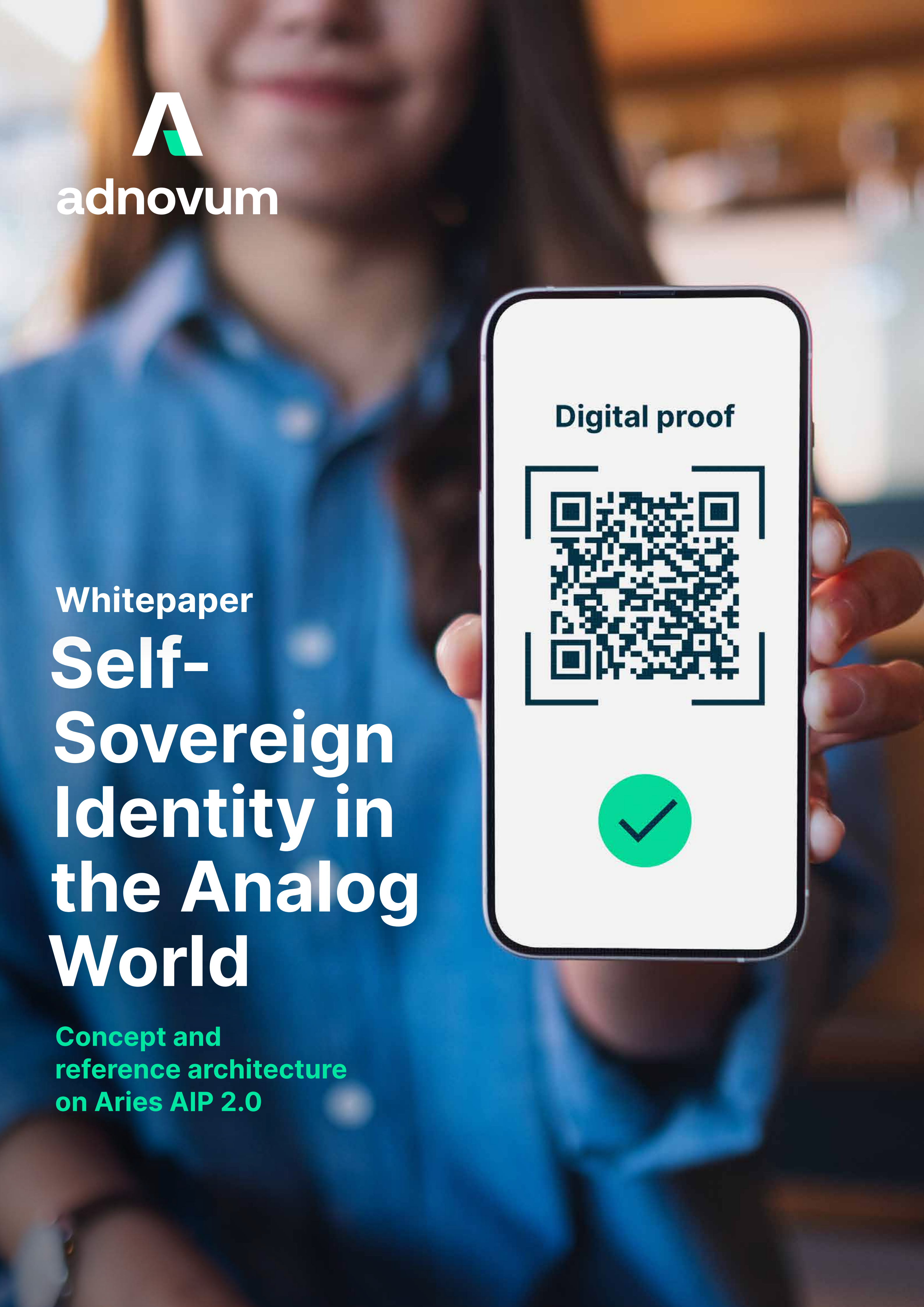
adnovum



Whitepaper

Self-Sovereign Identity in the Analog World

Concept and reference architecture on Aries AIP 2.0

Self-Sovereign Identity in the Analog World

Content

→ Conclusion → Author → Glossary → References

①

Introduction

Self-sovereign identity (SSI), a new generation of digital identity, is on the rise and experiencing global hype. It lets users regain privacy and control over their personal data by giving them the possibility to create and digitally store their identity on their smartphones and use it in lieu of its physical counterpart without relying on any centralized authority.

Governments and private companies look to it as a technology for the long overdue digitization of analog documents such as ID cards, passports, driver's licenses, etc. The digitization of official identities and of verification processes is one of the final steps on the road to greater automation and process simplification. Applications to government agencies will be possible without the hassle of visiting a counter in-person, and services requiring verification of residency or identity will be simplified and processed without multi-day delays.

Not for the online world only

Current SSI solutions focus on the online experience and on ways to integrate SSI into multiple online services through already available versions of mobile wallets. The attempt to seamlessly fit new digital identity representations into daily life, however, is still fraught with serious shortcomings in terms of convenience and ease of use.

Going one step further

Having successfully demonstrated the applicability and integration of SSI as a novel solution for online identity management in our SSI initiative [1], we decided to go a step further and analyze how the advantages of the SSI ecosystem of trust can be extended and transferred to the physical world.

This white paper explores a way to increase security, convenience, and interoperability in the use of mobile wallets for physical identity interactions with people or machines in close proximity through the next generation of Aries RFCs (AIP 2.0).

We moreover propose a reference software architecture optimized for SSI capability in mobile applications and test it in a mobile wallet implementation for Android.

