Ministry of the Interior and Kingdom Relations

NL Wallet Design Considerations

Version 1.0.3 draft

May 26, 2023

Contents

1 Introduction

Under the Working Agenda Value Driven Digitization [WA], the Dutch government is developing a digital identity wallet (henceforth called the NL Wallet). It will provide citizens with their own digital identity and control over their data, to be used in interactions with both the public and private sector. The first version of this wallet is to be released by the end of 2023 and piloted at small scale early 2024. The purpose, scope, high-level requirements, functionalities, initial assumptions, and dependencies of the NL Wallet project are described in the NL Wallet Project Start Architecture (PSA).

This document discusses several design considerations for the Dutch EUDI wallet based on the requirements listed in the PSA and those imposed by the European Architecture and Reference Framework [ARF] that is currently being developed. In this document we outline the challenges we face and the potential solutions we have considered for the NL Wallet and the technical and design choices that we deem to best fit these requirements.

This document can therefore be considered a prelude to the NL Wallet Project Solution Architecture Document [SAD], which takes the choices from this considerations document and further details them into a buildable solution.

2 Challenge: Holder binding

The NL Wallet should put the user in control of their identity and data, through the control of their personal device, a mobile phone. It must offer a high level of assurance to relying parties that the rightful user is actually in control of this device (holder binding), by using at least two authentication factors.

The following design decisions are made to ensure this holder binding:

1. The first authentication factor is possession of the user device, which is enforced by using a hardware bound key stored in secure hardware of the user device.
2. The second authentication factor depends on the type of situation:
    - In remote use cases, the user must enter a PIN.
    - In supervised proximity cases (with a human verifier), the portrait photo of the holder is shared with the human verifier.
    - In unsupervised proximity cases, the second authentication factor has yet to be decided. When the wallet is online, the PIN can be used. For offline use, we are still investigating other options for local PIN verification.
3. The verification of the PIN will primarily be done by a remote server operated by the Wallet Provider ('assisted' mode), because PIN brute forcing cannot be reliably prevented on most smartphones currently available. Once widely available local secure hardware offers a reliable option for PIN verification, this verification may be done locally (‘standalone’ mode).
4. To ensure a smooth transition from assisted to standalone mode, the protocols, data models, cryptography and software of the offline and online variants should be kept the same as much as possible.
5. Key attestation will be used to prove to the Wallet Provider that the possession factor is valid, i.e., that the key generated on the user device is actually hardware bound. Additionally, on iOS app attestation will be used because iOS does not allow using key attestation without it.
6. attestations will be issued in duplicate, one with a private key kept locally and one with a private key kept remotely. The attestations with private key kept remotely are used with remote PIN verification, those with the key kept locally are used with portrait photo comparison (TODO: or when local PIN verification can be done reliably).
7. The attestations for offline use will be short-lived to ensure that the impact of theft of device can be minimized.

2.1 Motivating the first authentication factor

For strong user authentication it is required that the user presents control over at least two independent authentication factors. We consider factors in the category possession, knowledge and biometrics.

The first factor can be the possession factor, implemented using the secure hardware of the device on which the wallet runs, which can generate secrets that are impossible to extract from. To ensure this possession factor, the wallet must be able to prove to the Wallet Provider that the key it generated is actually hardware bound. For this we use key attestation on Android and app and key attestation on iOS. This decision is motivated in section 2.5.

2.2 Motivating the second authentication factors

As to the second authentication factor, we decide that biometric sensors cannot be used (see section 2.2.1), leaving the following options depending on the usage situation:

- In supervised proximity use cases (i.e., person-to-person), the RP can itself verify the biometric second factor. This can be done for example by having the issuer issue a photo of the user as an attestation (part of the PID attestation or a (Q)EAA), which is then disclosed to the RP by the user, possibly along with other attributes. The RP then compares the photo with the person standing in front of them.
- In other use cases where the wallet has an internet connection, we will use the PIN as second factor. This is a common pattern in smartphone apps (e.g., other authentication apps or banking apps), so users are familiar with it.
- In unsupervised proximity cases where the wallet has no internet connection, the PIN must be verified locally. This is difficult if not impossible with the secure hardware that is widely available today. We are still considering options that are not yet described here.

2.2.1 Why we do not use biometric sensors

The NL wallet will not use the smartphone's biometric sensors (fingerprint or face ID) as a second authentication factor. This is partly because these sensors have shown[1] to be too easy to

[1] https://arxiv.org/abs/1809.03910
https://ios.gadgethacks.com/news/watch-identical-twins-fool-iphone-xs-face-id-0180855/
https://www.consumentenbond.nl/veilig-internetten/gezichtsherkenning-te-hacken
https://www.wired.com/story/hackers-say-broke-face-id-security/
https://www.forbes.com/sites/daveywinder/2019/11/02/smartphone-security-alert-as-hackers-claim-any-fingerprint-lock-broken-in-20-minutes/

circumvent. Also, in case your biometric data somehow leaks or is copied, then changing your biometry is impossible, in contrast with a knowledge factor which can be changed at will. Finally, by our knowledge, at least two other authentication applications that use a biometric factor in this way have had issues with it during a peer-review or audit for eIDAS High.[2] Note that the wallet may still use the smartphone's biometric sensors to unlock when the app starts.

2.3 Motivating remote PIN verification

For use cases without human supervision such as remote use cases, there must be a secure way to verify the user's PIN. In particular, since there are so few possible PIN combinations, it is critical that it is impossible to brute-force the PIN: after a number of wrong attempts (say, three), the user must be forced to wait some amount of time.

Since the aim of the project is to deliver within a year, this must be done with current generally available hardware, available to the majority of users. The majority of the currently available smartphones have no way of properly securing two authentication factors on their own (although in the future, this may change). For these devices we offer PIN validation by means of a remote online server. This is deemed the 'assisted' mode, as the wallet is assisted by a remote server.

