CYBERNETICA
Institute of Information Security

Profile for High-Performance Digital Signatures

Version 1.2

Margus Freudenthal

T-4-23 / 2017

Cybernetica research reports are available online at
`http://research.cyber.ee/`

Mailing address:
Cybernetica AS
Mäealuse 2/1
12618 Tallinn
Estonia

Profile for High-Performance Digital Signatures

Margus Freudenthal

Version 1.2

Abstract

Digital signatures and timestamps are a good way to prove integrity of messages exchanged between parties. However, a digital signature implementation may encounter performance, latency and reliability issues unless certain care is taken when designing a solution for efficient use of Public Key Infrastructure protocols.

This specification profiles XAdES and ASiC specifications to describe a signature scheme that is more scalable than the naive implementation. This signature scheme is used by Cybernetica's Unified eXchange Platform and the systems based on it such as the Estonian X-Road.

Contents

1 Introduction

1.1 General

Employing digital signatures and timestamping in a system that has high scalability and availability requirements is often not straightforward. The reason for this is that the standard PKI protocols (such as OCSP and timestamping protocol) are not designed for scalability and high performance. These issues are discussed by Ansper et al. in [1]. Whereas the referenced paper explains the problem and a general solution, this specification describes technical protocols that implement this solution. The technical protocol is employed in Cybernetica's Unified eXchange Platform[1] (UXP) and the Estonian X-Road system.

The signed documents are based on the Associated Signature Containers (ASiC) format [2]. For signatures, we use the XML Advanced Electronic Signature (XAdES) [8] format and for timestamps we use the IETF timestamping protocol [7]. For certificate validity information, Online Certificate Status Protocol (OCSP) [4] is used. Batch hashing scheme [3] is used to implement batch signatures and batch timestamps.

In UXP, parties communicate by invoking SOAP [5] services. The communicating parties install components called security servers that act as gateways by encrypting, signing, archiving and timestamping the exchanged SOAP messages. Security servers make the archived and timestamped messages available as ASiC containers that can be processed and verified off-line. The container contains message and all the necessary supporting information, such as signature, signing certificate, OCSP response(s) and timestamp.

In general, the ASiC container created by a security server is a self-contained item that contains the signed message and all the necessary information needed for verification. However, if SOAP attachments (see [6]) are used then only the SOAP message part of the full SOAP message package is required to be included in the ASiC container. The security servers may, but are not required to include attachments in the container. However, the container contains information that is sufficient to verify that externally stored attachments are indeed signed by the given signature. See Section 3 for details.

1.2 Terms and Abbreviations

ASiC	Associated Signature Container
CA	Certification Authority
HSM	Hardware Security Module

[1] Unified eXchange Platform is a system for connecting different organizations. It is used as a basis for several national data interchange systems such as the Estonian X-Road infrastructure.

MIME	Multipurpose Internet Mail Extensions
OCSP	Online Certificate Status Protocol
SSCD	Secure Signature Creation Device
TSP	Timestamp Provider
TST	Timestamp Token
UXP	Unified eXchange Platform
XAdES	XML Advanced Electronic Signature

1.3 Signature Creation Process

The following is a list of steps that a security server performs when creating a complete signed and timestamped message container.

1. The sender security server receives a SOAP message from the end entity (an information system communicating via the UXP). The message can have (binary) attachments according to the SOAP with attachments specification [6].

2. The sender security server acquires an OCSP response for the signing certificate. The OCSP response can be cached and reused for several messages that are sent during a configured time interval. This reduces the load on the OCSP server.

3. The sender security server signs the message. If the signing load of the security server is greater than the signing speed of the signing device[2] then the sender uses batch signatures to sign a set of messages with one operation. Batch signatures are also used to sign message with attachments.

4. The sender security server transmits to the receiving security server:

   - the SOAP message with attachments;
   - the XAdES signature containing, among other things,
     - signing certificate,
     - certificate chain from the signing certificate to the trusted CA, and
     - OCSP response(s) confirming the validity of the signing certificate and all of the intermediate certificates; and
   - the hash chain result and the hash chain, if the signature is a batch signature.

