Signing

Svensk e-identitet AB

Version 3.0-213-g5a97c7d, 2026-01-09

Table of Contents

Purpose

The purpose of this document is to provide deep knowledge of the signature process making it possible to verify the signatures of agreements signed using the eAvtal service.

Overview

This document describes how eAvtal handles the signature process, from how to make all agreement files tamper-proof, to how each Electronic ID (EID) provider implements its signatures.

System

For the agreement signatures to be trustworthy, it is important that a stable and high-quality system protects the signing process.

All parts providing evidence for the signature, such as hardware, operating system and software, have to be trusted.

Hardware

The eAvtal service runs on a virtualized hardware platform. It is hosted in a data center with high availability requirements, used by large banks and e-commerce companies.

Software

For a signature to be trusted, it is important that the timestamp can be reliable. Timestamps are used to show when certain events occur and are an important evidence when validating an agreement.

The platform synchronizes its time over the standardized Network Time Protocol (NTP).

Electronic Signatures

An electronic signature is any electronic means that indicates that a person consents to the contents of an electronic message. In the EU the eIDAS legislation regulate and defines three levels of such signatures:

- *Simple Electronic Signatures*
- *Advanced Electronic Signatures*
- *Qualified Electronic Signatures*

These signatures are created using an electronic identity that has a certain security Level of Assurance

(LOA) also defined by eIDAS. The LOA describes the degree of confidence the identity was issued, taking into account the enrollment process as well as protection of the authentication needed to use the identity.

- Low - For example self-registration.
- Substantial - Providing and verifying identity information in a structured way.
- High - For example a requirement to have a physical meeting before issuing an identity.

Simple Electronic Signature (SES)

- Basic electronic signature with low assurance.
- Typically equivalent to a handwritten signature.
- Minimal identity verification required.

Is defined as "data in electronic form which is attached to or logically associated with other data in electronic form and which is used by the signatory to sign".

Supported methods

- Draw Signature - A handwritten signature drawn on a digital canvas.
- API Signature - A signature created by consuming the web service exposed by Svensk e-identitet AB.
- MitID low - Danish MitID on LOA level low
- Click Signature - A signature created by personalized link.

Advanced Electronic Signature (AES)

- is uniquely tied to the signatory,
- uniquely linked to and capable of identifying the signatory
- created in a way that allows the signatory to retain control
- linked to the document in a way that any subsequent change of the data is detectable.

Electronic signatures are usually based on PKI, such as the Swedish BankID and Norwegian BankID.

Supported methods

- Swedish BankID
- Norwegian BankID
- Finnish strong identification
- Danish MitID - MitID on LOA level substantial

- SMS Signature - A signature created by responding to an SMS with context specific content.
- Email Signature with OneTimePassword - A signature created by One Time Password (OTP) based email service.

Qualified Electronic Signature (QES)

A qualified electronic signature is an advanced electronic signature which is additionally:

- Highest Level of Assurance and trust.
- created by a qualified signature creation device (QSCD)
- Requires a qualified digital certificate from a qualified trust service provider.
- Legal equivalence to a handwritten signature in many jurisdictions.

Supported methods

Electronic signatures based on PKI where the signature is the highest level of electronic signature under eIDAS, providing the utmost trust, legal equivalence to handwritten signatures.

- Swisscom Mobile ID

Cryptography

Cryptography is a central part of many signature solutions, for example to provide tamper-proof agreement files.

The following chapters introduce some concepts needed to understand cryptography based signatures.

Cryptographic Hash Functions

A cryptographic hash function is any function that can be used to map data of arbitrary size to data of fixed size where it is practically impossible to find two pieces of input that produces the same hash value. It is a one way transformation that produces the same result every time. The result is called a hash and since it is a one way function, it is not possible to re-create the input behind a hash.

Examples on hash functions are:

- SHA.2
- RIPEMD
- Whirlpool

Asymmetric Encryption

In cryptography, two types of encryptions are most commonly used, symmetric and asymmetric encryption. The difference between them is that the symmetric encryption uses one single key for both encryption and decryption, while asymmetric uses one key for encryption and another for decryption. The fact that there are two keys can also be used for creating signatures and serves as a base for many types of EID. The EID solutions usually use the asymmetric encryption in a Public Key Infrastructure (PKI).