We have considered and rejected three alternatives to this decision:

- The only secure hardware that is currently generally available to the majority of smartphone users is Apple's SE (Secure Enclave, not to be confused with a Secure Element, see also [SecEnc]), or Android's TEE (Trusted Execution Environment) or [StrongBox]. However, unfortunately neither of these offer a way to verify a PIN with a limited number of attempts.
- Using an (embedded) Secure Element (SE) on the smartphone, or an (embedded) Universal Integrated Circuit Card (UICC), e.g., a SIM card. These may be reprogrammed to support rate-limited PIN checking, if they don't already. The majority of the currently available smartphones do however not ship with those.
- Using a smartcard communicating with the smartphone over NFC. A significant fraction of smartphones is however not equipped with NFC, and additionally this would not make a very user-friendly or accessible UX (although for some users, the tradeoffs may be worth it).

[2] https://www.digitaleoverheid.nl/wp-content/uploads/sites/8/2020/03/Biometrie-voor-identiteitsverificatie.pdf Page 17

2.4 Motivating similarity between 'assisted' and 'standalone' modes

In the future, suitable secure hardware may be more widely available, so that the wallet may function completely locally, without the need for the online PIN server. Therefore, the protocols, data models, cryptography and software of the offline and online variants should be kept the same as much as possible, to facilitate a smooth transition to fully local wallets as soon as possible.

2.5 Motivating app and key attestation

To prove that the user is actually in control of some unique physical thing, during initialization of the wallet, their smartphone generates a unique key pair of which the private component is stored in a non-exportable way. In subsequent authentications, this private key is used to sign challenges to prove that it is still the same device.

To prove to the Wallet Provider that the key generated inside secure hardware, we use key attestation, a feature supported by iOS, Android TEE, Secure Elements (SE) and external secure hardware. During key attestation, the attestation public key is signed by a certificate guaranteed to be within the secure hardware. This ensures that the key is hardware bound, even if the mobile device's OS has been rooted or jailbroken before or after the creation of the key attestation.

For iOS, key attestation is only available in conjunction with app attestation. This means that the additional complications (described in section 2.5.2) that come with app attestation have to be dealt with.

2.5.1 Minimizing the privacy impact of app and key attestation

To minimize the effect on the user's privacy, we do not use key/app attestation on every authentication to a relying party, but only once: during the initialization of the wallet. After key/app attestation has been performed, the Wallet Provider issues an attestation to the device that key/app attestation has been performed successfully. This attestation is bound to a secret in the mobile device's secure hardware, which is verified during its issuance by the trusted authority using the app/key attestation. Using that attestation, the wallet can convince the PID attestation/(Q)EAA issuers and RPs as well.

2.5.2 App attestation and its drawbacks

With app attestation, the OS, using manufacturer (Apple/Google) online services, provides a signed attestation that the app is authentic and unmodified. This means creating an app attestation harms the privacy to some degree as it requires contacting Apple or Google. Additionally, app attestation has the following complications.

- It requires the user to be online.

- It requires relying on Google and Apple. Since those two are the only manufacturers offering this technology, using it makes the wallet as well as the surrounding software and ecosystem dependent and locked-in to these two manufacturers. If app attestation were required, it would become impossible to run the wallet on anything other than official iOS and Android smartphones: for example, FOSS Android forks such as LineageOS or /e/ would be excluded. This is undesirable.
- The security that app attestation truly offers can be doubted, since the attestation is at least partly provided by the mobile OS which cannot be trusted for such statements; its behavior may be altered after rooting/jailbreaking the device.

2.6 How to perform remote PIN verification (assisted mode)

For remote PIN verification we design a system in which the user's wallet holds the complete attestation, *except* the private key of the attestation. The private key of the attestation is stored instead at the PIN server, securely within an HSM. During a session the user first authenticates to the PIN server using their PIN, along with a challenge-response on an ECDSA private key specially for this purpose, stored within the mobile device's secure hardware. If this authentication succeeds then the wallet can instruct the PIN server to sign bytes using the attestation private key. This solution has the following properties:

- Within a session it should not learn the identity of the RP. This is best achieved by letting it communicate exclusively with the user, and not with the RP.
- It should not know the contents of any of the (Q)EAA/PID attestation attributes of the user, either when at rest in the wallet or when disclosed to RPs.
- It should be impossible for the user to perform a session without its cooperation (which it will only grant after a successful PIN verification).
- It should be impossible for the PIN server to impersonate the user.
- Ideally, the RP does not learn which PIN server is used within a session (there may be multiple within the system).

2.7 How to authenticate using the photo (supervised proximity cases)

In supervised proximity use cases, the wallet should be able to be offline so remote PIN verification is not an option. Furthermore, it is then impossible for the wallet to retrieve the remote private keys of attestations as proposed in the previous section.

Instead, for attestations that are to be used locally, their private keys will be stored in the wallet on device. This introduces a risk of abuse when the phone is stolen. To mitigate this risk, the attestations issued along with their private key must be short lived.

To enable both online and offline use of attestations, each attestation is issued twice to the the NL wallet:

- One with the private key stored at the PIN server;
- One with the entire attestation including the private key issued to the wallet.

The attestation should indicate to the RP which of these two versions is presented. This allows the RP to validate that a PIN was used as a second authentication factor. In the absence of such validation, the RP will need to use a photo attribute as the second authentication factor instead. Note: this ‘two version’ approach is not currently described in the ARF, and adding it might have implications for the common interface.

3 Challenge: Multishow unlinkability

The Dutch EUDI wallet aims to preserve the user's privacy as much as possible: it should strive to achieve the highest level of privacy that the use case supports. In particular, the wallet should support *multishow unlinkability*: when the user discloses the same set of attributes twice, the RP should not be able to tell if those two sets came from one and the same user, if the attributes themselves do not uniquely identify the user. That is, the RP cannot track the user solely because they use their attestations multiple times.

Part of the challenge is that the signature scheme ECDSA has to be used because this is the only suitable scheme that meets ARF requirements, i.e. being included on the SOG-IS list. We however advise to investigate other schemes such as BBS+ because they offer multishow unlinkability (see motivation in Chapter 7).