②

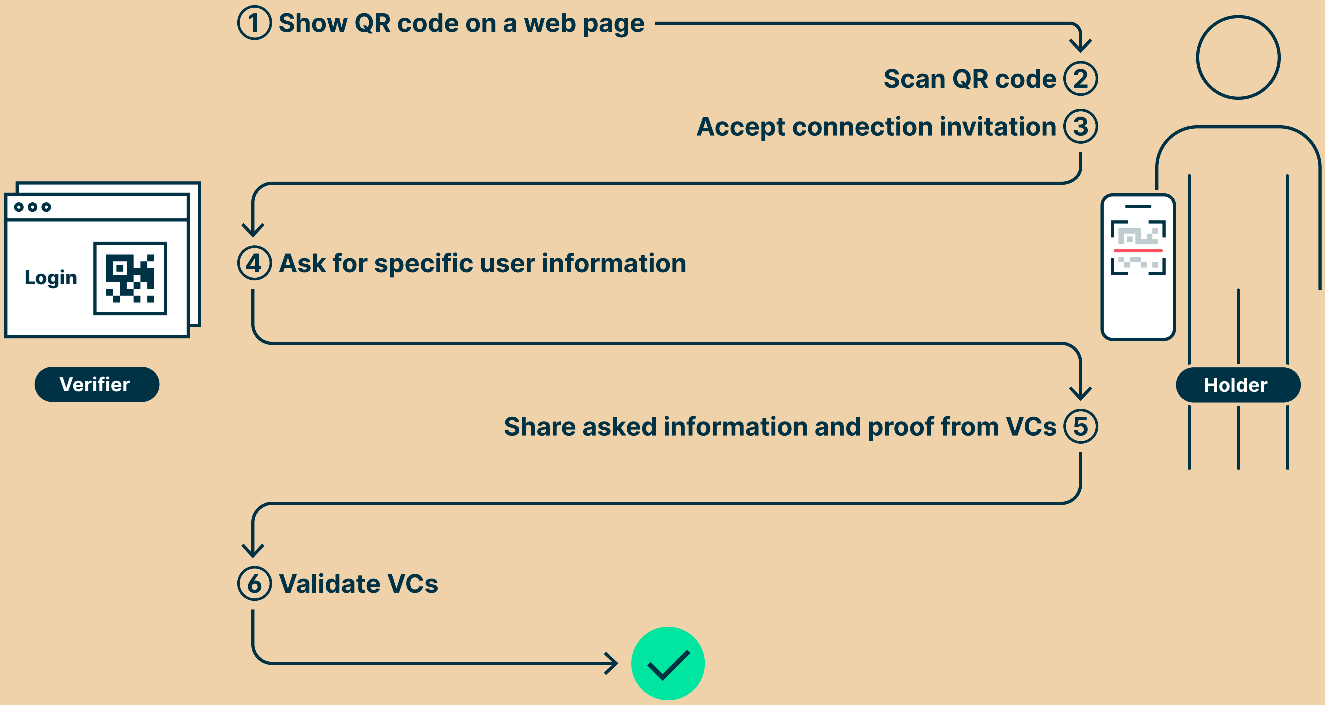
Starting Point

Customary use of SSI for online authentication

SSI lets users take control over their identity and enables a variety of use cases for trusted online interactions with others and for the mutual validation of identity. The first and most common of these use cases is to authenticate yourself by using your SSI wallet to scan a QR code proposed by the online service you want to access.

The QR code contains meta information such as public cryptographic keys to secure the connection and can be considered a secure invitation link. The user must first accept the web service's connection and then consent to sharing the requested information (see Figure 1).

Figure 1
Schematic representation of the SSI online authentication process


① Show QR code on a web page
Scan QR code ②
Accept connection invitation ③
Login
Verifier
④ Ask for specific user information
Holder
Share asked information and proof from VCs ⑤
⑥ Validate VCs

1. The verifier displays a QR code on a web page.
2. The holder opens their wallet to scan the QR code.
3. The holder accepts the invitation to connect.
4. The verifier requests specific user information and credentials from the holder.
5. The holder provides the verifier with the requested information and proof in the form of verifiable credentials (VCs).
6. The verifier validates the user information and proceeds with the associated processes.

The protocol described above is suitable for online authentication, but is too cumbersome for the deployment of SSI in the analog world where we need more convenient solutions to encourage widespread use.

Shortcomings of physical IDs

Current physical identification processes are difficult to digitize and to automatize. A person shows their physical ID card to the verifier, who checks it but cannot be certain of its authenticity. Moreover, if the verifier then needs to manually copy the relevant information for digital verification processes, there are bound to be data quality issues.

Take, for example, a police agent who, having requested a driver's license and other physical documents, needs to verify their authenticity by typing the ID number of the driver into an online application in order to obtain the required information from a central service.

With a digital driver’s license as verifiable credential the same case in point could be handled with significantly less data exposure for the driver (who would need to reveal only the relevant information for the situation at hand). The police officer, on the other hand, would get tamper proof evidence of the driver’s identity and driving capacity and could, if necessary, automatically forward this information, e.g., to obtain the complete registration status of the vehicle and the current driver’s license rating.

