Smoothing the Ride: Providing a Seamless Upgrade Path from Established Cross-Border eID Workflows Towards eID Wallet Systems

Roland Czerny[1,2][a], Christian Kollmann[3],
Blaž Podgorelec[1,2][b], Bernd Prünster[3][c] and Thomas Zefferer[3]
[1]*Institute of Applied Information Processing and Communications (IAIK), Graz University of Technology, Austria*
[2]*Secure Information Technology Center Austria (A-SIT), Austria*
[3]*A-SIT Plus GmbH, Austria*

Keywords: eIDAS, Wallet, eID, eGovernment, European Digital Identity Wallet, Attestation.

Abstract: The eIDAS regulation and its technical implementation successfully enabled cross-border eID use cases within the European Union. Established in 2014 as part of an EU regulation, its technological foundation is starting to show its age, particularly on smartphones. The *European Commission* (EC) is well aware of this fact, and large-scale pilots for the wallet-based, next-generation pan-European eID framework are on their way. This work fills the gap between both approaches and enables member states to provide wallet-based authentication to established service providers right now. Our prototypical implementation effectively demonstrates that cross-border, wallet-based eID workflows can be rolled out already, while catering towards the constraints of already operational infrastructure. We achieve this by introducing an eIDAS app, which supports both existing eIDAS-based cross-border authentication as well as interaction with wallet apps.

1 INTRODUCTION

Member states of the *European Union* (EU) operate national identity management systems and issue their citizens national *electronic identification* (eID) means, which allow for secure and reliable authentication at national public-sector online services. In a converging European society, it is crucial that national eID means cannot only be used in the issuing member state but also allow for secure authentication at online services provided in other EU member states. Using national eID means issued by *member state A* to securely log in to an online service provided in *member state B* has become known under the term *cross-border authentication.*

In the EU, *Regulation (EU) No 910/2014* (the so-called *eIDAS Regulation*) is the legal basis for secure and legally binding electronic cross-border authentication. On a technical level, cross-border authentication is achieved by federating existing national identity management systems. For this purpose, each member state operates a so-called *eIDAS Node* that is connected to the respective national identity management system. eIDAS Nodes from different member states form a circle of trust and can delegate user authentication to each other.

Cross-border authentication, as defined by the eIDAS Regulation, has been in productive operation in the EU for many years. The underlying technical framework has hence proven to be well suited for the intended use cases. However, these use cases still mainly cover web-based scenarios where users access online services from other member states using web browsers on desktop-type devices. When the eIDAS Regulation was enacted, such use cases had been predominating, and the choice of technologies for the Regulation's technical implementation seemed reasonable. In the meantime, mobile end-user devices such as smartphones have emancipated from desktop computers and laptops, and the use of dedicated mobile apps to consume online services has become the far more common usage scenario. Unfortunately, the current technical implementation of European cross-border authentication is not fully ready for these scenarios.

[a] https://orcid.org/0009-0000-9230-0833
[b] https://orcid.org/0000-0002-6322-746X
[c] https://orcid.org/0000-0001-7902-0087

Czerny, R., Kollmann, C., Podgorelec, B., PrÃijnster, B. and Zefferer, T.
Smoothing the Ride: Providing a Seamless Upgrade Path from Established Cross-Border eID Workflows Towards eID Wallet Systems.
DOI: 10.5220/0012091900003555
In *Proceedings of the 20th International Conference on Security and Cryptography (SECRYPT 2023)*, pages 460-468
ISBN: 978-989-758-666-8; ISSN: 2184-7711

To prepare cross-border authentication in Europe for future challenges, the EC has recently published a proposal[1] for a successor of the current eIDAS Regulation. The core concept of the EC's proposal is the so-called *European Digital Identity Wallet* (EUDIW). The EUDIW is supposed to be a technical component under full control of the user and enables secure storage and presentation of asserted identity information. The fundamental idea is that this information is no longer provided by central national identity management systems but in a decentralized way by the user and the user's wallet. The EUDIW not only improves privacy by putting the users in full control of their data, but it will also enable mobile usage scenarios involving mobile end-user devices.

