exchange with received shared secret key.

For ECDH key exchange is used the same keypair as for authentication. Since ECC authentication in turn relies on ECDSA signature scheme, this solution comes with security question, does the use of ECDH and ECDSA together is secure.

This question was studied in 2011 by Degabriele et al [DLPSS11]. They gave ECDH and ECDSA co-use proof in so-called generic attack model, where attacker is allowed to use only elliptic curve point group operations.

Generic attack model is weaker than so-called specific model, where attacker can access also the implementation, can use algebraic properties of the specific curve, etc.

At the same time, the best known attacks against elliptic curve cryptography are not able to exploit the operations beyond these of the generic group. This is among other things also the cause, why crypto primitives relying on elliptic curve allow to use much shorter keys compared to RSA.

At the same time it is worth to note, that also the security of ECDSA signature scheme itself

is proved only in generic attack model [HMV06].

ECC Transport key encryption

Encryption process is the following.

1. Sender takes from recipients certificate his ECC public key `Rp` and if needed also the description of used curve.
2. Sender generates ephemeral ECC private and public key pair `(Ss,Sp)`.
3. Sender calculates (locally) the result of ECDH key exchange operation `Ksr`, using own ephemeral private key `Ss` and recipients public key `Rp`.
4. Sender derives from the shared secret `Ksr` transport key an encryption key. For key derivation is used algorithm `http://www.w3.org/2009/xmlenc11#ConcatKDF`. Key derivation algorithm principles of operation and parameter coosing logic is explained in document [SP800-56Ar2] section 5.8 and appendix B. In key derivation linked derived key
   - identifier for the used algorithm (i.e., algorithm used to encrypt transport key `http://www.w3.org/2001/04/xmlenc#kw-aes256`)
   - recepient's identificator, which is in the same form as `EncryptedKey` element `Recipient` atribute value (for example, "TAMM,JAAN,37701023333,ID-CARD")
   - sender's ephemeral key, which is in the same form as in the `OriginatorKeyInfo` element included key.
5. Sender encrypts transport key with transport key encryption key (*key wrap*). See [XMLENC1] section 5.7.
6. Sender forms `EncryptedKey` element, whose subelements are following.
   - `EncryptionMethod` encryption algorithm is `http://www.w3.org/2001/04/xmlenc#kw-aes256`.
   - `CipherValue` is with shared secret protected transport key.
   - In the `KeyInfo` there is `AgreementMethod` element, where
     - `Algorithm` atribute denotes algorithm `http://www.w3.org/2009/xmlenc11#ECDH-ES` and
     - `xenc11:KeyDerivationMethod` subelement denotes key derivation algorithm `http://www.w3.org/2009/xmlenc11#ConcatKDF`.
       - `xenc11:ConcatKDFParams` subelement marks key deerivation algorithm parameters.
         - Hash function is `http://www.w3.org/2001/04/xmlenc#sha512`.
         - `AlgorithmID` atribute value is encryption algorithm value, for what key generated, possibly then `http://www.w3.org/2001/04/xmlenc#kw-aes256`.
         - `PartyUInfo` atribute value is senders public key value in the same form as it is in `OriginatorKeyInfo`.
         - `PartyVInfo` atribute value is recipients identity in the same form as in `EncryptedKey` element `Recipient` atribute.
     - Under `OriginatorKeyInfo` there is subelement `dsig11:ECKeyValue`

element, where as a value is senders generated ephemeral ECC keypair public key `Sp`.

- Under `RecipientKeyInfo` subelement there is subelement `ds:X509Data`, what denotes senders public key, for which shared secret is created. It is possible, that here can use whole client certificate, like so far is done.

See also [XMLENC1] EXAMPLE 42.