③

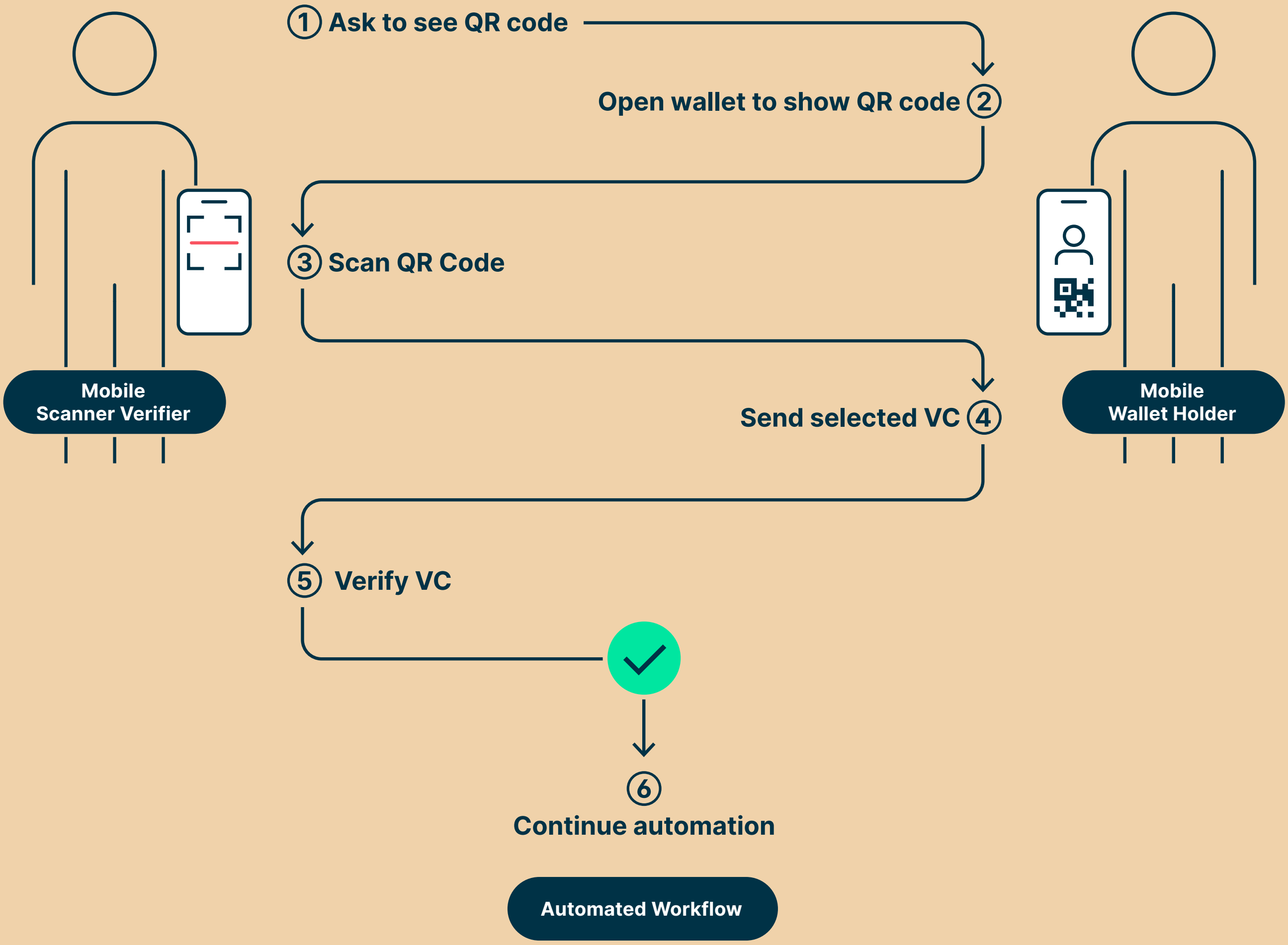
Target Solution

Definition of a generic use case

Cases like the police control described above can be aggregated into a generic use case that can serve as a basis for a wider adoption of SSI in everyday situations in the analog world. The goal is to provide a convenient way of proving user identity to other entities using QR codes or NFC readers:

A verifier, be it a machine or a physical person, should be able to ask for and receive proof of credentials via an easily integrable medium such as a QR code (see Figure 2).

Figure 2
Verifier asking a holder for proof of identity



1. The verifier asks to see QR code.
2. The holder opens the wallet and selects specific credentials to present proof.
3. The verifier scans the QR code with a mobile scanner.
4. Secure transaction of credentials.
5. The verifier validates the identity.