The concept of identity wallets has existed already before it has been picked up by the EC. However, its integration into the proposal for a new eIDAS Regulation has boosted the concept's popularity. This is also reflected by work published published by Gaehtgens (2022). Although currently hyped, it is expected, e.g. by Buchanan et al. (2022); Schwalm et al. (2022); Sharif et al. (2022), that with the introduction of the EUDIW the concept of identity wallets will remain relevant and play an important role in the future. However, at this stage, various technical aspects are still open, and no fully-fledged implementation of the EUDIW is available yet. This makes it difficult for online service to experiment with and use this new authentication method.

To tackle this issue, we propose a solution for online services to integrate, experiment with, and use identity wallets for authentication. While the EUDIW is still under development, we provide an implementation, which can be used to gain first hands-on experiences with this technology. The proposed solution is aimed at smoothing technical integration into existing online services. Thus, it does not require any fundamental changes for existing online services but introduces an app-based middleware component to emulate a standard eIDAS-based authentication process. Required trust in this additional component is achieved by augmenting the current eIDAS-based trust assumption using remote-attestation techniques. This distinguishes our proposal from other wallet-related trust models and solutions like *European Self-Sovereign Identity Framework* (eSSIF)) that rely on blockchains (Kubach and Roßnagel, 2021).

This paper introduces our identity-wallet solution and demonstrates a proof-of-concept, which has been designed, implemented, and operated in the scope of the H2020 project *mGov4EU*. The remainder of this paper elaborates on our proposed solution: Section 2 briefly surveys related scientific work and puts our proposal into the context of existing wallet solutions. In Section 3, relevant background information on selected aspects of the proposed solution is presented. The proposed solution is then introduced in detail in Section 4. Section 5 reports on the evaluation of the proposed solution, before the paper is finally concluded in Section 6.

2 RELATED WORK

Although identity wallets have gained attraction only recently, they have already been touched by several scientific publications. While, to the best of our knowledge, no solution has been proposed yet that supports cross-border wallet-based authentication in the scope of eIDAS, at least the publications discussed below can be considered relevant also for our own work.

(Abraham et al., 2021) presented a concept to achieve strong authentication (reaching *Level of Assurance* (LoA) high as defined by the eIDAS Regulation) using a mobile-based identity wallet. The author's proposed concept utilizes *verifiable credential* (VC) and *decentralized identifier* (DID); the concept has been validated using a proof-of-concept implementation. In contrast to our own work, the solution proposed by (Abraham et al., 2021) requires online services to integrate support for dedicated protocols (e.g., for resolving DID). (Jacobino and Pouwelse, 2022) presented a similar concept using eSSIF and *European Blockchain Services Infrastructure* (EBSI). In contrast to our solution, this wallet concept has the same requirements as the solution cited before: to enable wallet-based authentication, major modifications of the online service are required.

Relevant related work has also been published by (Ali et al., 2010), who have combined trusted computing concepts using remote attestations with the *Federated Identity Management System* (FIDMS) model. The authors have introduced an extended FIDMS, where the *identity provider* (IdP) attests to identity attributes and vouches for its platform integrity.

To summarize, the design and implementation of identity wallets have already been discussed in the literature, and some implementations of identity wallets have already been introduced. However, none of the proposed solutions has so far focused on easing integration with existing online services.

[1] https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52021PC0281&from=EN

3 BACKGROUND

Our solution relies on a variety of concepts and technologies and is intended to be embedded into a well-defined environment. This section therefore provides all required background information before discussing our contribution in Section 4. First, a generic model for digital identity wallets is introduced. Next, relevant technical details on implementing secure and reliable communication between mobile apps on smartphones are discussed. A brief survey on remote-attestation features of current mobile platforms then concludes this section, as these features underpin the security backbone of our solution.

3.1 A Model for Digital Identity Wallets

The basic concept of a mobile wallet is well established. It has, for instance, emerged in the payment sector. In this context, app-based wallets on smartphones are used to store credit-card data and to authorize payments at points of sale. Such solutions are already widely used.