To resolve this challenge, the following design decision is made:

1. When using ECDSA, attestations are issued multiple times to the wallet (e.g. 10 times), each having a different unique signature. These attestations are shared with Relying Parties only once, ensuring the unique signature cannot be used to correlate multiple disclosures.

This chapter explains the motivation for this decision.

When using ECDSA, the signature over an attestation is unique, making multiple usages of the attestation linkable. To achieve multishow unlinkability, we follow the suggestion of Section E.8.4 of the ISO 18013-5 spec: issuing all attestations in multiple so that the user doesn't have to reuse attestation instances (or at least not often). That is, during every issuance session of the PID attestation or (Q)EAA, the user receives not one, but (say) 10 attestations. Of course, this specific number is up for further consideration and discussion. That way they can use each attestation only once to avoid the linkability that reusage would introduce – or alternatively reduce reuse of an attestation by randomizing which attestation is used in which disclosure.

This solution is not ideal: it introduces extra load for attestation issuers as well as additional complexity in the software. When using ordinary signature schemes such as ECDSA to sign the attestations, we see however no practical alternative. In the long term more sophisticated cryptographical schemes such as Idemix or BBS+ can help. These schemes offer multishow unlinkability out of the box in a more elegant way. However as these schemes currently do not meet the requirements of the ARF, they are considered out of scope for now.

3.1 Linkable attestation content

Note that multishow unlinkability is only useful if the attributes by themselves do not identify the user (e.g., a full name or a social security number). In case attestations are disclosed which by themselves completely identify the user (e.g., a social security number), then there is no reason not to reuse the attestation afterwards in other identifying sessions, because the sessions will be linkable anyway through the attributes themselves. Note that this would require the wallet distinguishing between linkable and unlinkable attestation content. This means that the cost in terms of load of this approach is difficult to predict, since it depends on how often sessions will contain uniquely identifying attestations.

3.2 External linkability factors

It is important to note in practice there will likely be other data that does link the user, such as their IP address, cookies, or browser fingerprints. Indeed, avoiding the presence of such identifying data altogether is in practice very difficult. On the other hand, one can argue that the sole fact that the internet is to a large degree privacy-unfriendly does not absolve us from the responsibility to design a privacy-friendly wallet. As much as can be reasonably be achieved, the situation should at least not be made *worse* by the wallet, so that it still makes sense for privacy-conscious users to protect their privacy by e.g. using Tor to hide their IP address and minimize their browser footprint. Additionally, if the privacy unfriendliness of the internet would be improved in the future by new technological developments, then we should prevent the wallet from being the least privacy friendly component that the user uses.

4 Challenge: Attestation Linking

The NL Wallet should allow the user to securely disclose multiple attributes from multiple attestations within a single session – for example, when the user wishes to disclose their name from the PID attestation together with a diploma obtained from a university attestation issuer. This must be done in such a way that the RP (Relying Party) can convince itself that even though the attestations originated from different attestations, they were issued to a single wallet, and to a single user.

Part of the challenge is that the signature scheme ECDSA has to be used because this is the only suitable scheme that meets ARF requirements, i.e. being included on the SOG-IS list. We however advise to investigate other schemes such as BBS+ because they offer multishow unlinkability (see motivation in Chapter 7).

To resolve this challenge, the following design decisions are made:

1. When using ECDSA, every pair of attestations that must be linked together both contain the same 'linking attribute' (a unique number). When disclosing the two attestations together, both the linking attributes are disclosed to prove that the attestations are linked. The relying party must enforce that these attestations have the same linking attribute.
2. attestations are provided with several linking attributes instead of one, to ensure that the linking attribute cannot be used to correlate multiple disclosures of the same attributes. In other words, this restores multishow unlinkability.
3. An alternative is considered but decided against: the Wallet Provider knows that multiple attestations belong to the same user and can attest to this fact in such a way that the relying party can verify this. This is decided against because it is only possible in online use cases and would put extra liability on the Wallet Provider.
4. When BBS+ could be used as a signature scheme, every attestation would include a 'link secret'. This is a single large number that is never shared with any party to avoid correlation. The wallet can however prove to the relying party that two attestations are linked by proving in zero-knowledge that both attestations contain the same link secret. Given the fact that currently available secure hardware does not support BBS+, cooperation of the remote server would be required to achieve hardware binding.

The remainder of this chapter motivates these decisions:

4.1 How to use linking attributes

Without attestation linking, two users Alice and Bob users could join their attestations into a single wallet, and pool their credentials: they could disclose for example that Alice is male, if both of them

have an attestation containing their name and sex. Neither the mdoc speficiations (ISO 18013-5+22320-3/4) nor VC SD-JWT provide a mechanism for linked attestations that prevents such attacks.

We propose a security mechanism for use in the NL wallet that allows the issuer to bind two attestations together using a random and unique *linking attribute* that is included in each attestation: the issuer can link its new attestation to another existing one in the wallet by using the same linking attribute as that of the existing attestation. For example, this allows a (Q)EAA issuer to bind its credential to the user's PID. This allows the RP to defend against credential pooling by enforcing that the link attributes of the attestations that it receives are equal. To prevent the linking attributes from having a privacy impact (since by their uniqueness they could be used to distinguish and identify individual users), we propose that wallets are issued multiple copies of an attestation by the issuer (as introduced in the previous chapter), that each use random and distinct linking attributes. This prevents the user from needing to re-use already used attestations and linking attributes, so that they can use fresh and random ones each time (at the cost of periodically having to fetch fresh attestations from the issuer).

Schematically, our approach looks as follows.

