EUROPEAN COMMISSION
DIRECTORATE-GENERAL INFORMATICS

Directorate D - Digital Services
DIGIT D3 - Trans-European Services

eIDAS Bridge

WP2 - Use cases and technical specifications

Commission européenne/Europese Commissie, 1049 Bruxelles/Brussel, BELGIQUE/BELGIË - Tel. +32 22991111
Office: B-28 03/P002 - Tel. direct line +32 229-52574

TABLE OF CONTENTS

1. CONTEXT

1.1. Adding legal value to Verifiable Credentials

A Verifiable Credential (VC or Attestation) contains concrete information about an entity (usually the "Holder") and is being issued by an "Issuer". Verifiable Credentials can be considered electronic documents. However, currently, their legal value cannot be relied upon in a sense that it is open for interpretation, as there is no law that would confer binding legal value to VCs. Legal value would depend solely on the contractual relationship between parties.

In many cases, this might not be problematic (in terms of a VC - or derivative Verifiable Presentation's - capability do establish sufficient trust for enabling a transaction). However, there are cases which may require VCs with (regulatory supported) legal value. In such cases, we consider eIDAS as the currently available legal framework that could provide legal value to VCs. More specifically, by the use of "eIDAS" electronic seals, authenticy / integrity / non-repudiability and legal certainty can be given to the issued VC.

1.2. How could it be accomplished?

In order to add legal value to the VC, the following should be provided:

- Trusted Issuers (legal persons) should be issued a (qualified) certificate for electronic seals as defined under eIDAS.
- When issuing VC, Issuers would seal those with the (qualified) certificate for electronic seals
  (this of course under the conditions as set under the eIDAS regulation and corresponding implementing acts and standards)
- Holders and Verifiers receiving a VC should be able to validate the electronic seal
  (this of course in line with the best practices on eSeal validation as eg defined by ENISA).

1.3. The eIDAS Bridge (and scope)

This chapter presents the *eIDAS Bridge* as a component that will be used by a Wallet in order to provide electronic sealing functionalities. The main reason to do so is to abstract and separate the responsibilities of each of the components. Generally speaking, Wallets are supposed to be built to support SSI-related use cases, that do not necessarily include full eSealing functionality (as defined under eIDAS). The eIDAS Bridge component will provide "the bridge" between the user's wallet and allow to call esealing-capabilities.

Mind that for the proof of concept we describe the situation whereby the eSealing is done by the issuer itself. Eg for non-esealing scenarios, the wallet will use its local "secure enclave" and for "esealing" the wallet will use this "bridge". This "bridge"

component is first and foremost seen as a component that will be deployed "next to the wallet" as to support the holder's wallet to do eSealing and to call the holder's HSM to execute this eSealing. In future releases this design will need to be elaborated as to ensure both "local" deployment and to support "remote esealing" with all the security measures required under eIDAS for remote eSealing.