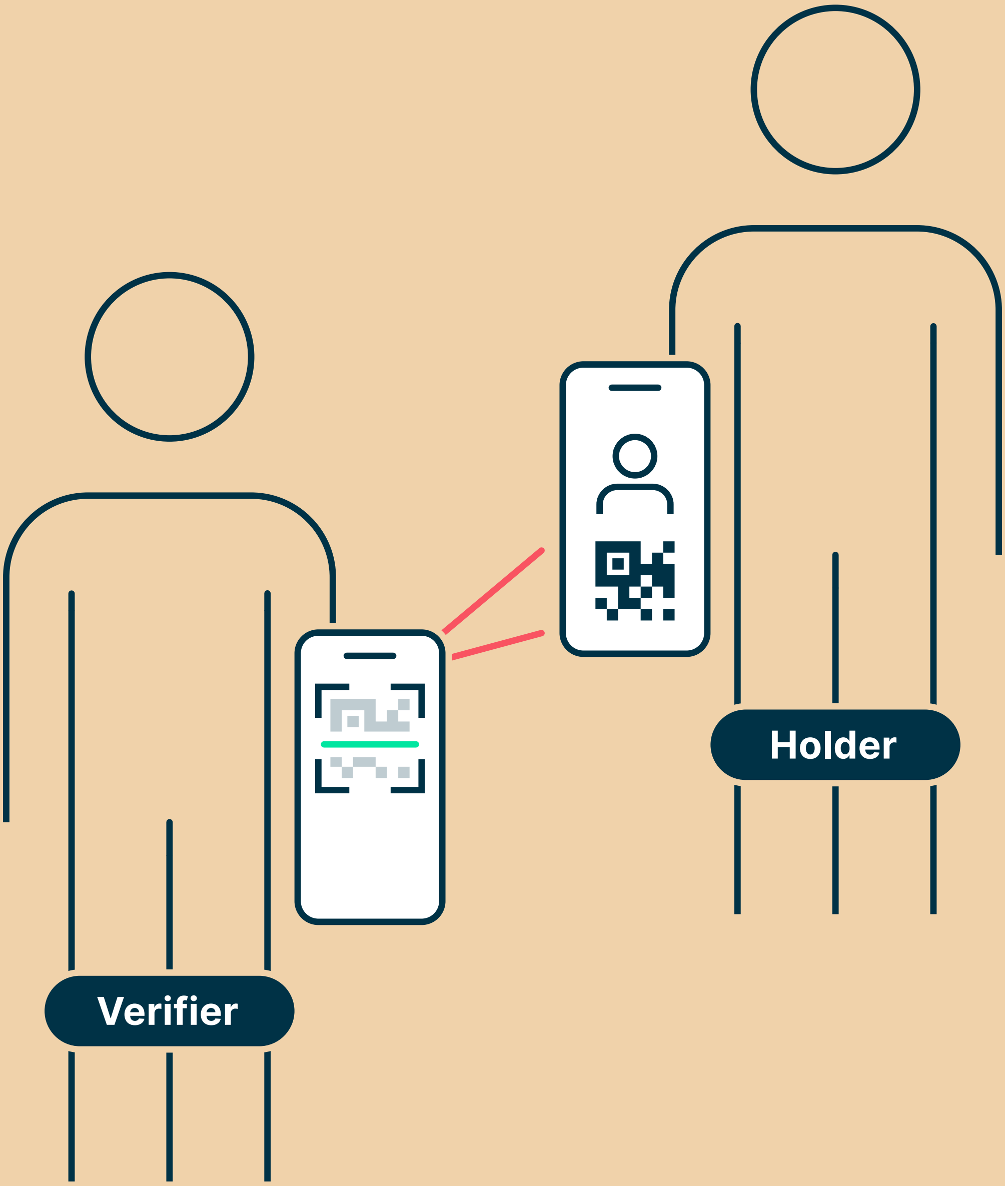
Technical deconstruction of our use case

Let us have a look at the technical prerequisites of the generic use case presented above. Each user is equipped with a client device allowing them to establish their identity. We assume the following about the client device:

Just like the physical wallet, a person usually has this device handy and can access its content whenever needed. To create this kind of accessibility and encourage the widespread adoption of digital identities in everyday life, we assume that customer devices are mobile phones with an integrated camera. Other devices such as laptops could also be used, but are less practical in terms of size and would potentially require an additional optical scanner for the verification process.

In contrast to online verification, where the verifier initiates the digital process, here the identity holder does so by presenting a VC in the form of a QR code or similar on their smartphone for the verifier to scan. This approach is chosen because it mimics the way an official document is normally presented to an authority and thus replicates the so-called «mental model» of the user [2]. The verifier checks the validity and integrity of the information so provided (see Figure 3).

Figure 3
Verifier asking a holder for proof of identity

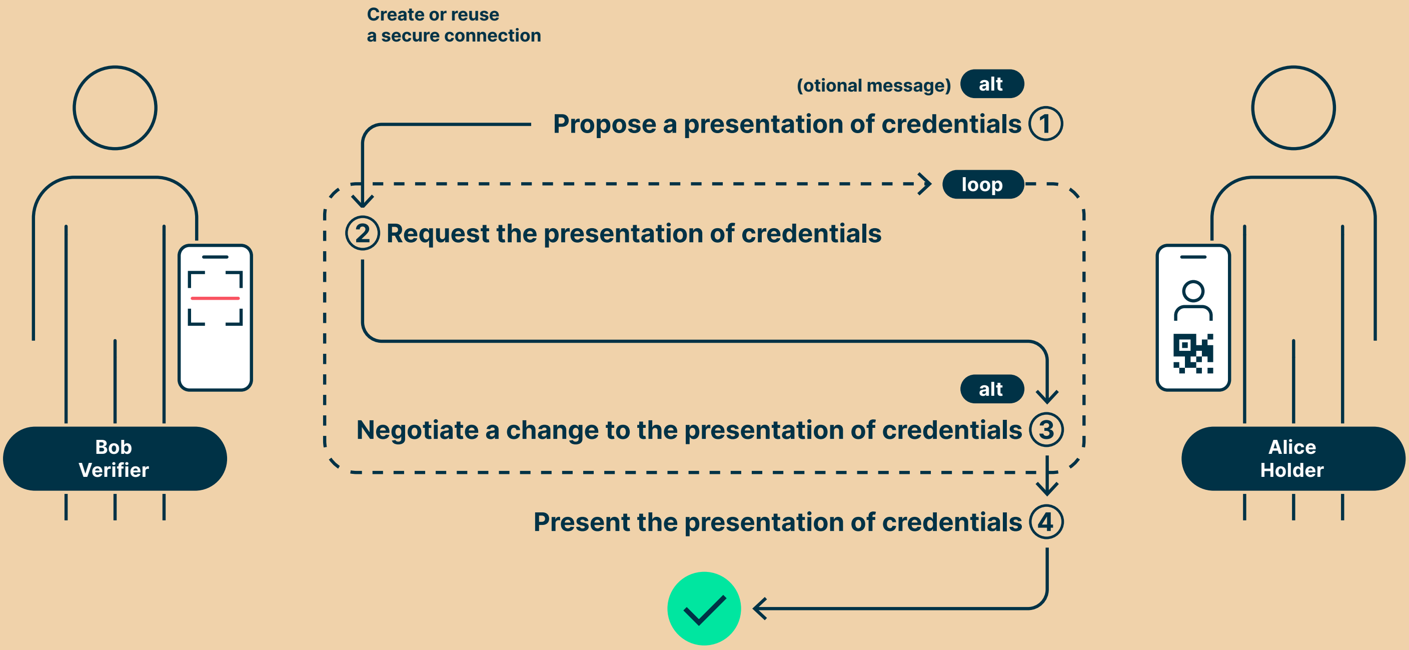


Setting up a private and secure communication between SSI users

In order to exchange personally identifiable information (PII), a secure and privacy compliant communication between the verifier and the holder must be set up. Such communication can be achieved by using the DIDComm standard [3] which is secure by default and provides a base layer for the protocols needed to exchange verifiable credentials (VC). The DIDComm standard defines how to establish a private and secure connection based on decentralized identifiers (DIDs) and DID Documents [4]. A popular protocol collection for SSI is the Aries project [5]. It defines and proposes implementations to issue and present VC proofs as well as several other protocols.