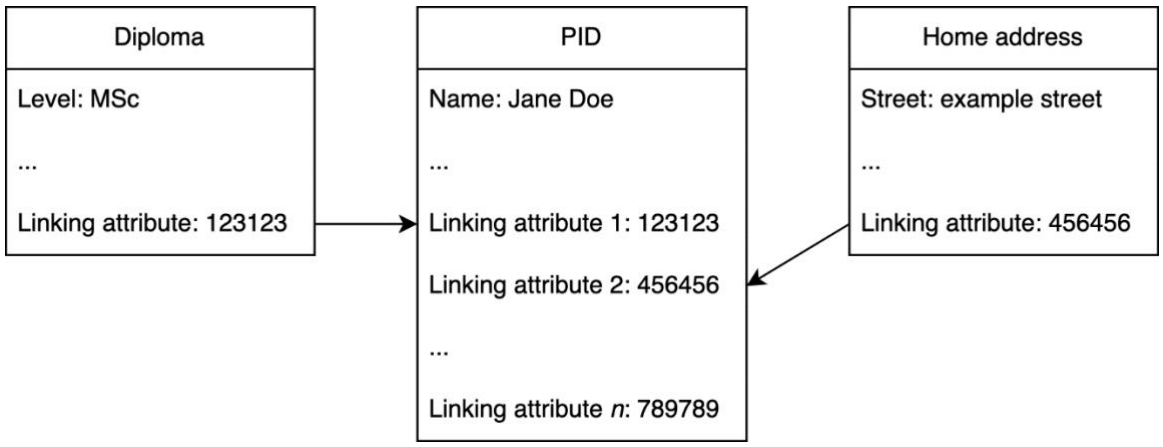

Figure 1: Two attestations being bound to a PID attestation using linking attributes

4.1.1 Detailed explanation

In more detail, the proposed mechanism works as follows. Although the mechanism allows any attestation to be bound to any other attestation, for now we assume that a (Q)EAA issuer wishes to bind its new credential to the user's PID attestation.

- In the PID attestation, the PID attestation issuer includes *n* linking attributes, each having random distinct values of sufficient length (say, 256 bits). These are ordinary attributes in

the sense that during usage of the attestation, the attribute may or may not be disclosed, like the other attributes in the attestation. Additionally (as explained in the previous chapter), within the session the PID attestation issuer issues *m* attestation instances to the user, each having the same set of *n* linking attributes. (Here, *n* and *m* are integers which are to be determined. They may be made configurable for the issuer so that it can tweak them if and when necessary. Probably *m* should exceed *n*, since a PID attestation instance is needed not only when disclosing linked (Q)EAAs but also when disclosing PID attestation attributes by themselves.)

- During issuance of a (Q)EAA, the user's wallet automatically chooses one of the linking attributes from their PID attestation, and discloses that to the issuer. The issuer includes that linking attribute in its own attestation. As with the PID, the (Q)EAA issuer issues multiple copies of the attestation. The user uses different linking attributes from the PID attestation for each of them.
- When the user wants to disclose attestations of this (Q)EAA, they must always disclose the contained linking attribute. Additionally, they pick one of their PID attestation instances, and from that disclose the corresponding link attribute. The RP accepts only if the linking attribute from the (Q)EAA is also contained within the set of disclosed PID attestation linking attributes. The user uses neither the (Q)EAA nor the PID attestation a second time.

This makes the attack from the previous section impossible. If Alice were to try to disclose:

- attestation 1: "Alice" and Alice's linking attribute,
- attestation 2: "male" and Bob's linking attribute,

Then the RP will reject because the two linking attributes will not coincide. Effectively, all attestations are bound to each other through the linking attribute.

4.2 How to ensure unlinkability

The mechanism presented above keeps intact the multishow unlinkability obtained through issuance of multiple attestation copies, as introduced in the previous chapter. A PID attestation instance is used each time a (Q)EAA is disclosed, but each time the user chooses a PID attestation instance that they have never used before. Distinct (Q)EAAs use distinct linking attributes, so that even though the linking attributes need to be disclosed when using the (Q)EAAs, they cannot be used to track or identify the user in such cases. However, once a particular (Q)EAA has been disclosed, neither it nor its linking attributes must be reused to maintain unlinkability.

This solution is not ideal: it introduces extra load for attestation issuers as well as additional complexity in the software. When using ordinary signature schemes such as ECDSA to sign the attestations, we see however no practical alternative.

4.3 Alternative: relying on the Wallet Provider

Alternatively, the RP can rely on the Wallet Provider to attest to the fact that the presented attestations belong to the same user. As explained in section 2.5, the remote server holds all private keys of attestations. The Wallet Provider needs only sign a message stating that certain public keys of attestations belong to the same user. The user's wallet can then forward this information to the relying party.

We decided against this alternative for two reasons:

1. This option is only usable when the wallet is online.
2. This option requires additional trust from the relying party in the wallet provider. In other words, it introduces a liability for the wallet provider because it has to attest to the relation between attestations.

If this approach is taken, it could operate as follows. An authentication system such as this always includes a challenge-response mechanism, in which the user uses their private key to sign a random challenge created by the RP. This convinces the RP that the user is not trying to perform a replay attack. When multiple attestations are used, then the user has to sign the challenge using the private keys of all attestations. Now when a PIN server is used, then this signing is done by the PIN server, since it holds the user's private keys. In this setting, the mechanism works as follows. To link the attestations together, the PIN server signs not only the RP-provided challenge, but it also includes the public keys of all used attestations in the message that it signs. That is, it signs an (appropriately encoded) tuple that looks as follows: `(challenge, publicKey1, …, publicKeyN)`, where the public keys come from the attestations. If the RP notices that the attestations that it receives use a PIN server, and the message that is signed has this structure, then it can infer that the attestations whose public keys are included in the signed message belong to the same user, since only in that case would the PIN server have constructed such signatures. Alternative: using proofs of knowledge with BBS+

Alternative cryptographic scheme such as BBS+ or Idemix provide more elegant and efficient options for attribute linking. When BBS+ could be used as a signature scheme, every attestation would include a 'link secret'. This is a single large number that is never shared with any party to avoid correlation. The wallet can however prove to the relying party that two attestations are linked by proving in zero-knowledge that both attestations contain the link secret. Given the fact that currently available secure hardware does not support BBS+, cooperation of the remote server is required to achieve hardware binding.