Wallet solutions such as the EUDIW rely on similar concepts and technologies but pursue different objectives and store identity information of their user (wallet holder). Instead of authorizing payments, users use their wallet and the data stored therein to securely authenticate at online services. From this basic functionality, a generic model for digital identity wallets inspired by *World Wide Web Consortium* (W3C) VC data model, described in (Sporny et al., 2021) can be defined. This model is shown in Figure 1.

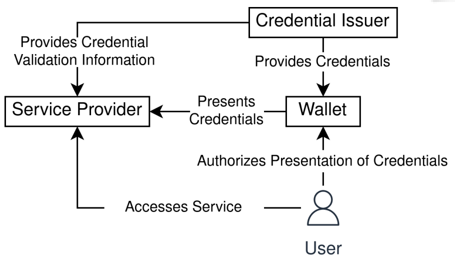

Figure 1: Generic Model for Digital Identity Wallets.

This model comprises the user (wallet holder) and three entities: the wallet, the credential issuer, and the service provider. The service provider is the entity providing an online service to the user. To access and use the provided service, the user needs to authenticate first. Authentication is carried out by means of the wallet, which transfers attested identity information (credentials) to the service provider. This needs to be authorized by the user, who therefore interacts with the wallet.

Before identity information can be transferred from the wallet to the service provider, this information must be stored in the wallet first. This is done by the credential issuer. The credential issuer retrieves the user's identity information from the respective sources (e.g., national registers), attests the information (typically by signing it), and stores the attested information in the wallet.

Note that the credential-issuing process, through which attested identity information is stored in the wallet, is typically carried out only once. Once available in the wallet, attested identity information (credentials) can be used repeatedly to authenticate at the service provider.

3.2 App-to-App Communication

A key feature of the wallet solution proposed in this paper is its fully mobile nature. Transitioning from browser-based workflows to mobile-app-centric ones demands thorough consideration of each involved app's responsibilities, since individual apps being responsible for strictly-scoped tasks enforces a clear separation of concerns. At the same time, it should be possible to fall back to a mechanism that works even in the absence of a specific app. Luckily, both iOS and Android support *claiming* of URLs by apps in a way that works similarly.

As the name suggests, app-to-app communication through claimed URLs works by passing information between apps encoded into URLs. For an app to claim an URL of a web service, a so-called *asset links* file needs to be hosted on the web server, which effectively authenticates the app that wants to claim this URL to the operating system on installation time. In practice, this works as follows: Any mobile app (including the browser) can issue an HTTP GET request to a claimed URL and the target app (the one which claimed the URL) opens up automatically. In case no app has claimed the URL, the web browser handles the request, thus providing a fallback mechanism.

In conclusion, a unified mechanism for app-to-app communication is available, which automatically falls back to a browser-based workflow if a particular app is not installed. If a procedure cannot be carried out in the browser, it is at least possible to guide the user towards installing an appropriate app. This can be necessary for security-critical procedures requiring remotely establishing trust in client software, which is not possible for mobile browsers.

The following section picks up on the topic of establishing trust in mobile applications through attestation. It covers Android and iOS and discusses the commonalities and differences of both platforms.

3.3 Remote Attestation

Current versions of iOS and Android natively support remote attestation, i.e., remotely establishing trust in a device, its operating system and specific apps. However, the capabilities and technical details of the approaches followed by each mobile platform differ significantly. Android natively supports attesting hardware-backed cryptographic key material. iOS, on the other hand, does not directly support attesting cryptographic material, but rather aims to establis trust in apps.

Platform specifics aside, a general requirement for any kind of meaningful and reliable remote attestation concept is the presence of cryptographic hardware modules and their tight integration with the operating system. Our use case heavily relies on assurance with respect to key material, which is stored in hardware, such that it can neither be extracted nor illegitimately used. As such, establishing trust in keys is a baseline requirement, even though remotely establishing trust in unmodified apps is also critical. Hence, we require a combination of the features natively supported by iOS and Android.

