EPiC Series in Computing
Volume 105, 2025, Pages 136–145
Proceedings of EUNIS 2024 annual congress in Athens

wwWallet.org: a Cloud Based Non-custodial Digital Identity Wallet.

Panagiota Stamatopoulou, Charalampos-Michail Katimertzis, Grigorios Katrakazas, Emmanouil Koukoularis, Angelos Ioannis Lagos, Nikos Voutsinas
Greek Universities Network
pstamatop@gunet.gr, charkat@gunet.gr, g.katrakazas@gunet.gr, koukoularism@gunet.gr, ailagos@gunet.gr, nvoutsin@gunet.gr,

Abstract

wwWallet introduces an innovative approach to Verifiable Credentials management, combining the convenience of cloud-based wallets with the enhanced privacy features of edge wallets typically deployed as native mobile applications. It is an open-source, web-based platform that surpasses the limitations of mobile devices while employing a non-custodial approach for users' cryptographic key control. The decentralized Root of Trust is a pivotal feature of wwWallet's design, bolstering both the security and privacy of user data. Aligned with the current version of the European Digital Identity Architecture and Reference Framework, wwWallet adheres to evolving specifications, ensuring its sustained relevance, compliance, and interoperability in the digital identity landscape.

State-of-the-Art Analysis

As digital identity wallets have become the vehicle of identity revolution, they have attracted a lot of attention from the research community, resulting in multiple technological innovations and numerous implementation approaches. The State-of-the-Art (SOTA) analysis reveals a rather fragmented landscape of solutions showing different novelty characteristics and market applicability. To correctly identify the parameters to be considered, it is important to note that Digital Identity Wallets is only one pillar of the Digital Identity Ecosystem. Many design options of Digital Identity Wallets are defined not as part of the wallet itself but in the context of frameworks and specifications that cross all the components of the issuer-wallet-verifier paradigm. Nevertheless, the most important design parameters are considered below in order to comprehensively evaluate and compare different options.

Installation mode. Smartphones and mobile native apps which are made available on the app stores are the most compelling proposition for providing users with a secure and convenient digital identity. This is attributed to the optimized user experience, and the ability to leverage device-specific features. This includes the options for biometric authentication, push notifications, and the tamper-resistant hardware enclave which are available on modern mobile devices. The alternative approach is web-based wallets. Web wallets are accessed through a web browser, do not require installation and may be available across various devices and platforms. A hybrid approach is also feasible on the basis of the Progressive Web Apps (PWA), which combines the user experience of mobile wallet app and the flexibility of web wallets.

Private key control. Non-custodial or self-custody wallets give users full control over their private keys. This approach enhances security and is aligned with the principles of decentralization, offering self-sovereign control. This mode of operation is realized either by invoking a remote Qualified Signature Creation Device, which is managed by a Qualified Trust Service Provider, or by leveraging the local tamper-resistant hardware device. The use of QSCDs, either remote (e.g. an HSM) or local (e.g. TPM/TEE) is one of the technical prerequisites to qualify a wallet solution for high-assurance identity credentials, such as Person's Identification Data (PID). Non-custodial wallets, when combined with native mobile apps, provide a higher level of security, and the ability to support offline scenarios. On the other hand, this approach raises concerns regarding the backup and recovering process of private keys and the dependency on mobile manufacturers implementations for signing and storing crucial identity data of citizens. Custodial wallets are managed and hosted by a third-party service provider, which usually also manages the private keys on behalf of their users. This model offers increased convenience at the cost of guaranteed privacy. In principle custodial and non-custodial wallets can be delivered via native mobile app, or as web wallets, using either local or remote QSCD. For example, a mobile app could use a cloud backend service which in turn accesses a remote QSCD by using the Cloud Signature Consortium specification for remote electronic signatures. Likewise, a web wallet implementation could use the platform's security enclave by using the CTAP2 of the FIDO Alliance and the W3C WebAuthn PRF extension to derive key material.