5 Challenge: Wallet Recovery

When a user loses access to their wallet, they need to be able to recover their wallet easily. To resolve this challenge we made the following design decisions:

1. The wallet provider binds the wallet account to a pseudonym derived from the BSN (using BSNk[3]) of the user which is specific to that wallet provider, to minimize what the wallet provider knows.
2. A user's attestations and logs will be stored in encrypted form at a cloud storage location of a user's choosing. Optionally, the user may altogether opt out of backup.[4]
3. When the user first initializes their wallet, they will present a pseudonym from BSNk to the wallet provider. The wallet provider then creates a wallet account bound to this pseudonym. When the user later initializes their wallet anew or on another device, they will present the same pseudonym. The wallet provider can then see that the user already has an account and can offer to restore the contents of their wallet.

The rest of this chapter motivates these decisions:

5.1 Motivating the use of pseudonyms

Instead of a pseudonym, the user could present their PID attributes like the BSN. The wallet provider could bind their account directly to the BSN. This would however imply that the wallet provider knows the identity of the holder, which may be undesirable.

5.2 Motivating remote storage for recovery

In order to be able to recover not only attestations, but also the transaction history of a lost wallet, these data should be stored somewhere external to the device. For most users it is likely preferable that this data is backed up automatically instead of manually, which requires the external backup provider to be accessible at all times. These external backups should obviously be properly secured and not accessible to the backup providers themselves.

[3] BSNk is a public service that offers pseudonyms derived from the BSN. This allows citizens to use different unrelatable pseudonyms with different relying parties. It also offers a persistent identifier towards a relying party, as it is always derived from the BSN which usually never changes.
[4] We note this is subject to further policy discussion.

5.3 Motivating the use of multiple cloud storage providers

Saving data remotely likely leads to large collections of encrypted data stored at a central location which may be a security hotspot, i.e. a breach of such external storage impact a large amount of users. To reduce this risk, the user may choose between a selection of cloud storage providers either operated by a third party or by the user themselves.

5.4 How recovery works

After their wallet is initialized and the PID is issued, the wallet requests the BSNk pseudonym provider for a pseudonym this user specifically for the Wallet Provider. As the pseudonym is derived from the BSN and the OIN of the Wallet Provider, the returned pseudonym will always be the same. Therefore, the wallet provider can check whether a wallet account has already been registered under that pseudonym.

If the pseudonym was not yet registered, this means the user creates a wallet for the first time. At this point, the user may be asked to pick a recovery option. Specifically, they must decide on a cloud storage provider they trust with storing their backup. A default option should always be available (either randomly chosen or designated), to avoid that users skip this configuration and end up without recovery possibilities.

If the pseudonym was already registered and recovery of the original wallet was enabled, the user may choose to recover the contents of this wallet. This means the attribute attestation private keys managed by the wallet provider are linked to the new user account and the attestation and log data is retreived from the selected cloud storage provider. The user has now regained access to the contents of the original assisted wallet.

6 Challenge: Wallet Blocking

A user should be able to block their wallet. Therefore a user should be able to reliably authenticate towards the Wallet Provider, without access to the wallet itself and whilst revealing as little information as possible.

Although we haven't yet made a decision on how to implement wallet blocking, we have considered the following mechanisms:

- *Revocation passphrase/link*
  As part of the assisted wallet creation the user is provided with a wallet blocking mechanism, e.g., a wallet blocking link similar to the revocation link discussed in the previous section.
- *Using directly identifying PID attributes, e.g. the BSN*
  After wallet personalization, the user uses its (fresh) PID to disclose a uniquely identifying PID attribute set to the wallet provider. The wallet provider links the user wallet account to this PID attribute set. A convenient such attribute set is the user BSN. The wallet provider next supports blocking mechanisms based on the user being able to prove they are the person this attribute set corresponds to. This could for instance be based on a PID in a different wallet or on a face-to-face process. If the attribute set used is simply the user BSN, a user could also logon to the wallet provider using DigiD (or any other Dutch recognized authentication means) to block the wallet. Actually, any notified European authentication means could be used to block the wallet as these also support authentication based on BSN.
- *Using pseudonyms*
  The use of directly identifying PID attributes at the wallet provider implies that the user data the wallet provider processes becomes more sensitive and riskier. This can be mitigated by using pseudonyms instead of PID attributes. To this end, after wallet personalization, the user requests a pseudonym attribute for the Wallet Provider at the BSNk attribute provider. The user then discloses this pseudonym to the wallet provider that links the user wallet account to this pseudonym. The wallet provider next supports blocking mechanisms based on the user being able to prove that they are the person this pseudonym corresponds to. This could for instance be based on a pseudonym attribute managed in a different (new!) wallet. Also, as the wallet pseudonyms are designed compatible with the pseudonyms used within the Dutch eID-scheme a user could also logon to the wallet provider using DigiD (or any other Dutch recognized authentication means) under pseudonym to block the wallet. Actually, any notified European authentication means could be used to block the wallet as these also support authentications based on pseudonym.

Due to its nature, there is no support to block the stand-alone wallet other than the user removing the wallet from their mobile device. However, the user can revoke all the attribute attestations they managed in their stand-alone wallet as discussed in the previous section.

7 Decision: Signature Scheme Support

The integrity and authenticity of personal identification data (PID) and attestations is guaranteed by means of electronic signatures. This requires deciding on a signature scheme that is supported by issuers, wallets and verifiers. It is therefore an ecosystem level decision that affects interoperability. Signature schemes have different properties and trade-offs. These trade-offs are discussed here.

We conclude the following:

1. The NL Wallet will at least support the signature scheme that is specified by the ARF. We assume the most likely candidate is ECDSA, given the following reasons:
    a. It is registered on the SOG-IS list of approved schemes which is a requirement from the ARF.
    b. It is currently the only supported scheme in the mdoc specification.
    c. It is currently the only scheme supported by widely available mobile secure hardware.
2. We consider Idemix and BBS+ to be valuable alternatives. The main benefits are multishow-unlinkability (see Chapter 3) and efficient attribute linking (see Chapter 4). However, these schemes are currently out of our development scope due to reasons listed under (1).
3. We consider BBS+ a more valuable candidate than Idemix because it is more efficient and modern. We therefore advocate further investigating this scheme and propose to add it to the SOG-IS list.
4. We consider a limitation of BBS+ to that currently available secure hardware in mobile phones does not support this, requiring the cryptographic functions to be performed externally. This in turn limits the offline capabilities of the wallet.