[2] This can happen when the key is stored in a slow hardware device such as a smart card.

5. The receiver security server verifies the signature. When verifying the OCSP responses, the receiver checks that all the responses are sufficiently fresh (the `producedAt` field indicates time that is not older than the configured threshold).

6. The receiver security server saves the received data items to the message log. The SOAP attachments are not saved in order to keep the size of the message log minimal.

7. After a preconfigured time has passed, the security server timestamps the collection of signatures. It uses batch hashing to create a single timestamping request for all the signatures received within a time interval. The timestamping operation produces a single timestamp token and hash chain result together with one hash chain for every timestamped signature. All these data items are stored in the message log.
   In some configurations, the timestamps are separately requested for every incoming signature. In this case, the batch hashing is not used.

8. When there arises a need to prove reception of the request to a third party, the security server creates a single ASiC container that can be verified without consulting additional data sources. The complete signed document will contain

   - SOAP message;
   - sender's signature on the message;
   - in case batch signing was used, a signed hash chain result and a hash chain proving that the message (and the attachments) participated in signature calculation;
   - timestamp proving that the signature was created before a certain time; and
   - in case batch timestamping was used, a timestamped hash chain result and a hash chain proving that the signature participated in the timestamp request.

2 Document Formats

Signing and timestamping are two separate operations and the decisions to use batch signing and batch timestamping are independent of each other. Therefore this section describes as two separate operations the process of adding a signature to a message and the process of adding a timestamp to a signature. Irrespective of the signing or timestamping method used, the ASiC containers contain files `mimetype` and `META-INF/manifest.xml` containing, respectively the MIME type of the ASiC container and the list of signed data files.

2.1 Adding Signature to a Message

2.1.1 Profile of XAdES Signature

The signature format used in UXP is a profile of the XAdES Basic Electronic Signature form (XAdES-BES, see [8], Section 4.4.1) and of the XAdES Baseline Profile [9] (B-level conformance). If batch signatures and batch timestamps are not used, timestamped signatures conform to XAdES Electronic Signature with Time (XAdES-T) (LT-level conformance in the Baseline Profile). See Section 2.2 for further details.

Table 1 lists the qualifying signed properties and table 2 lists the qualifying unsigned properties that are used in UXP signatures.

2.1.2 Signatures in ASiC Container

The use of signatures in ASiC container depends on whether simple or batch signatures were used.

Simple signatures are used if the SOAP message consists of only the XML part and the speed of the signing device allows creating a separate signature for each incoming message. Table 3 shows contents of the ASiC container for simple signatures.

Batch signatures are used if any of the following two conditions are true.

1. The SOAP message received by the security server contains attachments.
2. The signing device is not sufficiently fast to separately sign each incoming message.

Batch signatures use the batch hashing format described in [3]. Table 4 shows contents of the ASiC container for batch signatures.

SOAP Attachments are not stored in the container. However, they are referenced in the hash chain with `DataRef` element and therefore participate in the signature calculation and can be considered signed. When encountering `DataRef` elements that refer to files not present in the container, a signature verification application can either

- ignore the references, possibly issuing a warning;
- output the referenced files together with the message digest thus allowing the user to verify that the detached file indeed has the given message digest; or
- require that the user retrieve the referenced file and present it for verification.