2. COMPONENTS DESCRIPTION

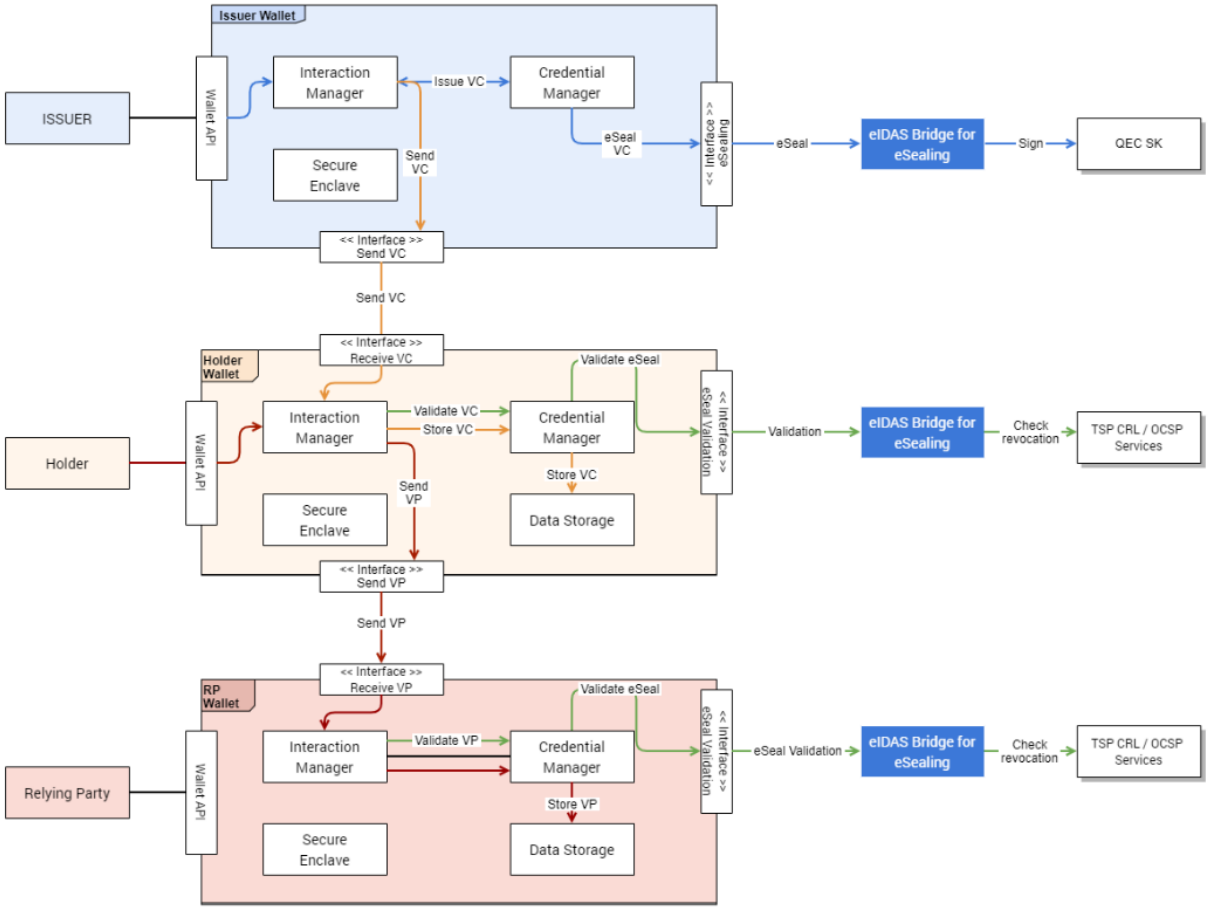
2.1. Electronic sealing components

There are 2 main flows related to the electronic sealing of VCs:

- the creation of the electronic seal by the Issuer
- the verification, by either the Holder or by the Relying Party.

In the following diagram, both flows are depicted together with the delivery of the VC between different wallets (Issuer to Holder, and Holder to Relying Party).

Mind that the diagram does not shown how the different entities interact with their respective wallets to trigger the different operations.