Signatures schemes & selective disclosure. The selective disclosure refers to the ability of an individual to granularly decide what information to share. Along with the isolation of the issuers from any information regarding the use of personal data by the holders, the selective disclosure is considered one of the digital identity wallet ecosystem cornerstones. Unlinkability and data minimization privacy properties are directly derived by the chosen selective disclosure mechanism. Multiple protocols have been proposed and used, with IETF SD-JWT being the dominant one, because of its simple approach and the adoption by the EUDI ARF for Type1 wallets. A similar approach to SD-JWT which is also based on a signed collection of salted attribute hash values is used by the mDL-MSO as part of the ISO/IEC 18013-5. However, concerns have been raised regarding the unlinkability strength of these specifications due to issuer signature tracking, if multiple verifiers collude. Alternative multi-message signature schemes (BBS+ and CL) which can present selected attributes together with proofs of the user's possession of the corresponding complete credential have been thoroughly researched and developed. These signature schemes enable enhanced privacy preserving ZKP properties. BBS+ comes along with Linked Data Proofs assertions utilizing the LD-Data data model format. One of the benefits of the LD-Data model and the approach of the JSON-LD ZKP BBS+ is that it embeds into the VC all the semantic information without external dependencies e.g. to a DLT to provide contextual definitions. However, BBS+ has certain limitations on the support of ZKP predicates like range proofs, membership proofs or logical combinations of them. Another family of novel ZKP protocols is based on the sk-SNARK, which also proves the possession of information without revealing the source data. sk-SNARKs applicability seems to be limited by its cryptographic complexity, since this scheme is based on algebraic equations. While ZKP via BBS+ seems to be one of the most promising technologies, the most important barrier of adoption at least by the EUDI ARF, is the lack of support by the tamper-resistant hardware devices and the fact that the corresponding crypto algorithms have not been certified by SOG-IS, yet.

Exchange protocols. OID4VC is a set of 3 protocols that enable self-controlled authentication (SIOPv2), issuance (OID4VCI), and presentation (OID4VP) of verifiable credentials expressed in any format, including W3C VC data model, ISO/IEC 18013-5 (ISO mDL) or AnonCreds. OID4VCI/VP along with the ISO/IEC 180013-5 have been chosen also by the EUDI Wallet ARF and seem to be the most commonly used communication protocols in the digital identity wallets market. The ISO mDL, is designed to facilitate the presentation and verification of driver licenses, either in offline mode (device retrieval) or in online mode (server retrieval). In the device retrieval mode the mobile device and the reader communicate over BLE or NFC and the messages follow the CBOR format. Server retrieval is performed via

OID4VP protocols using the JSON structured messages. Additional exchange protocols worth noting are a)the Credentials Handler API (CHAPI) which focuses on web browser flows and web applications to interact with the user's credential storage and presentation mechanisms, b)the Wallet And Credential Interactions (WACI) which is using the DIDComm v2.0 messaging protocol for the exchange of JWTs between the issuer the wallet and the verifier, c)the Verifiable Credentials API which leverages the recent VCDM2.0 and defines a set of RESTful API endpoints for the exchange of VCs encoded as JSON-LD.

Revocation Management. Revoking VCs, especially for natural persons, poses a challenging problem due to conflicting functional and privacy requirements. Starting with privacy considerations, the European Union Digital Identity and Attributes Regulation Framework (EUDI ARF) mandates that "*Qualified Electronic Attestation (QEA) Providers provide information or the location of services that can be used to inquire about the validity status of the QEAs, without having the ability to receive any information about the use of the attestations.*" This explicitly dictates that issuers should not be able to trace the usage of their issued credentials through the revocation mechanism. Other crucial aspects of the revocation mechanism include a) the performance overhead for the issuer, the holder, and the verifier, b) the verifier's ability to monitor the credential status beyond the presentation phase, c) the holder's ability to limit the duration for which a verifier can check the VC status, and d) whether the verification process requires the holder to be online.

Various strategies have been proposed to address the issue of VC revocations, including a) issuing short-lived VCs, b) directly checking the revocation status at the issuer side, which raises privacy concerns, and c) using a bitstring mechanism that enables a verifier to check the VC status in a privacy-preserving manner.

In addition to the W3C Status List 2021 specification, EBSI has recently introduced additional VC status mechanisms. These mechanisms allow the verifier to retrieve status information either via an EBSI registry through the holder, concealing information from the issuer and maintaining privacy regarding the use of VCs.