We have released a library as free and open-source software, which hides the technical intricacies, providing a uniform interface for remotely establishing this required trust to make all features available on both major mobile platforms[2]. More details on platform specifics are provided in the following two subsections.

3.3.1 Android Key Attestation

Android supports key attestation[3], i.e. trust can be remotely established in cryptographic material managed by secure hardware. More concretely, a public/private key pair is generated in hardware and an X.509 certificate containing the public key can be exported. This certificate contains a set of parameters, indicating whether the bootloader or the operating system have been tampered with. In addition, this so-called *attestation certificate* also contains information about the operating system version, the patch level and information about the application, which was used to generate the key pair. The app-related information comprises the package name and information on which *signing certificate* was used during the application's publishing process. Evaluating this compound information allows for remotely establishing whether an unmodified application was used to generate hardware-backed key material on an uncompromised device, running an untampered operating system. A more in-depth explanation can be found in the paper on this matter by Prunster et al. (2019)

Apple devices, in contrast, do not natively support key attestation and rely on an Apple-operated service to establish trust in apps. As a consequence, establishing the trust in cryptographic material is not directly possible.

3.3.2 Apple App Attestation

Apple directly approaches remotely establishing trust in smartphone apps. The *DeviceCheck*[4]/*AppAttest*[5] framework has apps contact Apple servers using *WebAuthn*[6] flows to establish whether a device has been jailbroken (i.e. compromised) and/or an app has been tampered with. Such an *attestation* is supposed to happen once.

The attestation procedure also involves the creation of a hardware-backed public/private key pair and the creation of a signature using this key pair. In contrast to Android, however, this key cannot be used for anything else, hence preventing *key attestation*. During app usage, however, the correct execution of certain critical procedures can be *asserted*. A so-called *assertion* is then created, which includes (amongst others) the return value of an operation. This assertion is signed by the very same key created for the initial *attestation*. Section 4.2.3 elaborates on how we used these properties to enable key attestation on iOS as part of our trust concept.

4 A SMOOTH UPGRADE PATH TOWARDS WALLET-BASED AUTHENTICATION

This section introduces the proposed identity wallet solution in detail, especially focusing on two aspects. The first aspect concerns the fact that our concept enables legacy *OpenID Connect* (OIDC) *relying partys* (RPs) (also called *service providers* (SPs)) to utilize next-generation wallet-based authentication in accordance with Figure 2. As a prerequisite to making our vision a reality, eIDAS-based cross-border authentication first needs to be made more user-friendly on mobile devices. Hence, this section also puts a special focus on this aspect.

[2]https://github.com/a-sit-plus/attestation-service

[3]https://developer.android.com/training/articles/security-key-attestation

[4]https://developer.apple.com/documentation/devicecheck

[5]https://developer.apple.com/documentation/devicecheck/validating_apps_that_connect_to_your_server

[6]https://webauthn.io/

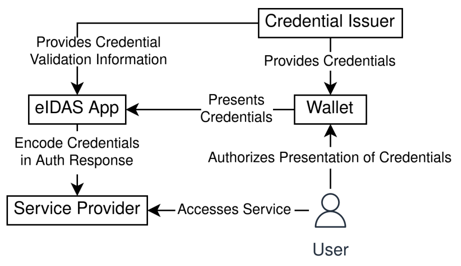
Credential Issuer
Provides Credential Validation Information
Provides Credentials
eIDAS App
Presents Credentials
Wallet
Encode Credentials in Auth Response
Authorizes Presentation of Credentials
Service Provider
Accesses Service
User

Figure 2: High-level view of components interacting to provide legacy SPs with wallet authentication

4.1 High-Level Concept

First and foremost, we introduce a dedicated *eIDAS app* to interface with service providers and eIDAS nodes. By itself, this already provides added value, since using dedicated apps for different tasks feels more native on mobile platforms. Far more importantly, however, this respects the security model of modern mobile operating systems, isolating apps operating on different data.