Asymmetric encryption make use of keypairs that if a text is encrypted using one of the keys, the only key that can decrypt the crypto text is the other key belonging to the pair. One key is private and should not be shared with anyone, the other is public and is used to verify signatures created with the private key.

Examples on asymmetric keys are:

- RSA
- DSA

Digital Signatures

When signing using asymmetric keys, the signature result is called a raw signature and is just a binary blob of a specific size, without any information at all. In order for the signature to be used in a distributed environment, there is a need to add information on who signed the blob, what was signed and so on.

The combination of metadata and the raw signature is called a digital signature. Some information is optional:

- Signature time
- X.509 Signature Certificate
- X.509 CA Certificates (who issued the X.509 Signature Certificate)
- Signature text (TBS)
- Hash algorithms used
- Signature algorithm used
- The raw signature

Examples on digital signature formats are:

- CadES/CMS/PKCS#7
- XAdES

Signing

To sign an agreement a number of actions need to be executed:

- Create text representation of agreement
- Create digital signature

Create text representation

For advanced electronic signatures, where the agreement contents is logically tied to the created signature, the text needs to be prepared in order to suit the signing method used. The textual representation of an agreement is also called To Be Signed, or TBS for short. The TBS is the actual text that is to be digitally signed. For the whole agreement to be tamper-proof, the agreement metadata with all agreement files needs to be signed. Digital signature clients have limitations on the size of TBS. It is not possible to sign the actual agreements, since there is a risk that the agreement files are too large. Therefore cryptographic hash functions are used to reduce the size of the agreement files and create a representation of it.

For non-advanced signatures, where the agreement contents is not directly tied to the signature, the above preparation is normally not needed.

Digital signature clients allow us to send two different types of TBS. They are:

- Visible TBS
- Hidden TBS - Swedish BankID only

Visible TBS

TBS text that is exposed to the signatories by the digital signature clients. General information about the Agreement is supplied to digital signature clients as a visible TBS. As the TBS will be different between signing clients and also have different localizations it will be saved together with the digital signature. In cases where hidden TBS is not supported, the visible TBS will contain information that otherwise would be in the hidden TBS. Where the TBS has length restrictions due to technical reasons, the hash may be base64 encoded (trailing `=` characters being omitted) and the hash may be calculated as if the ordered agreement files were a single file.

Example

```
I sign: "Employment ACME Inc"
eAvtal ID: 2394857302
Date: 26/04/2016
```

Hidden TBS

TBS text that is hidden from the signatories by the digital signature clients. Agreement content is reduced to fewer characters using cryptographic hash functions is supplied to digital signature clients as a hidden TBS. This is currently only supported for Swedish BankID.

Example

```
----------
Content:

"Contract"
1103 b497 984e 433c 38d4 f4ef f775 33ad 0cef 8c07 9995 5b37 7c8e e4df 66f7 de73

"Codes of conduct"
2cf9 688d c483 2085 1f12 6120 9001 58b2 ab2e 62b1 afd5 acc5 ebbb 5134 c7a4 90f2
----------
```

Create Digital Signature

For advanced electronic signatures, the cryptographic signature is created by applying the private key from an asymmetric key pair on a dataset that previously has been hashed. In this context, the dataset is the TBS. Different EID providers handles digital signatures in different ways, Signature Methods describes all methods supported.

For non-advanced signatures, signature creation will vary depending on the method.

Signature Methods

Currently, the following signing methods are supported:

- Swedish BankID (advanced electronic signature)
- Norwegian BankID (advanced electronic signature and authentication)
- Finnish strong identification
- Danish MitID
- Swisscom Mobile ID (qualified electronic signature)
- Draw Signature
- API Signature
- SMS Signature
- Email Signature with OneTimePassword

- Click Signature

These are described in detail below.

Swedish BankID

Swedish BankID is the largest EID provider in Sweden run by Finansiell ID-Teknik BID AB which is owned by a number of large swedish banks.

It is a PKI based solution with support for file based tokens as well as smart cards, supported on a number of platforms for desktop and mobile.

Signature format

The BankID client produces an XML Digital Signature as specified in BankID Signature profile.