Key Management. In the landscape of key management options for verifiable credentials, various decentralized identifier (DID) methods offer secure and flexible implementations. The did:key method employs self-contained identifiers using direct public key representation, while did:ebsi leverages the European Blockchain Services Infrastructure (EBSI) and did:ion aligns with the Microsoft Identity Overlay Network (ION), emphasizing scalability. did:web simplifies DID creation through web URLs, did:keri focuses on decentralized key management, did:indy is tailored for privacy-focused applications, and did:peer facilitates peer-to-peer network communication. did:jwk uses JSON Web Key (JWK) format for standardized key representation.

Differentiation factors of the various DID methods are the ability for key rotation and key history. Key rotation, a critical security feature, ensures the periodic updating of cryptographic keys associated with a DID. Additionally, the key history property allows the maintenance of historical key records for auditing purposes and ensuring a seamless transition of trust during key updates. Did:key supports neither key rotation, not key history, thus despite its simplicity makes it not ideal for production solutions. On the other hand did:ebsi because of the underline blockchain infrastructure provides both key rotation and key history. Key rotation and key history are also supported by did:ion, and did:keri.

Trust Management: Another key aspect of the digital identity wallet design is the trust management which is based on. Trust management encompasses the mechanisms that evaluate the reliability, security, and authenticity of entities participating in digital identity interactions. Noteworthy trust management frameworks are a)the OpenID Connect Federation, which uses JWTs and REST APIs to represent trust chains and trust marks, b)EBSI Trust chain, which provides trust list for issuers, schemas and verifiers anchored on the EBSI ledger c)Verified issuer certificate authority list (VICAL), which is defined by the ISO 18013-5 and represents a simplified master list of certificates for issuers.

wwWallet

Design

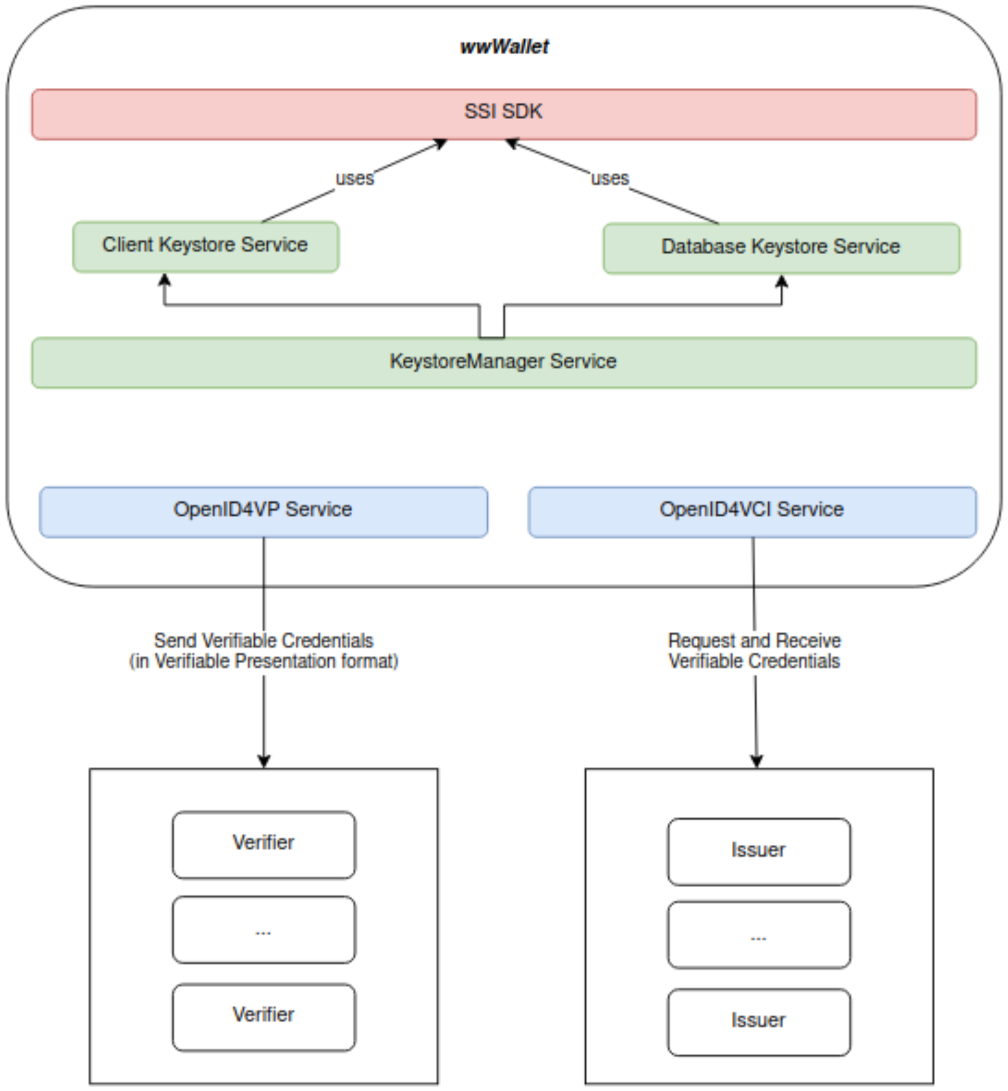
The wwWallet is an open source, web-based, non-custodial wallet solution with Progressive Web Application (PWA) capabilities, which can be considered as the alternative to the traditional native wallets that are already dominating the wallet market. The core architecture of wwWallet features a versatile design that facilitates integration of different components. Its modular structure enables the effortless incorporation of diverse key management methods and trust frameworks, ensuring adaptability and scalability. Through well-defined interfaces by using socket messaging, developers can easily plug in new modules or replace existing ones without disrupting the core functionality of the wallet. This inherent flexibility empowers wwWallet to evolve with changing requirements and technological advancements, fostering innovation and efficiency in its utilization.