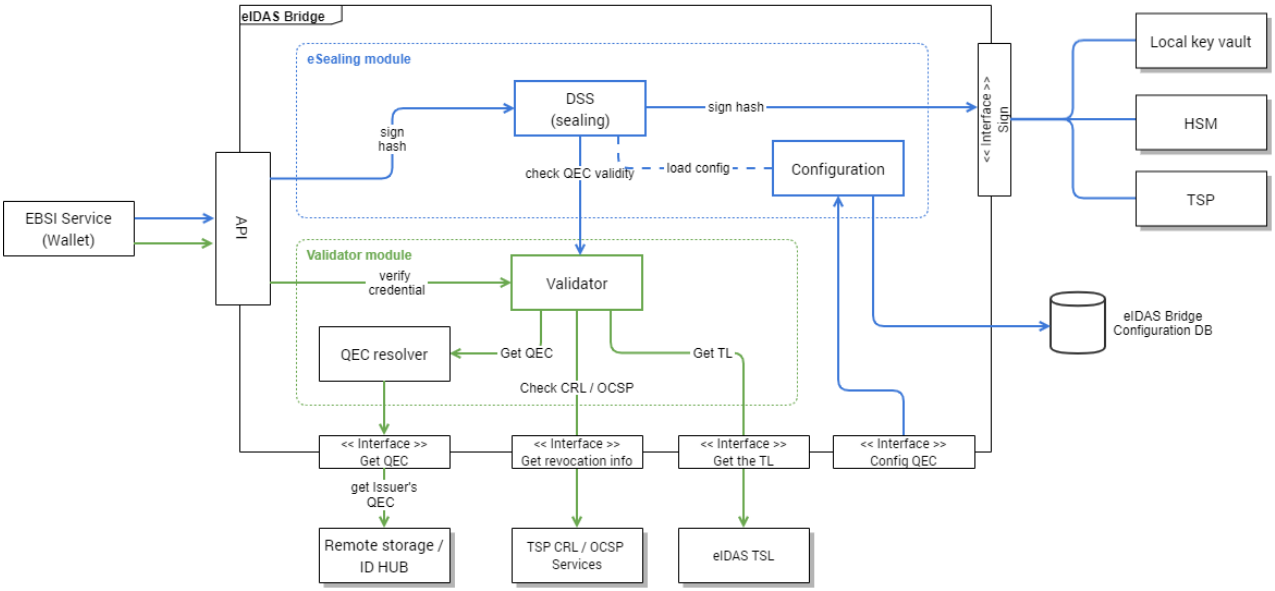


Note that the eIDAS-bridge is not a central component. Each party (or group of parties) can have its own eIDAS bridge and decide to which TSP(s) they choose to connect.

In the PoC, the eIDAS bridge should be first and foremost seen as a component that will be deployed "next to the wallet" as to support the holder's wallet to do eSealing and to call the holder's HSM to execute this eSealing.

2.2. The logical structure of the eIDAS bridge for electronic seal component

Below we can see the detail of the eIDAS bridge for eSealing component with the different main modules involved: electronic sealing and validation.



2.2.1. Component description

2.2.1.1. Electronic sealing module

Component	Responsibilities	Description	Integration with other components
DSS (Digital Signature Service)	• Sign a VC with QEC's SK • Validate QEC before signing VC	The DSS is responsible for digitally signing the hash of the VC received from the Wallet with the SK of its QEC for electronic seal. This means it should sign it, build the cryptographic proof, and return it to the wallet. Before creating the signature, it should interact with an external component in order to retrieve the QEC and perform a validation against the Validator component to verify that the QEC is still valid. It also should load the SK Interface Configuration from the Configuration component in order to be able to access the QEC's SK. As input parameters, the DSS could accept a full VC or just its hash, together with the DID or the DID Document of the signer. For EBSIv1 the full VC and DID Document are required.	To create the signature, the DSS needs to interface with the component that holds the SK. Before creating the electronic seal, the DSS needs to verify the validity of the QEC with the Validator component.
Configuration	• Set up the configuration for the access to the SK of the QEC	The configuration component is responsible for setting up how a specific Issuer will access its QEC SK, and provide the SK interface configuration to the DSS component when required. For EBSI v1 only a local key vault will be used. For production-ready deployment, an HSM to store the SK or a DSS service provided by a TSP should be used.	Before creating an electronic seal, the DSS component should load the configuration to access the Issuer's SK. The Configuration component should persist to an external device the configuration.

2.2.1.2. Validation module

Component	Responsibilities	Description	Integration with other components
Validator	• Validate the VC's proof • Verify QEC validity • Get QEC	The Validator is responsible for all the tasks related to the cryptographic verification of digital signatures and QEC. This includes the verification of the VC signatures, the QEC signatures and its revocation status (either by querying its OCSP responder or by getting the CRL list). In order to validate a VC seal, the Validator will have to get the QEC that was used to create the seal by querying the QEC resolver.	The validator needs to get the validation information from the TSL, the revocation information from the OCSP responder or the CRL list, and also needs to get the QEC from the QEC Resolver.
QEC resolver	• Get the QEC for a specific DID	The QEC resolver should be able to retrieve a QEC given a DID Document. More specifically, the QEC Resolver will parse the DID Document to get the service endpoint that can be queried to retrieve the QEC. It has been proposed that this service endpoint could be an ID Hub, but could also be any other endpoint that is publicly accessible. The QEC should return the full certificate chain that the Validator will use to validate the certification path.	The resolver will be queried by the Validator to get the QEC.
Configuration	• Set up the configuration for the QEC	The configuration component is responsible for setting up how a specific Issuer will access its QEC SK, and provide the SK interface configuration to the DSS component when required. For EBSI v1 only local key vault will be used. For production-ready deployments, an HSM to store the SK or a DSS service provided by a TSP should be used.	Before creating an electronic seal, the DSS component should load the configuration to access the Issuer's SK. The Configuration component should persist to an external device the configuration.