The eIDAS app supports two authentication modes: In *eIDAS mode*, the eIDAS app is responsible for orchestrating the communication between eIDAS nodes, connectors, and proxy services to enable traditional cross-border authentication workflows in accordance with Figure 3. However, if the user chooses to authenticate not via their home country's identity management system through eIDAS, but through their local mobile wallet, the eIDAS app does not forward an authentication request to an eIDAS node. In this *legacy wallet mode of operation*, it acts as an OIDC *Self-Issued OpenID Provider v2* (SIOPv2) service provider towards a mobile wallet app instead. The attributes received from the wallet app are then transformed into a regular OIDC authentication response and passed back to the service provider.

A precondition for this to work is obviously having such an app installed and ready to serve VCs through *verifiable presentation* (VP). Hence, we have implemented a wallet app, which aligns with the EU-DIW initiative, a corresponding provisioning service, and connected it to a production eID system for evaluation purposes (see Section 5). The issue of establishing enough trust in the eIDAS app to permit an on-device transformation of verifiable credentials to a traditional OIDC authentication response is absolutely crucial in this context and an integral prerequisite to making our concept work in practice.

4.2 Building a Foundation of Trust in Mobile Apps

Our solution to augment cross-border authentication with local wallet-based authentication in a mobile-first manner rests on a foundation of trust between a variety of actors and components. This section delves into each trust aspect individually, before providing functional details of our solution in Section 5.

4.2.1 Trust Relationships Between SP and Connector

From an SP's point of view, the eIDAS node connector (see also Figure 3) acts as an IdP. In accordance to the eIDAS trust model, SP and connector trust each other. When relying on OIDC as the authentication protocol, this is reflected by two facts: First and foremost, the SP is registered at the connector as an OIDC relying party. Secondly, the SP trusts the certificate the connector uses to sign authentication responses.

In the mobile setting we target, an SP app is assumed, while the proposed eIDAS app needs to claim the URL of the connector. From an SP app's point of view, it is not even possible to detect that an eIDAS app was involved at all, since the connector operates on authentication information received from the eIDAS node and transforms it into an OIDC-compliant authentication response. Hence, the service provider does not need to establish trust in the eIDAS app.

Our concept radically changes this trust relationship, as soon as the eIDAS app acts as a compatibility layer to augment legacy service providers with wallet-based authentication: The eIDAS app is tasked with transformation operations – from regular OIDC authentication requests to OIDC SIOPv2, and from attributes received through VP to regular OIDC-compliant authentication responses.

4.2.2 Establishing Trust in the eIDAS App

We have solved the issue of an SP trusting the eIDAS app by extending the functionality of the SP country's eIDAS node connector. From a user's point of view, the eIDAS app needs to be enrolled at an eIDAS node

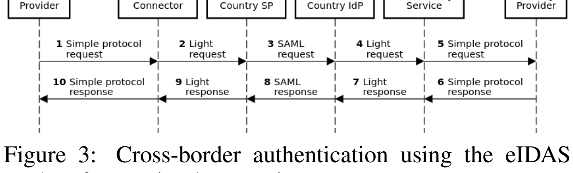

Figure 3: Cross-border authentication using the eIDAS Node reference implementation.

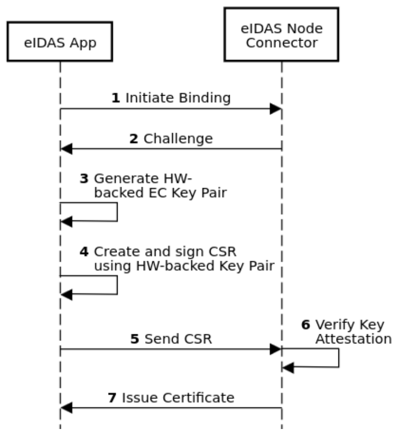

Figure 4: Creating a binding for the eIDAS app.

connector during a one-time setup process. Under the hood, the eIDAS node connector relies on key attestation to establish trust in an eIDAS app instance installed on an unmodified device. This trust relationship is expressed by the connector issuing a certificate for an attested instance of the eIDAS app, signed by the same key used to sign authentication responses during regular authentication procedures.