Table 1: The use of qualifying signed properties in signatures

Element	Comments
`SignedProperties`	
▷`SignedSignatureProperties`	
▷▷`SigningTime`	The time at which the signer (purportedly) performed the signing process. GMT time zone shall be used.
▷▷`SigningCertificate`	Identifier and digest of the signing certificate. The full value of the signing certificate can be found in the `ds:KeyInfo` element.
▷▷`SignaturePolicyIdentifier`	Policy under which the signature can be determined to be valid. Refers to this document.
▷▷▷`SignaturePolicyId`	
▷▷▷▷`SigPolicyId`	Method OIDAsURN is used, value of the OID is `1.3.6.1.4.1.3516.16.2`.
▷▷▷▷`SigPolicyHash`	Hash value of this document.
▷▷▷▷`SigPolicyQualifiers`	
▷▷▷▷▷`SigPolicyQualifier`	
▷▷▷▷▷▷`SPURI`	URL to this document. `https://repo.cyber.ee/dsig-profile-1.2.pdf`
▷`SignedDataObjectProperties`	
▷▷`DataObjectFormat`	This element will describe either a SOAP message (if batch signing was not used) or a hash chain result (if batch signing was used). See Section 2.1.2 for details.

2.2 Adding Timestamp to a Signature

Depending on circumstances, a security server produces either simple or batch timestamps. The default option is to use batch timestamps that significantly reduce the load of the TSP and are less dependent on TSP's availability. This is offset by the risk that a security server fails to timestamp the signature on a received message before the certificate used to sign the message is revoked (and it becomes impossible to prove that the signature was created using a valid certificate). If this risk is not acceptable, the security server can be configured

Table 2: The use of qualifying unsigned properties in signatures

Element	Comments
`UnsignedProperties`	
▷`UnsignedSignatureProperties`	
▷ ▷`SignatureTimeStamp`	This element is present only when batch timestamping is not used. See Section 2.2 for details.
▷ ▷ ▷`EncapsulatedTimeStamp`	The time-stamp token (an ASN.1 data object) generated by the TSP.
▷ ▷`CertificateValues`	Contains full certificate chain from the signing certificate (not included) to a trusted certification authority (included).
▷ ▷`RevocationValues`	
▷ ▷ ▷`OCSPValues`	Contains OCSP responses for all the certificates in the certificate chain, excluding the trust anchor.
▷ ▷`xadesv141:TimestampValidationData`	This element is used only when batch timestamping is not used. See Section 2.2 for details. If the timestamping provider's certificate is not present in the TST, then the certificate will be presented in this element.

Table 3: Contents of ASiC container for simple signatures

File	Comments
`message.xml`	SOAP message (content type is `text/xml`)
`META-INF/signatures.xml`	XAdES signature that conforms to Section 2.1.1 and references the `message.xml` file.

Table 4: Contents of ASiC container for batch signatures

File	Comments
`message.xml`	XML part of the SOAP message.
`sig-hashchainresult.xml`	Result of the Merkle tree calculation. This file will be signed.
`sig-hashchain.xml`	Hash chain calculation from the input data (`message.xml` and optional attachments) to the hash chain result.
`META-INF/signatures.xml`	XAdES signature that conforms to Section 2.1.1. One `ds:Reference` element in the signature refers to file `sig-hashchainresult.xml` that contains the result of the Merkle tree calculation. In order to verify that the message was signed, one needs to verify that it was part of the Merkle tree resulting in the signed hash chain result.

to separately timestamp every received signature. This eliminates the risk of failed timestamps, but puts a significant load on the TSP and creates a new single point of failure in the system.

Simple timestamps created by the security servers are compatible with the requirements to XAdES `SignatureTimeStamp` element. If the timestamp does not contain TSP's certificate then the certificate is stored in the `xadesv141:TimeStampValidationData` signature property.

Batch timestamps created by the security servers use batch hashing [3] to hash several signatures together in a Merkle tree and timestamp the root hash of the tree. With batch timestamps, the input to the timestamping process is the entire signature (contents of `META-INF/signatures.xml` file). Table 5 shows how the batch timestamps are stored in the ASiC container.

3 Verification Algorithm

This chapter describes the algorithm that should be used when verifying the ASiC containers conforming to this specification.