Verifier and holder need a DIDComm communication to use the Aries Protocol. Due to this constraint, the QR code displayed by the holder must be able to establish a DIDComm communication. In practice, the holder will generate an invitation to connect and present it to the verifier in the form of a QR code (Figure 4). The verifier scans the QR code and can then launch the DIDComm connection. Once the connection is established, the holder must notify the verifier that they wish to present a proof.

Figure 4
Proof presentation


Create or reuse
a secure connection
(otional message)
alt
Propose a presentation of credentials 1
loop
2 Request the presentation of credentials
alt
Negotiate a change to the presentation of credentials 3
Present the presentation of credentials 4
Bob
Verifier
Alice
Holder

The Aries RFC presentation protocol provides some flexibility by allowing the holder to send an optional message to the verifier to start the protocol.

To ensure interoperability with other mobile wallets, the holder's mobile wallet automatically starts the proof presentation process as soon as the DIDComm connection to the verifier is set up. The alternative, where the verifier initiates the proof presentation, is still possible, as each verifier application can simply ignore the holder's presentation request and send its own request.

As described above, it is thus theoretically possible to achieve the outlined use case by generating a connection invitation from the mobile wallet and sending an optional «present proof protocol» message to start the process. In the following chapters, a reference architecture for implementing the described protocol is presented.

Which wallet type to choose

An initial decision affecting the concept is whether the mobile wallet should be a local mobile wallet (or edge wallet), where all user data is stored on the user's smartphone, or a cloud wallet, where the application sends commands to a wallet in the cloud.

Edge wallets are mobile applications containing an SSI agent. They include most SSI functionalities, which means that users manage their own digital identity and are truly self-sovereign.

In peer-to-peer communication, however, there is a major problem as access to the IP network is restricted for client devices. To communicate with another client, a publicly accessible mediator must relay the incoming messages to the target client device (see Figure 5).

Another concern is the backup and recovery of digital identities. This should be convenient and easy for users. A practical approach for the secure encryption of data and the backup and recovery process can be found in the referenced thesis [6]. The concept is inspired by blockchain implementations for the secure generation of cryptographic keys. These keys are stored in the hardware security module of the smartphone and can only be accessed with a registered biometric credential. Wallet backups are encrypted and uploaded to the cloud and automatically retrieved when the user installs the mobile wallet on a new device.

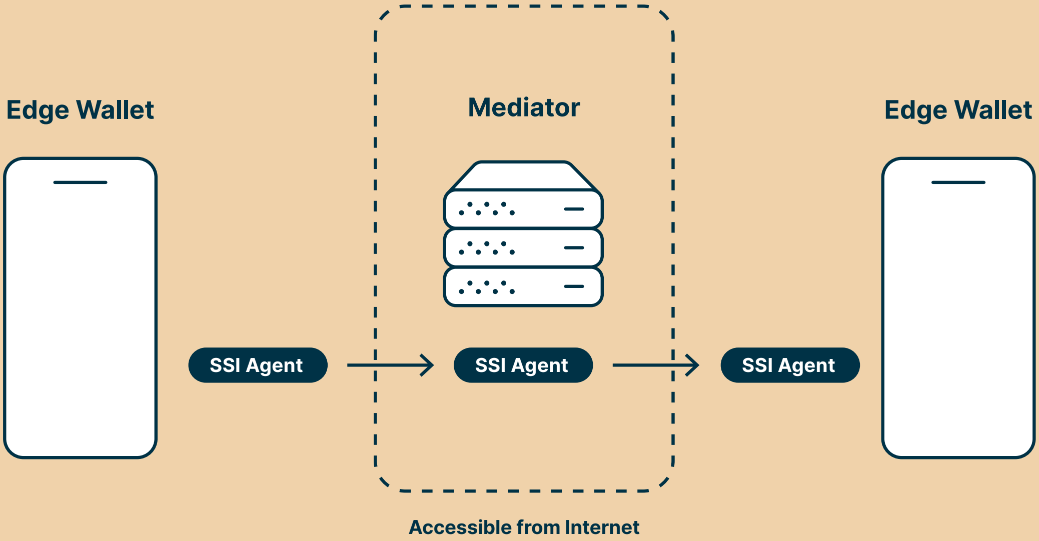
Cloud wallets, on the other hand, defeat the purpose of a user-centric wallet where the holder has full control as all user data is stored on a central platform. Edge wallets may need a mediator to communicate with each other, but cloud wallets require additional security measures between the mobile app and the wallet.

Considering the two options, we prefer the edge wallet for the following reasons:
1. greater flexibility due to the built-in implementation of the SSI protocol.
2. more privacy and data protection, as the user identity is stored on the device itself.
3. better security, since the hardware security module of the client device can be used. We consider the mediator problem to be minor, as the SSI edge agent can be easily set up and maintained by a trusted partner.