2.2.2. Interfaces description

Interaction	Functionality	Description	Components involved
Configure QEC	• Configure QEC SK's access	For every Issuer, there must be a Configuration element that describes how the Issuer will access its QEC SK in order to create the electronic seals. This interface should be accessed only by the persons or mechanisms authorized to access the SK of the QEC.	This interface is accessed to directly modify the Configuration module.
Get QEC	• Get the QEC of the Issuer of the VC	In order to validate an electronic seal, the Validator component needs to fetch the QEC that has signed the VC in order to validate the digital signature. Together with the QEC, it must retrieve the full certificate chain.	This interface is called from the Validator component. To get the QEC the remote storage (ID Hub or similar) will be queried.
Check CRL/OCSP	• Get the revocation status of a certificate	Before creating an electronic seal and during its validation, the Validator component needs to check the revocation status of the QEC querying the OCSP responder or by fetching the certificates revocation list (CRL).	This interface is called from the Validator component.
Get the TL	• Retrieve the eIDAS Trusted List	The validator needs to have an updated version of the eIDAS Trusted List in order to be able to validate a QEC.	This interface is called from the Validator component.
Sign	• Sign payload with SK	The Digital Signature Service component will need to sign the VC hash using the QEC SK. This interface should be an abstraction for the DSS module that is implemented from the configuration loaded from the Configuration module.	This interface is called from the DSS component.

3. MAIN ELECTRONIC SEAL FLOWS

3.1. Electronic sealing of VCs

Depending on whether a signature is generated by a natural or legal person, it is legally termed eSignature and eSeal, respectively. Depending on the LoA and Policy the signature can be basic, advanced, or qualified (eIDAS compliant).

Verifiable Credentials are signed once, even though an entity may possess multiple keys.

3.1.1. Approach

In the proposed approach, the Issuer will have two different types of keys, "DID keys" and "(Q)eSeal keys".

As to then "Seal" a VC there are different approaches. The one targeted for the PoC is to:

- **Sign the VC once with the QEC key. One eSeal will be added to the VC, as described in the W3C Verifiable Credentials Data Model.**

Note that other options are also possible (not used in the PoC), eg:

- Sign the VC twice. First sign the VC with the DID keys and later seal it with the QEC keys, creating thus 2 proof objects (one for "transport/authenticity-reasons" and one for "giving legal effect"), a possibility also described in the W3C Verifiable Credentials Data Model.
- Sign the VC once and nest it inside another VC. Even though this option is not contemplated by the W3C, it would be logically possible, since it means to define a context and declare it in the VC where a particular field has indeed a full VC inside.

3.1.2. Qualitative technical assessment

The following table intends to show the strengths and weaknesses of the different options elaborated in the previous section.

Green fields show strengths and that no noteworthy issues are to be expected. Orange fields mark expected noteworthy, but manageable issues, i.e. potential but manageable weaknesses. Red fields mark expected issues that are so grave that they should be avoided, i.e. grave weaknesses.

Criteria	Option 1: Sign VC once	Option2: Sign the VC twice	Option 3: nested VC
Interoperability	Green	Green	Red
Reliability	Green	Green	Green
Flexibility	Green	Orange - Green	Orange
Holistic	---	---	---

Openness	Green	Green	Orange
Mature	---	---	---
User Experience	Green	Green	Green
Scalability	Orange - Green	Green	Red
Feasibility	Green	Orange	Red
Privacy-by-design	---	---	---
Security-by-design	---	---	---
Simplicity	Green	Orange - Green	Red

3.1.2.1. Recommendation

For the sake of simplicity, the 1st option would be recommended, since it offers the same levels of reliability with easier implementation. Both would be technically feasible though and aligned with W3C recommendations.

By listing the QEC PK in the DID Document, we should achieve the same level of interoperability in option 1 as in option 2.

3rd option would be discarded since it is less *standard*, and presents more risks of interoperability, besides being the most complicated option of the 3.