The currently supported key management method of wwWallet is the encryption of users' keys on the frontend by using the `prf` extension capability. This process is facilitated not only by the user's FIDO passkey, but also through the device's authenticator capabilities, such as those found in Android devices. The platform leverages OpenID protocols for the issuance and presentation of verifiable credentials, with these processes primarily handled on the backend. Moving forward, we aim to migrate these functionalities to the frontend, making the client-side application self-contained and independent of backend services . One additional next step of the wwWallet project is the implementation of different trust frameworks other than the EBSI Trust Framework, with the OpenID Federation being the most prevalent choice.

wwWallet architecture has adopted a component-based and modular design steering away from monolithic structures. This strategic approach provides flexibility and adaptability, enabling interchangeability without disrupting the overall implementation. Crucial to this modularity is the strategic use of dependency injection techniques and the integration of configurable components. Notable examples of these pluggable elements are trust frameworks like OpenID Connect Federation or EBSI Trust Chain that can be used alternately, key management methods for decentralized identifiers (DIDs) and adaptable data models for varying use cases. Importantly, wwWallet architecture allows configurable signature schemes and supports multiple credential formats, ensuring a dynamic setup for diverse applications.

Key components of wwWallet architecture are the following:

- SSI SDK: The SSI SDK is a distinct packaged software which abstracts the complex and repetitive functions that are required to be executed from various components and different software stacks.
- KeystoreManager Service: Since the wallet supports different types of keystores, the KeystoreManager Service exposes an interface which can be implemented in order to add new keystores.
- OpenID4VP and OpenID4VCI Services: These services form the upper layer of the inbound and outbound communication protocols that are required to receive and send credentials.



Key Management

wwWallet is currently leveraging the WebAuthn protocol's PRF extension to dynamically derive encryption keys, such as the prf key, enhancing the security of wallet contents through secure encryption and decryption processes that are tied closely to the user’s WebAuthn authentication. This method efficiently generates session-specific keys and minimizes the exposure of long-lived encryption keys which are used to encrypt the wallet's private data. Building upon this foundation, wwWallet plans to transition from using PRF to implementing Asynchronous Remote Key Generation (ARKG). This shift aims to harness ARKG's capabilities for generating seeds and deriving keys for both signing and Key Encapsulation Mechanisms (KEM), enabling the creation of single-use, unlinkable public keys and promoting even more secure, ephemeral key exchanges. Integrating ARKG, wwWallet strengthens its defense against quantum threats, ensuring future-ready security.