Audit log

Each signature transaction provides information used in user interface and audit logging:

- personNr - In the form of a Swedish personal identity number.
- personSurname - Surname in capital letters.
- personGivenName - Given name in capital letters.
- startDate - Date when signing was started.
- signatureMethod - Will be `bankid` or `bankid-otherunit`.
- transactionId - Transaction ID from backend system.
- signature - Signature, base64 encoded.
- tbs - Data being signed, base64 encoded.
- tbsHidden - Hidden data being signed, base64 encoded.

How to validate

Validate the signature according to BankID Signature profile.

Output from validation to provide to *Signee Validation* in Container Format is:

- User attributes of the first certificate of the XML element *KeyInfo* with ID *bidKeyInfo* in the signature.
- Hashed TBS.
- TBS Hash algorithm.

Norwegian BankID - advanced electronic signature

Norwegian BankID is the largest EID provider in Norway run by BankID Norway AS.

It is a PKI based solution that uses a one time password in combination with a personal password, supported on a number of platforms for desktop and mobile.

Signature format

The BankID client produces a detached PKCS#7/CMS signature as specified in Cryptographic Message Syntax, where the detached data being signed is the TBS.

Audit log

Each signature provides data for the *Agreement Signed* specified in Container Format. The data provided is:

- personNr - Person unique ID, not the Norwegian personal identity number.
- personSurname - Surname.
- personGivenName - Given name.
- personDateOfBirth - Date of birth for the person.
- startDate - Date when signing was started.
- signatureMethod - Will be `norbankid-high`, `norbankid-netcentric` or `norbankid-mobile`.
- signatureAction - Will be `signature`.
- transactionId - Transaction ID from backend system.
- signature - Signature, base64 encoded.
- tbs - Data being signed in the detached PKCS#7 signature, base64 encoded.

How to validate

Validate the signature according to *5.6 Message Signature Verification Process* in Cryptographic Message Syntax.

Output from validation to provide to *Signee Validation* in Container Format is:

- User Attributes of the signer certificate in the element *SignedData certificates* in the PKCS #7 signature.
- Hashed TBS
- Hash algorithm

Norwegian BankID - authentication

Authentication based signing is also provided by the means of Norwegian BankID.

Audit log

Each signature provides data for the *Agreement Signed* specified in Container Format. The data provided is:

- personNr - Person unique ID, not the Norwegian personal identity number.
- personSurname - Surname.
- personGivenName - Given name.
- personDateOfBirth - Date of birth for the person.
- startDate - Date when signing was started.
- signatureMethod - Will be `norbankid-substantial`, `norbankid-netcentric` or `norbankid-mobile`.
- signatureAction - Will be `authentication`.
- transactionId - Transaction ID from backend system.

How to validate

As an authentication alone does not digitally sign the agreement content, the person details such as given name, surname and date of birth must be compared with the requested signee.

Finnish strong identification

Strong identification broker services are provided by Telia Finland for banks and mobile operators in the Finnish Trust Network ("FTN").

Identity format

Telia produces a signed JWT which contains information about the person.

Table 1. Signed Attributes

Attribute Name	Attribute Value
`urn:oid:1.2.246.21`	Social security number / personal identity code / HeTu - henkilötunnus
`urn:oid:2.16.840.1.113730.3.1.241`	Full name
`urn:oid:2.5.4.4`	Surname
`urn:oid:1.2.246.575.1.14`	Given name

More information can be found in Telia Tunnistus.

Audit log

Each signature transaction provides information used in user interface and audit logging:

- personNr - Finnish personal identity code.
- personSurname - Surname
- personGivenName - Given name
- startDate - Date when signing was started.
- signatureMethod - Will be `eid-fi`.
- transactionId - Transaction ID from backend system.
- signature - Signed JWT (from Telia).
- publicKey - Public key (JWK) for the signed JWT (from Telia).

How to validate

The signed JWT can be verified using the public key.

Danish MitID

Danish MitID signing is utilizing Transaction Signing from `Nets eID Broker`.

Signature format

The signature flow for Danish MitID is performed as a text followed by end-user authentication, so there is no signature produced. Thus a signature format does not exist.