Furthermore, with edge wallets, offline scenarios via direct connections via Bluetooth or other future transport modes are technically feasible, although not yet standardized and implemented. This is another clear advantage compared to cloud wallets [6].

In summary, for the physical identity use case, edge wallets give users more control over their digital identity and allow for more useful features.

Figure 5
Edge wallets necessitate a mediator to communicate



Creation of a mobile (edge) wallet

Platform

To test the hypothesis of a physically enhanced SSI in an edge wallet, we decided to create a native Android prototype. The reason for this choice is the easier access to native components (e.g., Secure Enclave and HSM), better encapsulated datastore management and better support for background tasks than when using cross-mobile solutions such as React Native or Flutter.

Opting for native development comes with additional time costs, as two separate mobile wallet applications need to be developed. Nevertheless, native apps have the advantage of being more responsive, more secure and offering a better look and feel.

Aries

One of the goals is to use Aries AIP 2.0 [7] for this implementation. The only compatible Aries framework that can be embedded in a mobile application is Aries Framework Go.

Thanks to the possibility of compiling the Aries Framework Go to a C callable library (native library binary that can be called from higher level programming languages) with Java or objective-C wrappers [8], it is possible to embed it in Android and iOS applications, and with extensions also in cross-mobile solutions. For Android, the Aries Framework Go can be integrated as an Android archive (aar) like other Android libraries.

Components of an Android software architecture with an Aries framework

The optimal architectural choice is best described by Figure 6. The application uses an activity to manage global user interface (UI) elements, such as the top toolbar or the bottom navigation bar. The activity must also initialize critical components like the navigation components.

Navigating between views is managed by the navigation components [9]. At each view transition, the previous fragment is destroyed, and the next fragment is loaded.

Each fragment that contains data has a ViewModel companion. The ViewModel itself contains all the data in the view and acts as an intermediary between the fragment and the other components. The abstraction avoids business and data dependencies in the fragments making up the user interface. It also allows for extended interaction with services and coroutines [10].

In general, fragments and ViewModels do not fetch data but get it reactively via data streams.

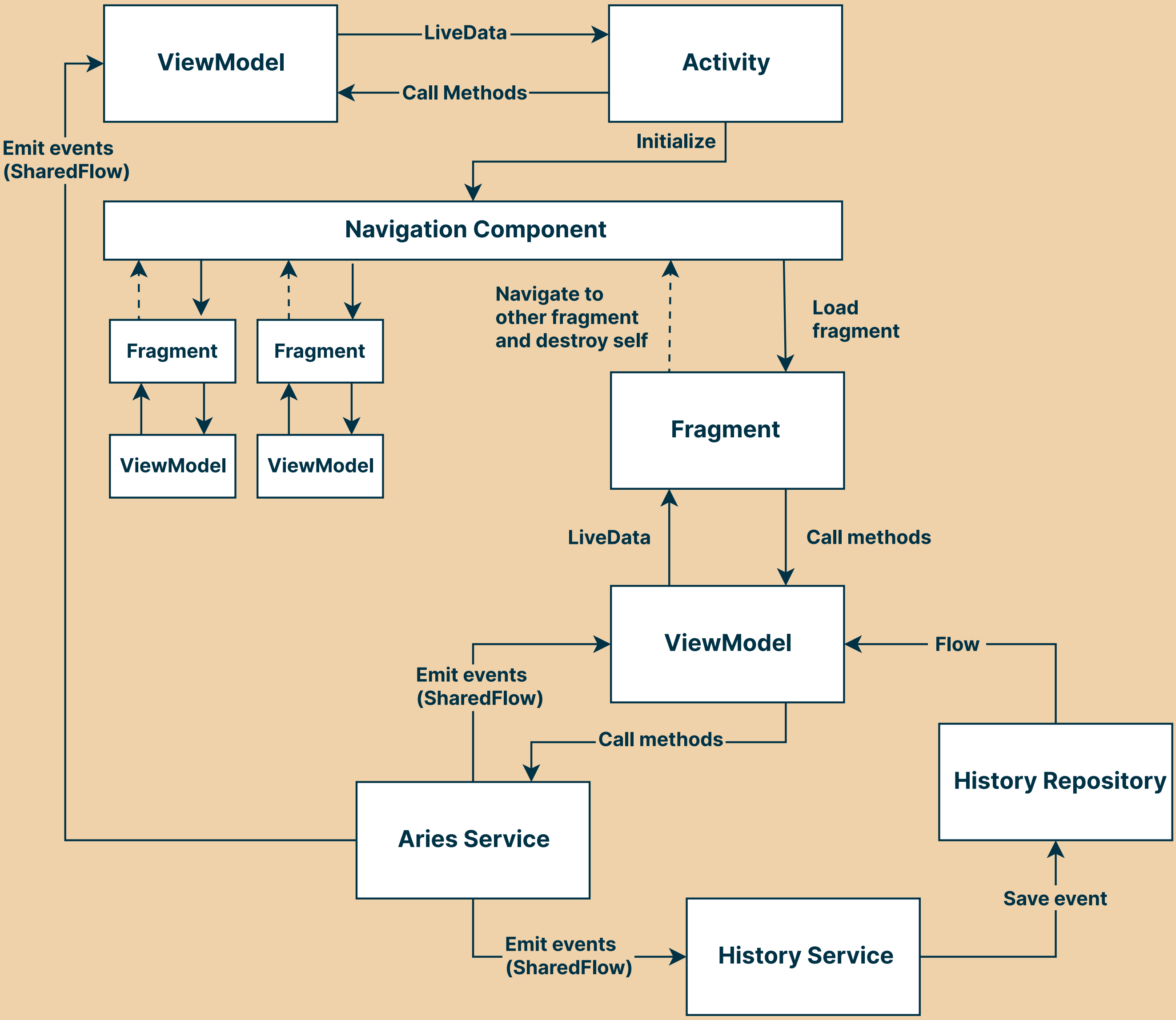
There are several possible patterns in Kotlin, three of which are used in this architecture:

– **LiveData** are usually located between fragments and ViewModels. They contain data and can be observed by the fragments for changes. They are also lifecycle aware, meaning that if a fragment is destroyed, the observer is automatically removed to avoid leaks [11].
– **Flows** are data streams that typically occur between services, repositories, and ViewModels. They support asynchronous data transformations and are only executed when collected. They support basic operations that can be useful, such as flow concatenation, where multiple data sources with the same object type can be merged. The resulting data stream is updated each time one of the data sources sends new values [12].
– **SharedFlows** are similar to flows with one major difference [13]: SharedFlows are executed even if they are not collected by anyone. In practice, this means that a component starting to listen to a SharedFlow will not receive old data transmitted before it was listening.

The Aries Service (which interacts with Aries Framework Go) uses the SharedFlow internally as an Event Bus [14] to distribute Aries events to all components of the application. This architecture choice allows asynchronous operations globally over the activity, limited to the current view with the ViewModel, or completely in the background like the History Service, which stores the events occurred without user interaction.

The application also includes global dependency injection with Hilt. Hilt is recommended by the Android documentation to increase the testability and scalability of Android applications [15].

Figure 6
Overview of the Android architecture



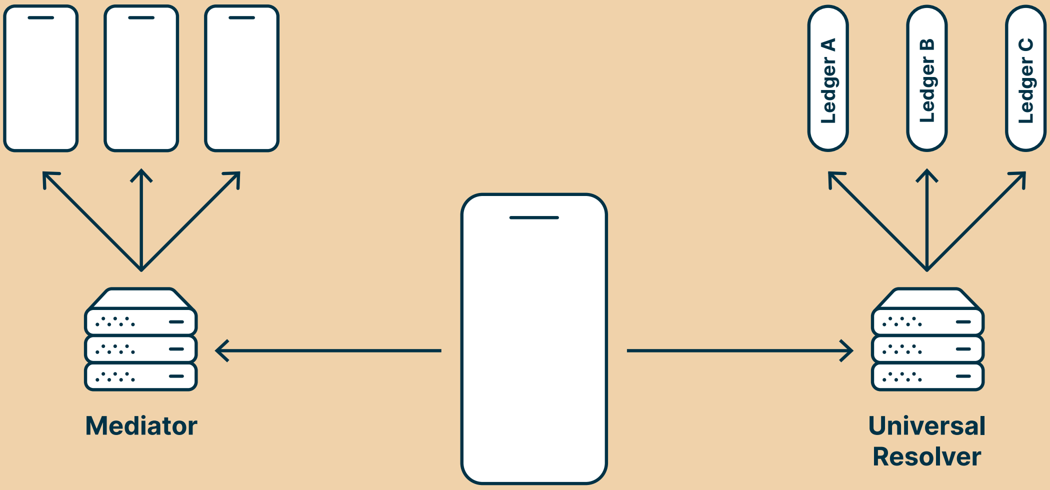
The proposed reference architecture allows for a loosely coupled integration with an Aries framework and uses asynchronous execution to improve the responsiveness, maintainability, and extensibility of the application.

Additional infrastructure needed for the mobile application
As mentioned earlier, mobile edge wallets cannot act completely independently due to the nature of their host devices. To receive messages, they must coordinate with a publicly available mediator.

In addition to the mediator, there is another service, the DID Resolver. It is not essential for the moment, but will play a bigger role in terms of interoperability in the future, as it will not be possible to integrate every ledger technology into the mobile wallet.

The Decentralized Identity Foundation (DIF) provides an open-source solution that can implement a variety of DID methods on a configuration-by-configuration basis. Aries Framework Go supports requests to the DID Resolver and by extension allows the Android prototype wallet to support multiple ledger technologies.

Figure 7
Infrastructure dependencies for mobile edge wallets



Privacy issues and future governance issues

Self-sovereign identity is apt to solve a number of privacy issues and provide users with a convenient and secure way to store and prove their digital identities – online and in the analog world. However, the associated systems are not (yet) perfect and may raise privacy concerns.

Mediators and resolvers
In the case of an edge wallet, these concerns mainly relate to the use of mediators and resolvers. Both services receive requests from participating wallets and could infer information about the user from the data they collect.

A fixed mediator, for example, knows all the user's DIDComm connections and can identify who they are talking to if the DIDs of the involved parties are public, which is the case with issuers. Even if the mediator only gets to see encrypted data, it may still be able to correlate the frequency and size of the encrypted messages to guess which protocols are being used and how often the communication is taking place.

The same holds true for DID resolvers. Their use is convenient because the wallet does not need to implement a separate resolver for each DID method. The disadvantage, however, is that these services know the holder's IP address and can correlate it to the public DID they are supposed to resolve.

The trust anchor problem
Standards and implementations in the SSI context are evolving rapidly and may soon produce practical solutions for this kind of problem. There remains, however, another major issue to be resolved: Layer four of the SSI concept is governance, which enables trust between systems and users and forms the backbone of interoperability. Trust can be represented, for example, by authorities endorsing trustworthy sources. Such entities are called certificate authorities (CA) or trust anchors. Trust anchors are essential to ensure that the identity of a person or thing is legitimate. Most people are familiar with trust anchors through the root CA certificates of operating systems and browsers, which ensure that a website is secure and authentic [16].

In the SSI ecosystem, the most important identities are the public DIDs of issuers. An issuer provides information about an entity in the form of VCs. Any verifier must request the resolution of the issuer's DID to obtain the public key required to verify that the VC's proof is valid. But while the verifier can verify that the VC is technically valid, he has no way of knowing whether the information provided by the issuer in the VC is correct.