The issued certificate references a hardware-backed key, which, in turn, is used by the eIDAS app to sign those authentication responses. Since the SP trusts the signing certificate of the connector, and this signing certificate was used to create a certificate chain down to the eIDAS app's certificate, the SP also considers authentication responses signed by the eIDAS app authentic and trustworthy. The sequence diagram depicted in Figure 4 illustrates the technical details of the binding process between connector and eIDAS app:

1. The app initiates the binding process
2. The connector generates and responds with a challenge (a cryptographic nonce)
3. The eIDAS app creates a hardware-backed key pair
4. This key pair is encoded into a *certificate signing request* (CSR)
5. The CSR is sent back to the eIDAS node connector (along attestation information)
6. The eIDAS node connector then verifies key origin, app and device integrity
7. If all checks succeed, the eIDAS node connector issues a binding certificate for the eIDAS app

While key attestation, as provided on the Android platform, supports this out of the box, iOS does not directly support it. To address this issue, we have managed to emulate this Android-specific concept on iOS, as explained in the following section.

4.2.3 Enabling Key Attestation on iOS

In addition to the required cryptographic material used to create an app attestation (see Section 3.3.2), we mandate the creation of a second public/private key pair, which can be freely used by the application. The creation of this key pair is *asserted*. Hence, we obtain an assertion, which contains the public part of this second key pair. We also mandate the counter contained in this assertion to be equal to `1`, meaning that the very first asserted procedure carried out in the application is precisely the creation of this second key pair. This procedure, in effect, provides key attestation on iOS.

We have created prototypical implementations of all components relevant to our concept, to evaluate its feasibility, and integrated them with a production eID system. More in-depth technical details of our overall concept can be found in the following section.

5 EVALUATION THROUGH IMPLEMENTATION

Implementing all building blocks relevant to our concept, and subsequently connecting them to a production eID system provided a setup mirroring a production environment. This deployment was then used to demonstrate the feasibility of our approach. The implementation itself is designed to be modular and hence consists of several parts based on the eIDAS Node reference implementation from the EC.

5.1 eIDAS Node

As the setup for demonstrating our concepts is based on the eIDAS node integration package provided by the EC[7], it consists of several components for the Java platform.

Our modifications mainly target the connector, since this component interacts with service providers and therefore serves as the junction point between eIDAS-based cross-border authentication and mobile-wallet authentication. Hence, this section therefore focuses on the modifications made to the connector.

Since the node connector acts as an IdP towards the SP, it is the only component an SP is directly in-

[7]https://ec.europa.eu/digital-building-blocks/wikis/display/DIGITAL/How+to+implement+or+operate+an+eIDAS-Node

terfacing with. As mentioned before, our implementation relies on OIDC, as it is well established on mobile platforms.

5.1.1 OIDC

In Section 5.2.1, we describe how legacy SP apps are enabled to use wallet-based authentication. Since OIDC Auth Code flow, which is an established defacto standard, requires identity information to be communicated in a backchannel, the SPs backend needs to be able to retrieve an identity token from the eIDAS node connector's endpoint. When using wallet-based authentication, however, the identity token is generated in the eIDAS app on the user's device. We, therefore, extended the eIDAS node connector to accept identity tokens from the eIDAS app. This identity token can then be retrieved by the SP backend using the Auth Code from the OIDC response like in a normal eIDAS authentication flow.

5.2 eIDAS App

We have implemented the app in Kotlin for the Android platform. However, we designed the app using only features for which equivalent ones exist on the iOS platform.

The eIDAS app is a bridge between service provider and identity provider. The eIDAS app handles OIDC Auth Code flow requests from the SP and acts like an IdP. It provides the user with two options. First, the user can authenticate using the normal eIDAS flow. This is described in Section 5.2.2. The second option is to authenticate using a wallet app, which is described in more detail in Section 5.2.1.

From the perspective of the IdP, the eIDAS app acts like an SP. The eIDAS app sends an authentication request to the IdP and receives the response.