Audit log

Each signature provides data for the *Agreement Signed* specified in Container Format. The data provided is:

- personNr - Person unique ID, not the Danish personal identity number.
- personName - Full name.
- personDateOfBirth - Date of birth for the person.
- startDate - Date when signing was started.
- signatureMethod - Will be one of: `mitid-low`, `mitid-substantial` or `mitid-high`.
  - The method `mitid-<LOA>` specifies what Level of Assurance (LOA) that was requested when initiating the signature.

signatureAction - Will be `authentication`.

- transactionId - Transaction ID from backend system.
- tbs - Data that was shown to user before authenticating, base64 encoded.

More information can be found in Danish MitID.

How to validate

Due to the nature of this kind of signature, the signature in itself cannot be machine validated in the same way as an advanced signature. In case of dispute, contact the EID provider, Nets, to validate the entries in the audit log.

Swisscom Mobile ID

Swisscom Mobile ID is a qualified electronic signature method provided by Swisscom Trust Services.

It is a PKI based solution that uses a smartphone application to authenticate signees. After successful authentication a short lived certificate is issued that is used to produce a Qualified Electronic Signature.

Signature format

Swisscom Trust Services produces a detached CMS signature as specified in Cryptographic Message Syntax, where the detached data being signed is the tbsHidden and message is the message shown to the user in the Mobile ID client.

Audit log

Each signature provides data for the *Agreement Signed* specified in Container Format. The data provided is:

- mobile - Mobile no used to trigger the signee's authentication of the signature.
- personName - Full name.
- personSurname - Surname.
- personGivenName - Given name.
- personSerialNumber - Swisscom evidenceId.
- startDate - Date when signing was started.
- signatureMethod - Will be `swisscom-qes`.
- transactionId - Transaction ID from backend system.
- signature - Signature, base64 encoded.
- message - Message shown in the Mobile ID client when user approves the signature, base64
-

encoded.

- tbsHidden - Data signed in the detached CMS signature, base64 encoded.

More information can be found in Swisscom AIS and Swisscom SRS.

How to validate

Validate the signature according to *5.6 Message Signature Verification Process* in Cryptographic Message Syntax.

Output from validation to provide to *Signee Validation* in Container Format is:

- User Attributes of the signer certificate in the element *SignedData certificates* in the CMS signature.
- Hashed TBS
- Hash algorithm

Draw Signature

The draw signature signing method is an electronic counterpart to traditional handwritten signatures. Using draw signature, the end user draws a signature on a web canvas. The signing method is not classified as an advanced electronic signature in that it does not tie the user to the contents signed and does not identify this signatory. Yet, the method can be suitable in use cases where advanced electronic signatures are not required.

Signature format

The signature is represented as an SVG/PNG image.

Audit log

Each signature provides data for the *Agreement Signed* specified in Container Format. The data provided is:

- personNr - In the form of a swedish personal number.
- personSurname - Surname
- personGivenName - Givenname
- signatureMethod - Will be `draw-signature`.
- signature - The SVG or PNG image, base64 encoded.
- signatureContentType - The content type of signature eg. image/svg, image/png.

How to validate

Due to the nature of this kind of signature, the signature in itself cannot be machine validated in the

same way as an advanced signature. Also, as the case of traditional handwritten signatures where there are no formal requirements placed, ocular comparison does not add to trustworthiness.

API Signature

The API Signature is an electronic signature which is done by consuming the web service exposed by Svensk e-identitet AB. Using API Signature, the end user signs the agreement by making a request to the web service. Using this method, user can also send the timestamp of actual time when the sign event took place. This signing method is not classified as an advanced electronic signature in that it does not tie the user to the contents signed and does not identify this signatory. Yet, this method can be suitable in use cases where advanced electronic signatures are not required.

Audit log

Each signature provides data for the *Agreement Signed* specified in Container Format. The data provided is:

- personNr - In the form of a swedish personal number.
- personSurname - Surname
- personGivenName - Givenname
- signatureMethod - Will be `api-signature`.
- signedOn - Time at which the agreement was actually signed (stated by user)

How to validate

Due to the nature of this kind of signature, the signature in itself cannot be machine validated in the same way as an advanced signature.