3.1.3. Electronic sealing process

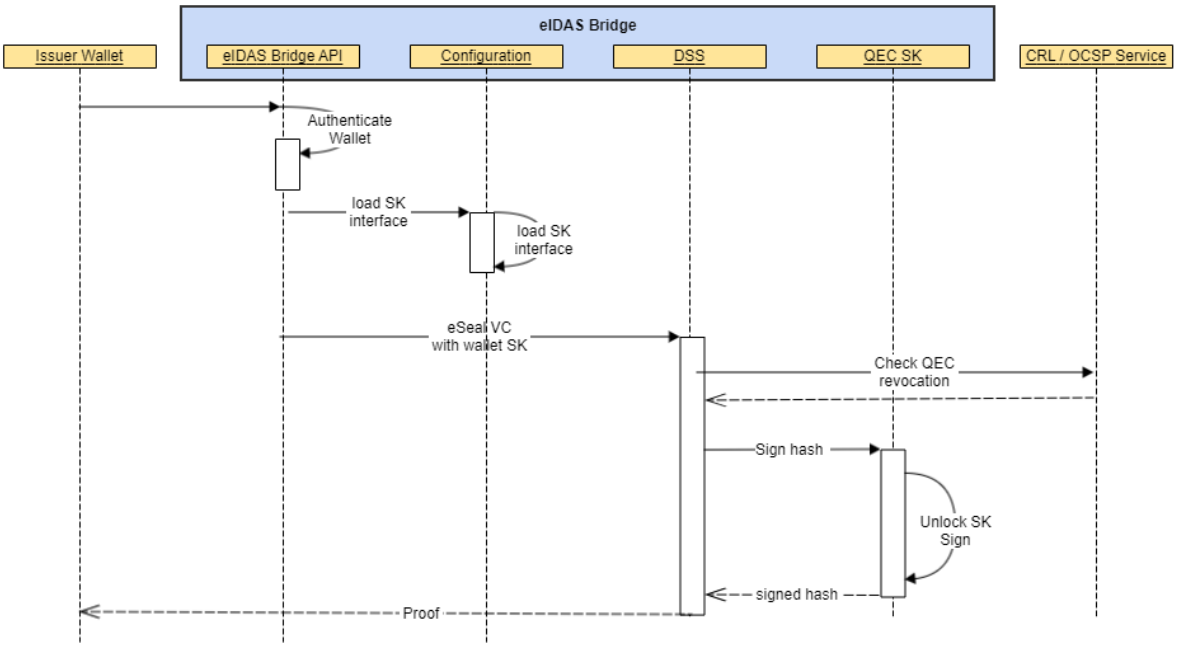
eIDAS Bridge will receive eSeal requests from eIDAS Wallet containing:

- Authenticated request.
- VC to be eSealed or, optionally, only the hash of the VC to be signed.

For EBSIv1, the following flow will be implemented:

1. eIDAS Bridge will verify the request comes from an authenticated source (a wallet) and will route the request to the Digital Signature Service module,
2. It then will load the required QEC SK interface.

3. The DSS will prepare the Proof and will send the hash to be signed to the QEC SK interface.
4. The interface will reach the corresponding SK repository to sign the hash.
5. The DSS will assemble all the parts of the proof and send it back to the wallet:
    1. If as input it has received the full VC, it will return the full VC with the Proof.
    2. If as input it has received only the hash of the VC, it will return just the Proof.



3.2. Electronic seal verification

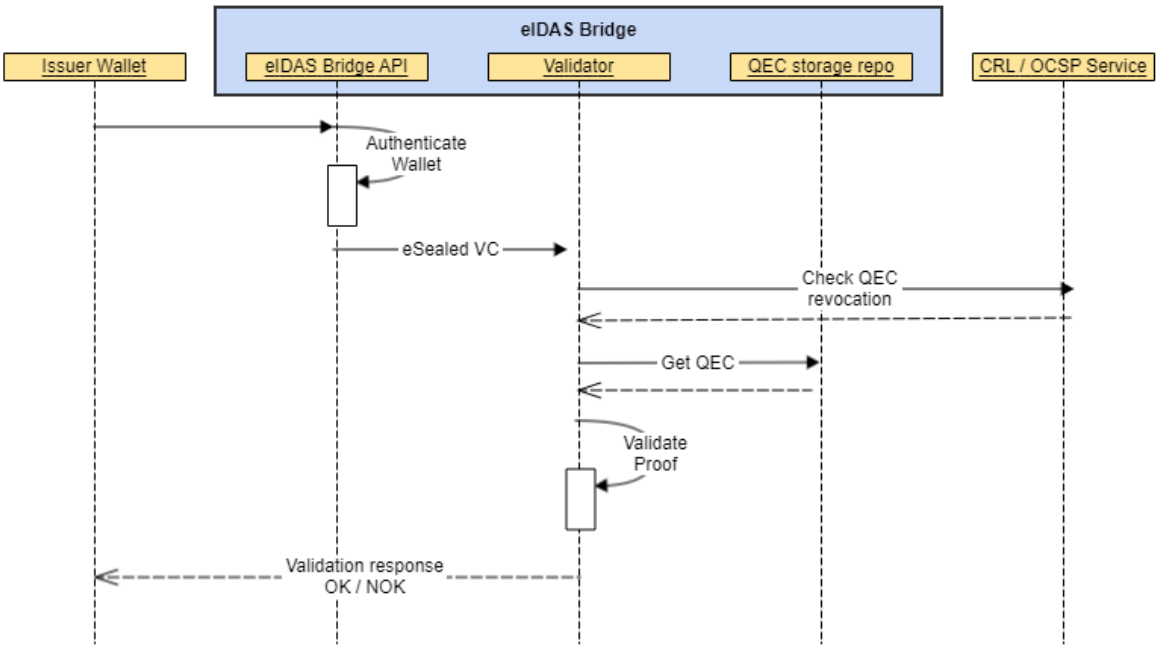
eIDAS Bridge will receive verification requests from eIDAS Wallet containing :