Handling Requests with the eIDAS App. Upon installation, the eIDAS app claims the URL of the eIDAS connector. This step enables opening an authentication request in the eIDAS app instead of the browser.

Usually, the SP app would send an authentication request to the eIDAS node connector of the SPs home country. If the eIDAS app wants to intercept such requests, it would need to register every eIDAS node connectors URL. Since app links need to be claimed at compile time and it is not feasible to release a new version of the eIDAS app whenever an eIDAS node connector URL is added or changes, we introduce a relay service. It has the simple task of redirecting the user to a supplied query parameter. With the relay service in place, the SP app sends the request not directly to the eIDAS node connector, but puts its URL in a query parameter to the relay service. This has the advantage that the eIDAS app only needs to register the URL of the relay service to be opened upon an authentication request, while retaining full browser compatibility.

5.2.1 Legacy SP Wallet-Based Authentication

As argued before, we provide the user with the option to use wallet-based authentication with legacy SPs. The protocols used for wallet-based authentication, however, differ from established ones used in production.

The eIDAS app supports the OIDC SIOPv2 protocol ((Yasuda et al., 2023)), where identity attributes are asserted using verifiable credentials.

From a high level perspective, the eIDAS app translates VCs from the SIOPv2 response to an ordinary OIDC response for legacy SP apps. After the user selects wallet-based authentication, the eIDAS app sends an OIDC SIOPv2 request to the Wallet app. The response contains verifiable credentials in the form of verifiable presentation. Next, the eIDAS app validates and verifies the VCs. After a successful verification, the eIDAS app generates and signs an identity token from the information provided by the VCs. Since OIDC Auth Code flow requires the identity information to be communicated in a backchannel, the identity token is sent to the eIDAS node connector. The response to the SP app is an OIDC Auth Code, which is subsequently used by the SPs backend to retrieve the identity token from the eIDAS node connector.

5.2.2 eIDAS Authentication

For regular cross-border authentication flows, the eIDAS app handles communication between the eIDAS node connector, the respective nodes of the SP and IdP home countries, and the eIDAS proxy. The eIDAS app follows redirects and forwards *Security Assertion Markup Language* (SAML) and Light messages between the involved components.

When an eID app is installed and has claimed the IdP URL, it will be opened for authentication. If no such app is installed, a so-called custom tab is opened in the eIDAS app to render a web UI of the IdP for the user to enter their credentials.

In any of the two cases, the response of successful authentication is handled by the eIDAS app again. Like before, the eIDAS app passes messages between the involved components. As a last step, the eIDAS app opens the redirect URI specified by the SP app with the OIDC Auth Code. This causes the SP app to be opened and the eIDAS App is done at this point. The SP app's backend can now retrieve the identity information using the OIDC Auth Code from the response.

In summary, the eIDAS app pushes a mobile-first incarnation of the established eIDAS cross-border authentication flow. When used in conjunction with a wallet app, however, it provides wallet-based authentication also to established service providers. More details on how attributes are added to such a wallet app are provided in the following section.

5.3 Wallet and Provisioning Service

The basis for implementing remaining central parts of our solution, the wallet app and the provisioning service, is an open source multi-platform library implementing the VC data model, which we have developed specifically to fill this gap[8]. This library is used up by the native wallet apps (i.e. in Kotlin for Android and Swift for iOS) as well as the provisioning service (in Kotlin for the JVM).

The wallet app and the provisioning service interact using a well-defined protocol to load the identity attributes of the user onto the mobile device. This shared communication protocol is implemented along the lines of the ARIES RFC 0453 for issuing credentials[9]. During the provisioning, the provisioning service acts as an SP towards the national eID system and transforms the received identity attributes into a format suitable for the wallet app. The attributes are encoded as verifiable credentials, which is in line with the technical specification of the EUDIW, and bound to key material securely stored on the mobile device. The authenticity of this key material is ensured using key attestation, although this time, the procedure is carried out between the wallet app and the provisioning service.