As discussed in the previous sections, there exist other cryptographic schemes that deal with issues like unlinkability and attestation linking more elegantly, primarily [Idemix] and [BBS+]. Because these are not on the SOG-IS list they are considered out of scope for now. However, their interesting qualities make them worth considering.

7.1 Background on Idemix and BBS+

Idemix was introduced in the early 2000s, and uses RSA-like cryptography (the issuer's public key is a 2048 or 4096-bit product of two prime numbers and the corresponding private key consists of those two primes). BBS+, dating from 2004 and revised in 2016, operates on elliptic curves leading to greater efficiency. In terms of features, the two are however nearly identical:

- Using an interactive protocol, an issuer can digitally sign a set of attributes, creating an attestation, and give that to a user.

- Before disclosure, the user can *randomize* the issuer signature over the attributes, creating a new and never-before signature that is still valid over the attributes of the attestation.
- During disclosure, the user can hide attributes present in the attestation that are not relevant to the session using zero-knowledge proofs, achieving selective disclosure.
- If two non-disclosed attributes have the same value, then the user has the ability to prove that equality in zero-knowledge, i.e., without disclosing them or providing any information about the attributes other than equality to the RP. This mechanism allows for a much simpler and efficient way to bind multiple attestations within the wallet than the linking attribute mechanism discussed in the previous chapter, as follows. Before issuance, the user generates a large random number (called the user's *secret key*). The user never communicates this number to anyone in any way. During issuance, the user ensures that this number is included in the attestation as an attribute, without disclosing this number to the issuer. The user uses this same number across all of their attestations. Then, when the user discloses attributes to an RP out of two or more attestations, they prove to the RP in zero-knowledge that this number has the same value in each of the used attestations. From this, the RP can infer that all of those attestations belong to one and the same user and wallet. In a sense, this secret key therefore acts as a sort of keyring that binds all of the user's attestations together.

The second and third point leads to multishow unlinkability (see Section **Error! Reference source not found.**). In fact, even the issuer cannot track usages of its own attestation even if it were to collaborate with RPs. This property is as far as we know impossible to achieve when using conventional cryptography such as ECDSA, and is a privacy improvement over ECDSA-based schemes. Of course, this only holds if the disclosed attributes by themselves do not identify the user. Thus, if the user twice discloses an attribute stating for example that they are over 18, which lots of other people will also have, then the cryptography of Idemix and BBS+ do not grant the ability to the RP to link those two sessions as coming from the same user (not considering other possible identifiers such as IP addresses or cookies).

7.1.1 Idemix vs BBS+

Idemix and BBS+ compare as follows.

- BBS+ is more modern and significantly more efficient than Idemix, since it uses elliptic curves instead of RSA-like cryptography (although in practice, the difference is not noticeable on modern smartphones).
- In BBS+, generating an issuer private key is straightforward: any number between 1 and an upper bound can serve as an issuer private key. In Idemix, however, to fully achieve the unlinkability property mentioned above the two prime numbers that constitute the issuer's private key have to have a special property: they have to be so-called *safe* primes, meaning

that if p is the prime number then $(p - 1)/2$ must also also prime. Otherwise, the issuer has the ability to break the unlinkability property, i.e., track usages of its attestation by collaborating with RPs. To convince wallets and RPs that it chose the primes correctly, the issuer would during key generation have to generate a special zero-knowledge proof that can convince wallets and RPs that the primes are chosen correctly. The only implementation of this that we know of, by the [IRMA] project, produces proofs of some 700 MB for 2048 bit keys. This almost certainly makes it impossible to use Idemix private keys inside an HSM.

- There exist BBS+ implementations used in VCs, contrary to Idemix.

We note that Idemix and BBS+ are not the only two cryptographic schemes that use zero-knowledge proofs to achieve multi-show unlinkability; several others exist in the computer scientific literature. However, these two have received by far the most attention and implementations. Additionally, none of them that we know of are significantly more efficient or otherwise better than BBS+.

7.2 No mobile hardware support for BBS+/Idemix

The secure hardware of modern mobile devices (Apple's SE or Android's TEE/StrongBox) do not support Idemix or BBS+, and this is unlikely to change anytime soon. If the cryptography is done on currently available mobile devices it must therefore be done in software. This means that strong device binding (i.e., binding a part of the credential to the device's secure hardware in such a way that it can impossibly be extracted from the device) is difficult to achieve, which in turn means that it will be difficult to achieve eIDAS high or even substantial in such use cases.

This might be solved by letting a remote server operated by a trusted entity (in practice, probably the wallet issuer) manage the BBS+/Idemix secret keys of the attestations of the user, as suggested in more detail in [Idemix-eIDAS-High]. In this setting, the user's wallet authenticates itself using a hardware-bound ECDSA key to this server, after which the wallet and server jointly compute an attestation disclosure. Neither the wallet nor the server has the ability to compute disclosures without the cooperation of the other. The server knows only the attestation secret keys and not the attributes themselves, and indeed there is no technical necessity for this server to know anything about the user apart from a random identifier and their keys. Additionally, it communicates exclusively with the user and never with RPs, so the server does not get to know to whom the user disclose their attributes. Thus, the privacy impact of using such a server is limited – although it is not zero: the server does get to see the user's IP address and when the user uses their attestations, and if it were to collude with RP's, then together they could deanonymize the user.

Using an online server in this fashion would mean that Idemix and BBS+ can only be used in this setting in online scenarios. This situation might change if the mobile device gains support for these algorithms, or if they are equipped with a programmable Secure Element. In the meantime,

supporting offline use cases will require falling back to cryptography supported by the secure hardware of currently available mobile devices, i.e., ECDSA. This means that the wallet would have to support BBS+/Idemix together with ECDSA, increasing the complexity and maintenance burden of the wallet.