SMS Signature

SMS Signature is an electronic signature created by responding to an SMS with context specific content. The content of the SMS is One Time Password (OTP), a 6 digits number sent to the registered mobile no of the signee. The Signee has to submit the OTP in order to sign the agreement. The OTP Number is generated by TOTP algorithm by hashing the current time along with the agreement information. OTP can be regenerated and verified by the same logic provided the given time & agreement information are same.

Audit log

Each signature provides data for the *Agreement Signed* specified in Container Format. The data provided is:

- mobile - Mobile no of the signee.

- timestamp - Date when signing was started.
- agreementName - Name of the agreeement.
- signatureMethod - Will be `sms-signature`.
- OTP - One Time Password.

How to validate

One Time Password (OTP) can be verified using the following algorithm from the attributes (agreementName, mobile, timestamp).

- HASHLG : Use SHA-256 algorithm to generate hash of the given input.
- hash (array): HASHLG(agreementName + mobile + timestamp)
- random (function): hash[${param1}] & ${param2} << ${param3}
- otp : Follow the below steps to generate OTP from the hash.
  - offset = random(hash.length - 1, 15, 0)
  - binary = random(offset, 127, 24) | random(offset + 1, 255, 16) | random(offset + 2, 255, 8) | random(offset + 3, 255, 0)
  - otp = binary % 1000000
  - If the output is less than 6 digits, add the remaining digits as '0' in prefix.
- Now verify the ${otp} generated against the OTP that you can find in the auditLog.json.

Email Signature with OneTimePassword

Email signature with OneTimePassword is an electronic signature created by One Time Password(OTP) based email service. The content of the Email is One Time Password (OTP), a 6 digits number sent to the registered email address of the signee. The Signee has to submit the OTP in order to sign the agreement. The OTP Number is generated by TOTP algorithm by hashing the current time along with the agreement information. OTP can be regenerated and verified by the same logic, provided the given time & agreement information are same.

Audit log

Each signature provides data for the *Agreement Signed* specified in Container Format. The data provided is:

- email - Email address of the signee.
- timestamp - Date when signing was started.
- agreementName - Name of the agreeement.
- signatureMethod - Will be `email-signature`.

- OTP - One Time Password.

How to validate

One Time Password (OTP) can be verified using the following algorithm from the attributes (agreementName, email, timestamp).

- HASHLG : Use SHA-256 algorithm to generate hash of the given input.
- hash (array): HASHLG(agreementName + email + timestamp)
- random (function): hash[${param1}] & ${param2} << ${param3}
- otp : Follow the below steps to generate OTP from the hash.
  - offset = random(hash.length - 1, 15, 0)
  - binary = random(offset, 127, 24) | random(offset + 1, 255, 16) | random(offset + 2, 255, 8) | random(offset + 3, 255, 0)
  - otp = binary % 1000000
  - If the output is less than 6 digits, add the remaining digits as '0' in prefix.
- Now verify the ${otp} generated against the OTP that you can find in the auditLog.json.

Click Signature

Click signature is an electronic signature created by using a personalized link. The personalized link can be shared by email or directly in an integrated application. When the signee visits the link, it will take him to the agreement review page where he can click the sign button to sign the agreement.

Audit log

Each signature provides data for the *Agreement Signed* specified in Container Format. The data provided is:

- email - Email address of the signee.
- timestamp - Date when signing was started.
- signatureMethod - Will be `click-signature`.

How to validate

Due to the nature of this signature , we can't validate the authenticity of the signature. Signatories has to keep the link secret.

Reference documentation

Document name	Description
Overview	Gives an overview of the eAvtal agreement function and links to the other documents.
Container Format	Describes the details of how agreement information is stored within the container PDF.
Signing	Describes how eAvtal handles the signature process.
Validation	Describes how to validate a container. Makes references to other documents for validation details.
BankidValidation	Signature Profile for BankID, v2.3
Telia Tunnistus	Telia Tunnistus - Integration guide to identification broker service, v1.8
Nets eID Broker	Technical reference for service providers, v1.2.3
Swisscom AIS	All-in Signing Service Reference Guide, v2.15
Swisscom SRS	Integration Guide Smart Registration Service, v1.8
Cryptographic Message Syntax	https://www.ietf.org/rfc/rfc2630