- An authenticated request (to authenticate the Wallet),
- an eSealed VC to be verified or, optionally, the proof + the hash to be verified. (

For the PoC, the following flow will be implemented:

1. The Bridge will route the request to the Validator module.
2. Validator module will parse the VC to get the eSealing Proof.
3. It will get the Qualified Electronic Certificate (QEC) from the remote storage (ID Hub).
4. It will verify the revocation status of the QEC.

5. Once verified, it will validate the electronic seal using the PK from the QEC.
6. It will send the verification status to the Wallet.



4. SUPPORTING FLOWS

4.1. Setup

The process of electronic sealing requires a previous setup of the following elements:

- Wallets' DIDs
- QEC for electronic sealing for the Issuer
- Public access to the QEC
- Restricted access to the QEC SK.

This document assumes DIDs have already been created. QEC for electronic seals must be provided by a qualified TSP.

4.1.1. Configure access to the QEC private key (SK)

The eIDAS Bridge component needs to restrict access to every Issuer's QEC SK from unauthorized use. The following options have been considered:

- QECs are provided in a keystore file (e.g. PKCS#12)
- QEC SK are generated on an HSM that the eIDAS Bridge has secure access to.

- Electronic sealing service is provided by a TSP

For every possible approach, every Issuer will have to configure the eIDAS Bridge with the required configuration and options to access its QEC SK. Since the Bridge can be a shared component, this configuration should be stored protected in a secure manner (e.g. encrypted with Issuer's DID SK for instance). This secured configuration could be unlocked and loaded whenever required by the Issuer or its Wallet.

For the PoC, QEC certificates provided in a keystore file will be used.

4.1.2. Configure access to the QEC

In order to the electronic seal to be validated, the Validator component will need to get access to the QEC since VC Proofs do not contain validation information about the certificate. The following options are being considered:

- Use an ID Hub to store the QEC
- Use a public repository (i.e. Trusted Issuers List)
- Expect that TSPs provide a service lookup for QEC

In any case, this service should be available to any Validator wishing to access the QEC. It is proposed that this service is published in the DID Document of the Issuer.

For the PoC ID Hub will be used to store the QEC.

The DID Document update could be triggered when onboarding an Issuer to ESSIF.

5. ANNEX

5.1. Example of an ESSIF Verifiable ID

A non-normative example of verifiable ID sealed by a qualified eSeal service implemented by eIDAS Bridge:


```
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://essif.europa.eu/schemas/vc/2019/v1",
    "https://essif.europa.eu/schemas/eidas/2019/v1"],
  "id": "did:ebsi-eth:00000001/credentials/1872",
  "type": ["VerifiableCredential", "EssifVerifiableID"],
  "issuer": "did:ebsi-eth:00000001",
  "issuanceDate": "2019-06-22T14:11:44Z",
  "credentialSubject": {
    "id": "did:ebsi-eth:00000002",
    "currentFamilyName": "Franz",
    "currentGivenName": "Hinterberger",
    "dateOfBirth": "1999-03-22T00:00:00Z",
    "placeOfBirth": "Salzburg, Austria"
  },
  "proof": [ {
    "type": "EcdsaSecp256k1Signature2019",
```

```
    "created": "2019-06-22T14:11:44Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:ebsi-eth:00000001#eidas-key-1",
    "proofValue": "BD21J4fdlnBvBA+y6D...fnC8Y="
  } ]
}
```

5.2. Example of an eSeal proof on an ESSIF Verifiable ID or Attestation

```
"proof": {
    "type": "EcdsaSecp256k1Signature2019",
    "created": "2019-06-22T14:11:44Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "did:ebsi-eth:00000001#eidas-key-1",
    "proofValue": "BD21J4fdlnBvBA+y6D...fnC8Y="
}
```