7.3 Open issue: revocation with Idemix/BBS+

The Verifiable Credentials standard as well as (probably) the mdoc standards (ISO 18013-5+22320-3/4) will use the [VC-SL21] standard for revoking attestations. This standard works in short by including a unique identifier in the attestation, which is then published on a blacklist when the credential is revoked by the issuer. During disclosure, the RP checks that the unique identifier in the attestation is not on this blacklist.

Using such a unique number to track revocation in Idemix or BBS+ credentials would break the unlinkability property of these schemes. Therefore, most implementations use different cryptographic revocation mechanisms, called accumulators. The attestation issuer generates this accumulator, keeps it up to date, and is responsible for making it available to wallets and RPs. These accumulators keep intact the unlinkability properties of Idemix and BBS+, by allowing the user to prove in zero-knowledge that their credential has not been revoked.

Multiple cryptographic accumulators exists in the scientific literature and in implementations, but as far as we know the RSA-B accumulator is the only one that has the significant advantage that its value does *not* have to be updated by the issuer every time it issues an attestation to a user. In terms of feasibility, this makes an enormous difference. However, like Idemix it uses safe prime composites, leading to the necessity of large correctness zero-knowledge proofs discussed above in Section 7.1.1. Additionally, it is not currently known if this accumulator can be used together with BBS+.

7.4 Conclusion

Although BBS+ and Idemix bring their own complexities, they have several significant advantages over ECDSA-based schemes. Additionally, they have already been implemented in existing wallets that are used in production settings (although none of these are certified as supporting eIDAS high, that we know of), showing that it can be done. For these reasons, we suggest placing increased focus on using BBS+ or CL implementations (or schemes with similar unlinkability features).

8 Technology choices

8.1 Developing support for mdoc first

The ARF requires the wallet to support the mdoc standard (ISO 18013-5+22320-3/4), as well as the combination of the [VC], [SD-JWT] and [OpenID4VC] standards. These compare as follows.

- The mdoc standard supports close proximity flows over Bluetooth/NFC/Wi-Fi direct, as well as online use cases. By contrast, VC-SD-JWT/OpenID4VC only supports online use cases.
- The ISO 23220 series is not yet finished, but mdoc is for a large part defined by ISO 18013-5 which is finished and stable. By contrast, the SD-JWT and OpenID4VC specs are relatively new and currently under active development, and have changed to a significant degree in the past months.

Given the limited capacity of the development team, the large scope of the project, and the challenging deadlines, we have decided to not develop support for both of these simultaneously, but one at a time. Implementing mdoc results in more supported use cases, and should additionally be easier and result in greater interoperability with other implementations because it is more stable. For these reasons, we have decided to first focus on developing support for mdoc credentials, and VC-SD-JWT/OpenID4VC after that.

To smooth the development process when we do start implementing the VC-SD-JWT/OpenID4VC standards, we will keep close tabs on the development of these standards. Additionally, during the entire development process of the wallet we will continuously take into account that the software and protocols will also need to be able to support VC-SD-JWT/OpenID4VC in the near future.

8.2 Choice of development stack

One of the fundamental choices of every development project is to choose a development stack. For apps, the biggest choice is whether to use native technology to build the apps or to use one of the cross platform or hybrid stacks. In this document we lay out the arguments for the choices for the NL wallet implementation.

8.2.1 Comparison of native and hybrid/cross-platform approaches

The following general arguments exist for a native approach:

1) Since the apps serve as an example / reference we should stick to the most common development paradigms which is native.

2) We will rely on OS level features. Any features that are relevant for the wallet (such as the introduction of a new biometric device) will be made available first on the native platforms

and only later (if at all) in hybrid / cross platform solutions. In other words, cross-platform frameworks are designed to reduce the TCO for commodity applications. Wallets aren't there yet and won't be. That bakes in risk.

3) Reuse and review between other EU member states and our implementation is likely to be easier if we choose the more common technology platform[5].

4) Similarly, reuse of the EU reference implementation will be easier if we use the same stack. The EU reference wallet will be native, as can be seen in the EU requirements (see annex B)

5) It will be easier to attract developers if we stick to the native implementations.

6) If the need arises to consult with the platform vendors (Apple and Google) there will be less friction if we use their tooling directly and not through an abstraction layer.

7) We want to make use of the accessibility features that the platforms offer. Although some hybrid solutions offer a wrapper for these, it might be easier for accessibility specialists to work with the native functionality.

8) For the end user we want to provide an experience that is seamlessly executed on their platform of choice. Although most hybrid solutions mimic or reuse native UI elements, they tend to either consolidate the UI across platforms or lag behind when the OS makes subtle changes (for example, 'tap title bar to scroll to top' is a standard UI gesture in iOS, but requires manual labor in react native and is not available in flutter).

9) Finally, hybrid and cross platform technologies come and go and not always stand the test of time. PhoneGap / Cordova has waned, Xamarin and React Native have both lost ground to

[5] The following text is taken from the tender for the EU reference wallet:

"For the core of the Wallet eco-system the Contractor will develop, maintain, and continuously improve fully interactive, **native mobile apps for both Android and iOS platforms** that are built using the platform-specific technology stack and development tools defined by Google (Java, Kotlin, Android Studio, Android Developer Tools, Android SDK) and Apple (Swift, XCode IDE) respectively. Later releases may include the possibility to add third party HTML, JS, and CSS based hybrid applications rendered on the mobile apps with platform specific customizations of view and navigation that provide extra functionalities."

Flutter. For apps that have longevity as one of their properties, choosing the 'technology of the moment' might later hurt the project.

10) Our main competition are Google/Apple, the platform holders. Their wallets are built natively and they have the advantage of being platform holders and can't be assumed to be cooperating. We're already at a disadvantage, building cross-platform increases that disadvantage.