Progressive Web Application Capabilities

wwWallet is on track to evolve into a Progressive Web Application (PWA), which is a type of application software delivered through the web, essentially fusing the features of web pages and mobile apps. This choice will give wwWallet an edge compared to its competitors because it will provide its users with a highly accessible platform and a user-friendly experience, resembling that of a native application. More specifically, wwWallet is set to be downloadable as both a desktop and smartphone application, while simultaneously offering offline capabilities and protection against slowdowns during periods of intense traffic and server load.

While evolving to a PWA would expand the wallet’s usage and broaden its audience, it remains uncertain whether Apple will embrace this transition, considering their traditionally strict app store policies. Android platforms, however, typically embrace PWAs, allowing them to be smoothly integrated into their ecosystem.

International Exposure and Interoperability

wwWallet has gained international attention through its participation in key events focused on digital identity and credential management. Currently, we are preparing for the EBSI Demo Days, where GUnet, in collaboration with the University of Lausanne, will utilize wwWallet to store and present university credentials in a cross-border scenario.

Additionally, wwWallet has already showcased its potential during JFF Plugfest 3 - VC for Education (October 2023), where it facilitated a mock scenario of verifiable credentials management between a Greek and a Canadian university. The wwWallet Launchpad, a complementary service, is provided as an extra means of issuing and presenting mock verifiable credentials to test interoperability and for demonstrative purposes.

Security Considerations

The previously outlined design has some drawbacks, while some of the intended advantages are debatable.

Rotating the main encryption key

The **main encryption key** (256-bit AES-GCM encryption key) cannot easily be replaced since that would require updating each corresponding key wrapping. This could potentially be solved by using ECDH to derive the wrapping key for the main key.

Cleartext session key

The **session key** (256-bit AES-GCM encryption key) is used for user convenience: once logged in, the session key can be used for the duration of the session to access encrypted data. However, this might not improve security much; it may not be materially less secure to simply store the **main encryption key** itself in cleartext in session storage.

Weak encryption at rest

Although the user's wallet contents itself are stored in encrypted form, we also store all encryption keys needed to decrypt them, whereas It would be preferable to retrieve or derive the encryption keys from data not stored in cleartext on the client device.

Encrypting privateKey twice

Likewise, encrypting the user's proof signing secp256r1 private key twice may not materially improve security either. An adversary with read access to the other `PrivateData` members will likely also have read access to both of local storage and session storage, thus having access to the keys needed to decrypt the private key anyway.

Software keys, not hardware keys

All cryptographic keys used by wwWallet are fundamentally software keys, held in browser memory. This is due to technical limitations, some of which may change in the future.

However, note the difference between encryption keys and signing keys. Encryption and decryption inherently involves exposing the cleartext anyway, so encryption keys do not need to be kept secret from parties accessing the cleartext.

The WebAuthn `prf` extension leverages the HMAC secrets that a hardware FIDO security key can provide and derive from these the encryption keys. Thus the encryption keys are software keys held by the browser, but the WebAuthn authenticator holding the PRF key is needed in order to access the PRF outputs. This is sufficient hardware binding for encryption keys, assuming the browser is benevolent.

Ideally the user's private key should be a hardware-bound key, so that even if the main encryption key or the `PrivateData` (contain both sensitive data, namely the private key, and non-sensitive data, including the user's DID and public proof signing key) is exposed, an unauthorized party would still not be able to sign proofs on behalf of the legitimate user. This is not currently possible due to technical limitations. Although WebAuthn authentication keys may be hardware-bound, they do not support signing arbitrary data. Future extensions to the WebAuthn API might add the possibility to sign arbitrary data with a hardware-bound private key.

References

1. Meet wwWallet.org. GitHub: https://github.com/wwWallet
2. Lundberg, E. (n.d.). Webauthn ARKG Extension. Retrieved from GitHub: https://github.com/Yubico/webauthn-arkg-extension