ECC Transport key decryption

Decryption process is the following:

1. Recipient finds `Recipient` atribute which is meant for him in the `EncryptedKey` element.
2. Recipient verifies that it understands and supports all on the sender's side used algorithms and parameters.
   - Data encryption used algorithm `http://www.w3.org/2009/xmlenc11#aes256-gcm`.
   - `EncryptionMethod` element denotes key encryption algorithm `http://www.w3.org/2001/04/xmlenc#kw-aes256`.
   - `AgreementMethod` element algorithm is `http://www.w3.org/2009/xmlenc11#ECDH-ES`.
   - `xenc11:KeyDerivationMethod` subelement denotes key derivation algorithm `http://www.w3.org/2009/xmlenc11#ConcatKDF`.
   - `xenc11:ConcatKDFParams` hash function is `http://www.w3.org/2001/04/xmlenc#sha512`.
   - `AlgorithmID` atribute value matches transport key encryption used algorithm.
   - `PartyUInfo` atribute value matches with sender public key in `OriginatorKeyInfo`.
   - `PartyVInfo` atribute value matches with recipients identity in `EncryptedKey` elements `Recipient` atribute.
   - In the `RecipientKeyInfo` subelement contained `ds:X509Data` subelement is identified recipients authentication certificate, for what he has private key.
3. Recipient takes `OriginatorKeyInfo` subelement `dsig11:ECKeyValue` senders ephemeral public ECC key `Sp`.
4. Recipient performs in his ID card ECDH key exchange, using private authentication key `Rs` in the card and senders public key `Sp`. Key exchange result is shared secret `Ksr`.
5. Recipient brings the shared secret `Ksr` to transport key encryption key. Sender must have used `http://www.w3.org/2009/xmlenc11#ConcatKDF` algorithm. Algorithm parameter values are the following:
   - `AlgorithmID` atribute value is encryption algorithm value, for which the key was generated, so `http://www.w3.org/2001/04/xmlenc#kw-aes256`.
   - `PartyUInfo` atribute value is senders public key in the same forn as in the `OriginatorKeyInfo`.
   - `PartyVInfo` atribute value is recipients identity in the same form as in the element's

`EncryptedKey` atribute `Recipient`.

6. Recipient uses the derived key data to decrypt transport key. Sender must have used `http://www.w3.org/2001/04/xmlenc#kw-aes256` algorithm.
7. With the obtained key is decrypted recipients data, using algorithm `http://www.w3.org/2009/xmlenc11#aes256-gcm`. Among other things recipient verifies checksum of calculated GMAC authentication code. If it does not match, the cryptograms transport integrity has not been preserved and as a result is displayed corresponding error message.

References

- **[XMLENC]** "XML Encryption Syntax and Processing", https://www.w3.org/TR/xmlenc-core/
- **[XMLENC1]** "XML Encryption Syntax and Processing Version 1.1", https://www.w3.org/TR/xmlenc-core1/
- **[CDOC10]** "CDOC 1.0 Notes and caveats", https://github.com/martinpaljak/idcrypt/wiki/CDOC-1.0
- **[CDOC20]** "CDOCv2 DRAFT v0.1", https://github.com/martinpaljak/idcrypt/wiki/CDOC-2.0
- **[SP800-56Ar2]** "NIST Special Publication 800-56A Revision 2. Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography", http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-56Ar2.pdf
- **[SP800-38D]** "NIST Special Publication 800-38D. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC", http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38d.pdf
- **[Bleichenbacher98]** Daniel Bleichenbacher: Chosen Ciphertext Attacks against Protocols Based on the RSA Encryption Standard PKCS #1. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 1–12. Springer, Heidelberg (1998).
- **[BFKSST12]** Romain Bardou, Riccardo Focardi, Yusuke Kawamoto, Lorenzo Simionato, Graham Steel, Joe-Kai Tsay: Efficient Padding Oracle Attacks on Cryptographic Hardware. In CRYPTO 2012. LNCS, vol. 7417, pp. 608-625. Springer, Heidelberg (2012).
- **[DLPSS11]** Jean Paul Degabriele, Anja Lehmann, Kenneth G. Paterson, Nigel P. Smart, and Mario Strefler. "On the joint security of encryption and signature in EMV." Cryptology ePrint Archive: Report 2011/615, https://eprint.iacr.org/2011/615
- **[HMV06]** Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone. Guide to elliptic curve cryptography. Springer Science & Business Media, 2006.