11) When targeting a high level of OS integration and performance any cost benefits enabled by a cross-platform solution will likely be eaten up by leaky abstractions / impedance miss-matches.

12) The minimum OS version requirements of the application can be satisfied by native applications, whereas some hybrid / cross platform applications set goals that might be less ambitious (e.g. iOS 11-13 is 'best effort' support in Flutter, iOS14+ is fully supported, while iOS 11/12 are required to be supported if the iPhone 6/6S are to be included in the targeted devices). Note that in the case a full-fledged framework is used, these OS requirements are not only for the core library, but also for any needed extensions. For example, the Flutter NFC wrapper requires iOS13+.

The arguments for a hybrid or cross platform approach are:

1) It can save time as code needs to be written only once. Although this is only partially true as the Ui would still require changes across the platforms and testing still needs to be done on both platforms, there is still an expected efficiency in development time by using reusable code. (See Annex B for some best practices to reduce the amount of effort it takes to develop natively)

2) A smaller team could implement the frontend. Smaller teams reduce synchronization overhead and therefore costs.

3) Potential to increase reusability for industry or smaller EU countries. Having a single code base reduces the requirements/costs for forks or combinations. Similarly targeting a web-tech hybrid reduces the quality level of developer required (React/JavaScript developers are a dime-a-dozen and even vaguely qualified developers can be productive).

4) When a web-tech hybrid approach is used then there will be more reuse possible with other wallet modalities - for example web wallets.

Aside from these arguments, there are a number of other considerations to take into account:

1) Existing skill set in a team
2) How many platforms should be targeted (the more platforms that need to be targeted, the bigger the benefit
3) What other member states will be using.

8.2.2 Decisions for the wallet apps

Based on the arguments presented the previous section, we have come to the following approach:

1. The 'engine' (the cryptographic heart of the wallet) will be developed as a 'shared core' module, which is cross-platform.
2. We will continue to support a Flutter implementation, targeting all platforms where we don't deem a native app necessary.
3. We implement a native iOS implementation, considering the arguments above about version support and closeness to the UI that iOS users expect, to be decisive factors to want a native implementation.
4. At a later stage, we add other native implementations if requirements surface that warrant such an implementation. The arguments that the wallet should serve as an example and relies heavily on device specific features such as interaction with the SE or TEE of a device, suggests we should also consider creating a native Android implementation.

8.3 Shared core language

For the shared core of the wallet, we have considered various option for a programming languages:

- Rust
- Go
- C++
- Kotlin Multiplatform

From these three we have chosen the Rust programming language. A core library written in Rust can be compiled to a library that can be reused in both native and Flutter apps.

We have decided against Go because it produces significantly larger libraries than Rust.

We have chosen not to use C++ because it would add complexity and limit the amount of people that can work on this core.

Kotlin Multiplatform limits implementation to Android and iOS, whereas Rust can also be reused in other operating systems.

8.3.1 Backend / middleware stack

For the backend we choose a 'best tool for the job' approach, where the technology is determined by tooling that we reuse. E.g., an OIDC/SAML bridge may be written in Python because we can reuse a good open source component for it and mostly requires configuration, whereas other choices might be made for other components.

For 'core wallet' functionality such as issuance, we will use Rust, because of symmetry with the shared core inside the wallet app. By choosing the same language, we can easily make test suites that test the issuance and consumption process in one test.

For the Backend For Frontends (BFF) that forms the glue between all backend services and the frontends, we will choose Java, as it is a proven enterprise stack that is commonly used in government projects and likely to find hosting support for.

9 References

Reference	Description / Link
[ARF]	eIDAS Expert Group, EDI Architecture and Reference Framework, work in progress.
[BBS+]	Man Ho Au, Willy Susilo, and Yi Mu. *Constant-Size Dynamic k-TAA*. SCN 2006, LNCS vol. 4116. Jan Camenisch, Manu Drijvers, and Anja Lehmann. *Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited*. https://eprint.iacr.org/2016/663
[EDI-Values]	Waarden, kansen en uitdagingen rond het Europese Digitale Identiteit raamwerk, https://open.overheid.nl/documenten/ronl-0662761f27b090409a5b68476bf4e550c69c9562/pdf
[eIDAS-LoA]	Guidance for the application of the levels of assurance which support the eIDAS Regulation
[eIDAS]	The European Parliament and the Council of the European Union, Electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC, regulation 910/2014, 2014.
[eIDAS-1502]	European Commission, eIDAS implementing regulation 2015/1502, 8 September 2015.
[eIDAS-EDI]	European Commission, Regulation of the European parliament and of the council amending Regulation (EU), No 910/2014 as regards establishing a framework for a European Digital Identity, 2021/0136, 3 June 2021.

[Idemix]	Jan Camenisch and Anna Lysyanskaya. *An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation*. In: Advances in Cryptology — EUROCRYPT 2001, LNCS vol. 2045. IBM. *Specification of the Identity Mixer (Idemix) Cryptographic Library*
[Idemix-eIDAS-High]	https://github.com/WebOfTrustInfo/rwot11-the-hague/blob/master/advance-readings/combining-eIDAS-High-with-unlinkability.md
[IRMA]	The IRMA Project: I Reveal My Attributes. https://irma.app/, https://github.com/privacybydesign/irmago
[ISO 18013-5]	NEN-ISO/IEC 18013-5, Personal identification – ISO-compliant driving licence – Part 5: Mobile driving licence (mDL) application
[OpenID4VC]	The OpenID4VCI (issuance), OpenID4VP (disclosure), and SIOPv2 standards: https://openid.net/openid4vc/
[SD-JWT]	Selective Disclosure for JWTs: https://datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/
[StrongBox]	https://source.android.com/compatibility/11/android-11-cdd
[SOG-IS]	SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms: https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.2.pdf

[VC-SL21]	Verifiable Credentials Status List 2021, a standard for efficiently maintaining revocation lists.
[WA]	Value Driven Digitalisation Work Agenda, https://www.government.nl/documents/reports/2022/11/30/value-driven-digitalisation-work-agenda