Table 5: Contents of ASiC container for batch timestamps

File	Comments
`ts-hashchainresult.xml`	Result of the Merkle tree calculation. This file will be timestamped.
`ts-hashchain.xml`	Hash chain calculation from the signature (the entire contents of the `META-INF/signatures.xml` file) to the hash chain result.
`META-INF/ASiCManifest.xml`	Manifest connecting the TST with the timestamped data object (`ts-hashchainresult.xml`).
`META-INF/timestamp.tst`	RFC 3161 compatible timestamp token. The input to the timestamping process is the digest of the `ts-hashchainresult.xml` file.

1. If the container contains separate timestamp token (file `META-INF/timestamp.tst`), verify the token. The input to the TST must be a hash chain calculation. Verify that the batch hash calculation takes the signature (file `META-INF/signatures.xml`) as input.
   The `genTime` field of the TST will become the **time of signing**.

2. If the container does not contain a separate timestamp token then extract the TST from `SignatureTimeStamp` unsigned property in the signature (file `META-INF/signatures.xml`). The input to the TST must be the `ds:SignatureValue` element according to the XAdES specification.
   The `genTime` field of the TST will become the **time of signing**.

3. Verify the signature. The certificates in the certificate chain must be valid at the time of signing. The OCSP responses used to validate the certificates must be fresh – the time difference between the `producedAt` field of the OCSP response and the time of signing must not exceed threshold configured in the security policy of the verifier.

   If the signature refers to the hash chain result then verify the hash chain result. The input to the batch hash calculation must be the SOAP message (file `message.xml`). Additionally, the hash chain may refer to attachments that are not present in the container. In this case, the verifier should retrieve the attachments and use the message digests present in the hash chain to verify them.

References

[1] Arne Ansper, Ahto Buldas, Margus Freudenthal, and Jan Willemson. High-Performance Qualified Digital Signatures for X-Road. In *18th Nordic Conference, NordSec 2013*, LNCS 8208, pages 123–138. Springer, 2013.

[2] Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC), Version 1.3.1. Technical Specification TS 102 918, ETSI ESI, June 2013.

[3] Margus Freudenthal. Using Batch Hashing for Signing and Time-Stamping. Research Report T-4-20, Cybernetica AS, 2013.

[4] X.509 Internet Public Key Infrastructure. Online Certificate Status Protocol - OCSP. Request for Comments 6960, Internet Engineering Task Force, June 2013.

[5] Simple Object Access Protocol (SOAP) 1.1. W3c note, May 2000.

[6] SOAP Messages with Attachments. W3c note, December 2000.

[7] Internet X.509 Public Key Infrastructure. Time-Stamp Protocol (TSP). Request for Comments 3161, Internet Engineering Task Force, August 2001.

[8] Electronic Signatures and Infrastructures (ESI); XML Advanced Electronic Signatures (XAdES). Technical Specification TS 101 903, ETSI ESI, December 2010.

[9] Electronic Signatures and Infrastructures (ESI); XAdES Baseline Profile. Technical Specification TS 103 171, ETSI ESI, March 2012.

A Example ASiC Containers

A.1 Simple Signature and Simple Timestamp

This example corresponds to the situation where the SOAP message does not contain attachments and the signing device has sufficient speed to sign each message separately. Additionally, the security server timestamps each signature separately. In this case the ASiC container will contain the following files:

- `mimetype`
- `message.xml`
- `META-INF/signatures.xml` (contains `SignatureTimeStamp` unsigned property)
- `META-INF/manifest.xml`

A.2 Batch Signature and Batch Timestamp

This example corresponds to the situation where the SOAP message does contains attachments (not included in the ASiC container) or when the the signing device has insufficient speed to sign each message separately. Additionally, the security server timestamps several signatures together creating a batch timestamp. In this case the ASiC container will contain the following files:

- `mimetype`
- `message.xml`
- `sig-hashchainresult.xml`
- `sig-hashchain.xml`
- `ts-hashchainresult.xml`
- `ts-hashchain.xml`
- `META-INF/manifest.xml`
- `META-INF/signatures.xml`
- `META-INF/timestamp.tst`
- `META-INF/ASiCManifest.xml`