To solve this trust problem, a verifier needs to be sure of the issuer's own trustworthiness. This can be achieved through whitelists created by an authority certifying the identity of companies, such as the Global Legal Entity Identifier Foundation (GLEIF) [17], through a VC issued by a CA certifying the issuer's credibility, or a linked domain challenge where the issuer must have control over a domain name, etc.

There is also a potential trust issue between the VC holder and the verifier. Holders could be tricked by fake verifiers into revealing personal information (phishing). The solution to this problem will have to be similar to the solutions presented above for issuers, taking into account that other holders can also verify VCs.

Summarizing, we can state the following: While solutions as the one we present in this paper are undoubtedly a big step forward in the digitization of physical identification, an optimal solution for the governance of trust has yet to be found.

④

Conclusion

In this paper, we explore a way to digitally enhance the physical verification process. This is done by developing a novel SSI approach to physical world interactions optimized for mobile wallets with the help of an experimental next generation version of AIP. A native mobile wallet prototype application for Android, built according to our proposed software architecture and using Aries AIP 2.0, serves as a proof of concept and showcase for this approach.

We thus provide a blueprint for a smoother SSI experience in the analog world. While some potential privacy downsides clearly remain to be addressed, SSI offers compelling advantages over the current state of affairs in the analog world which is still relying on physical forms of identification, such as driver's licenses and passports. By offering a reference design compatible with easy-to-use, real-world SSI, we hope to help precipitate a future where everyone can verify their identity and selectively share their information with optimal privacy, security and autonomy.

Author

Michel Sahli
Security Consultant
michel.sahli@adnovum.ch

5

Glossary

AIP	Aries Interop Profiles are community agreed standards.
Aries RFCs	Aries RFCs – Requests For Comments – are topics, features and concepts for standardizing the Aries ecosystem.
CA	Certificate Authority
DID	Decentralized Identifier: a type of identifier enabling a verifiable, decentralized digital identity.
DID Resolver	Service that resolves multiple DID methods like SOV, ION, etc.
DIDComm	Secure and private communication channel between two devices built on the DID standard.
DIF	The Decentralized Identity Foundation is an engineering-driven organization focused on developing the foundational elements necessary to establish an open ecosystem for decentralized identity and ensure interoperability between all participants.
GLEIF	The Global Legal Entity Identifier Foundation is an international organization listing every company in the world by means of a unique identifier.
(Hyperledger) Aries	Hyperledger Aries provides a shared, reusable, interoperable tool kit designed for initiatives and solutions focused on creating, transmitting and storing verifiable digital credentials.
NFC reader	Operating in a frequency range centered on 13.56 MHz, Near Field Communication (NFC) readers enable contactless, short-range communication between compatible devices at close proximity.
QR code	A quick response code is a type of two-dimensional (2D) bar code used to provide easy access to online information through the digital camera on a smartphone or tablet.
VC	A VC or Verifiable Credential is an open standard for digital credentials.

⑥

References

1. **Exploring the potential of Self-Sovereign Identity with representative use case, Adnovum Blog, January 12, 2022:** https://www.adnovum.com/blog/exploring-the-potential-of-self-sovereign-identity-with-representative-use-cases
2. **Mental Models, Nielsen Norman Group:** https://www.nngroup.com/articles/mental-models/
3. **DIDComm Messaging:** https://identity.foundation/didcomm-messaging/spec/
4. **Decentralized Identifiers (DIDs) v1.0, W3:** https://www.w3.org/TR/did-core/
5. **Hyperledger Aries Specification:** https://github.com/hyperledger/aries
6. **Application of Self-Sovereign Identity in the Physical World, Master Thesis, Michel Sahli:** https://heig-vd.ch/docs/default-source/docs-groupe-ysecurity/sahli_mse_tm_sa21.pdf
7. **Aries Interop Profile Version: 2.0:** https://github.com/hyperledger/aries-rfcs/tree/main/concepts/0302-aries-interop-profile#aries-interop-profile-version-20
8. **Aries Agent Mobile: h**ttps://github.com/hyperledger/aries-framework-go/blob/main/cmd/aries-agent-mobile/README.md
9. **Navigation, Android Developers Guide:** https://developer.android.com/guide/navigation
10. **Kotlin coroutines on Android, Android Developers Guide:** https://developer.android.com/kotlin/coroutines
11. **LiveData, Android Developers Guide:** https://developer.android.com/reference/kotlin/androidx/lifecycle/LiveData
12. **Kotlin Flows, Android Developers Guide:** https://developer.android.com/kotlin/flow
13. **StateFlow and SharedFlow, Android Developers Guide:** https://developer.android.com/kotlin/flow/stateflow-and-sharedflow
14. **How To Implement The Event Bus Pattern With Kotlin SharedFlow In Your Android App. Medium:** https://medium.com/tech-takeaways/how-to-implement-the-event-bus-pattern-with-kotlin-sharedflow-in-your-android-app-768529828607
15. **Hilt and Dagger, Android Developers Guide:** https://developer.android.com/training/dependency-injection/hilt-android#hilt-and-dagger
16. **What Is a Certificate Authority (CA):** https://www.ssl.com/faqs/what-is-a-chain-of-trust/
17. **GLEIF – Global Legal Entity Identifier Foundation:** https://www.gleif.org/en/

And your digital business works

adnovum.com

adnovum

At our headquarters in Zurich and our other offices in Europe and Asia we employ about 600 IT specialists.

Adnovum Informatik AG
Badenerstrasse 170
8004 Zurich

Phone +41 44 272 61 11
info@adnovum.ch