When receiving an authentication request from the eIDAS app using OIDC SIOPv2, the user is presented with a screen confirming the release of their identity attributes. The wallet app creates a verifiable presentation wrapping the requested VCs which have been signed by the provisioning service. The VP as well as the SIOPv2 response are signed with the hardware-bound key that was attested before by the provisioning service and sent back to the eIDAS app. The eIDAS app has the responsibility to validate and transform this response into a traditional OIDC response, as described above.

[8]https://github.com/a-sit-plus/kmm-vc-library

[9]https://github.com/hyperledger/aries-rfcs/tree/main/features/0453-issue-credential-v2

In summary, this results in a chain of trust, which ensures that all identity information is authentic: The SP trusts the eIDAS node connector. The connector vouches for the eIDAS app. The eIDAS app verifies the authenticity of VCs received from the wallet app. The wallet app is trusted by the provisioning service. The provisioning service providing attributes to the wallet app is authorized by a national eID system to do so, by acting as an SP towards it.

6 CONCLUSIONS

The sustained trend towards mobile usage scenarios requires new concepts and solutions for secure and reliable cross-border authentication based on national electronic identities. The EC has recognized this need and has proposed the concept of a EUDIW. Fully-fledged and widely adopted implementations of this concept can be expected to be available in a few years.

As a first step towards this ambitious goal, we have proposed a first rudimentary implementation of a digital-identity wallet that complies with the provisions of the EC and their proposal. The contribution of our solution described in this paper is threefold: **Proof of Concept:** Our proposed wallet solution demonstrates that the basic concept proposed by the EC is feasible and can be realized with current technologies. **Integration Support:** Our proposal provides service providers, i.e., operators of online services, with a working wallet solution, which they can integrate and use for testing against their services. Integration tests are facilitated by supporting a legacy mode of operation, which hides most wallet-related specifics from the service provider. **Enhanced Security:** The proposed solutions integrate remote-attestation features and hence successfully evaluate the usefulness of this cutting-edge technology for secure wallet solutions on current smartphone platforms.

To summarize, the proposed wallet solution offers benefits for various stakeholders involved in the developments and use of the upcoming EUDIW. Our wallet solution will be further developed and tested. Results and findings will then support the implementation of the EUDIW.

ACKNOWLEDGEMENTS

This paper is based on work conducted in the project *mGov4EU*. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 959072.

REFERENCES

Abraham, A., Schinnerl, C., and More, S. (2021). Ssi strong authentication using a mobile-phone based identity wallet reaching a high level of assurance. In *SECRYPT*, pages 137–148.

Ali, T., Nauman, M., Amin, M., and Alam, M. (2010). Scalable, privacy-preserving remote attestation in and through federated identity management frameworks. In *2010 International Conference on Information Science and Applications*, pages 1–8. IEEE.

Gaehtgens, F. (2022). Hype cycle for digital identity, 2022. Technical Report ID G00770428, Gartner.

Jacobino, S. and Pouwelse, J. (2022). Trustvault: A privacy-first data wallet for the european blockchain services infrastructure. *arXiv preprint arXiv:2210.02987*.

Kubach, M. and Roßnagel, H. (2021). A lightweight trust management infrastructure for self-sovereign identity. *Open Identity Summit 2021*.

Prünster, B., Palfinger, G., and Kollmann, C. P. Fides – Unleashing the Full Potential of Remote Attestation. pages 314–321.

Schwalm, S., Albrecht, D., and Alamillo, I. (2022). eidas 2.0: Challenges, perspectives and proposals to avoid contradictions between eidas 2.0 and ssi. *Open Identity Summit 2022*.

Sporny, M., Longley, D., Burnett, D., Kellogg, G., and Allen, C. (2021). Verifiable credentials data model 1.0. W3C Recommendation 2021-06-08, World Wide Web Consortium (W3C).

Yasuda, K., M., J., and T., L. (2023). Self-issued openid provider v2. https://openid.net/specs/openid-connect-self-issued-v2-1_0.html. Accessed: 2023-03-14.