Defining Unified Signature API for mobile apps to integrate with secure signature creation devices (SSCDs)

1st Ammar Bukhari
Producht Manager - Identity Technologies
Methics Oy
Espoo, Finland
ammar.bukhari@methics.fi

2nd Jarmo Miettinen
Chief Executive Officer
Methics Oy
Espoo, Finland
jarmo.miettinen@methics.fi

3rd Muttukrishnan Rajarajan
School of Science and Technology
City University of London
London, United Kingdom
r.muttukrishnan@city.ac.uk

***Abstract*—Secure Signature Creation Devices (SSCDs) are building blocks for performing a legal value digital signature. As the world moves more towards digital transactions, it increases reliance on digital signatures. Interfacing and enabling a SSCD to work with a mobile phone application allows these applications to perform digital signatures of legal value (i.e. Qualified Electronic Signatures - QES, in EU). However, the lack of standardized APIs for interfacing SSCDs with mobile apps poses significant hurdles to widespread adoption, and poses many challenges.**

This paper introduces a novel approach to address the challenges of integrating Secure Signature Creation Devices (SSCDs) with mobile applications by proposing a solution called MUSAP utilizing Unified Signature API. MUSAP enables secure communication between mobile apps and SSCDs, ensuring the protection of private keys and compliance with legal frameworks such as the eIDAS regulation.

Implementation details and use cases demonstrate the practical application of the Unified Signature API, showcasing its versatility in supporting both centralized and decentralized identity technologies. The entity issuing identity assertions (certificate or verifiable credential) holds the responsibility for verifying both user's identity, key material and provide a Level of Assurance (LoA) selection mechanism. Key material verification can be achieved by Controlling the key generation process, verifying device trust and relaying key attestations. MUSAP is released as open-source solution in Github.

***Index Terms*—SSCD, eIDAS, Digital Signature, EDIW, MUSAP**

I. Introduction

In today's world we are surrounded by digital transactions which are only going to increase exponentially. Entities involved with these digital transactions do verify authenticity, integrity and non-repudiation of the digital signatures in order to trust. However these digital transactions have been regulated with different regulations. In EU, the first formal Directive 1999/93/EC served as a cornerstone document within the union member states, which allowed them to implement national eID schemes, for example Estonia in 2002, Austria in 2005, Finland in 2008 with eID scheme ID cards and Finnish Mobiilivarmenne (X.509 based Mobile ID) was launched 2011. etc.

After more than a decade, eIDAS (910/2014) also relied on digital signatures, and now today in 2024, eIDAS 2 (2024/1183) to replace eIDAS is published to introduce European Digital Identity Wallets (EDIW or EUDIW). These wallets will need to be made available for end-users in European countries by 2026.

In order to create signatures Digital Identity Wallets need to interface with a certified Security Signature Creation Device (SSCD). In this scenario, we assume that an application on a smartphone lacks the capability to independently serve as a secure signature creation device. While eIDAS 2 (2024/1183) directive has been adopted by EU member states, it requires the Identity Wallets to enable the end-users to create Qualified Electronic Signatures/Seals (QES), which are of the highest level of digital signatures.

In Europe, in order to comply with the requirements for the Qualified Electronic Signature i.e. QES, signature creation devices need to have compliance with the Common Criteria 3.1 standards, more specifically with the EAL 4 level or higher. [1], [2] These standards rely on different Protection Profiles (PP), which are documents created by groups of security professionals referring to different international standards, and contains set of requirements for related products. One SSCD needs to rely on a PP which has been jointly approved by SOG-IS, and this SSCD when certified at the highest level in Europe allows these products to act as a Qualified signature creation device (QSCD). Though, SSCD is a broader term that can encompass various types of signature creation devices, whereas "QSCD" specifically refers to a subset of SSCDs that adhere to the highest security standards defined by the European eIDAS Regulation for creating qualified electronic signatures within the EU.

Despite the maturity of digital signatures, the operational challenges of interfacing signature creation device of the user's choice, remain a hurdle for widespread adoption.

Interfacing SSCDs with mobile device (mobile application) is a challenge. Absence of standardized APIs for integrating

MUSAP is funded through European Union's Horizon 2020 research and innovation program through the NGI TRUSTCHAIN program under cascade funding agreement No. 101093274.

SSCDs into smartphone applications further increases these challenges, limiting the development of comprehensive and user-friendly MUSAPs. Similarly, end users who wants to create digital signatures to secure their transactions from mobile device face a series of challenges, mostly in the incompatibility issues with mobile devices [3] which do not offer interfacing support.

The main motivation to develop a MUSAP based on interfacing different SSCDs with the mobile phone is to develop a standardized way to abstract the complexities of different signature creation device technologies, paving a way to interface one or more creation device. It is noted that eIDAS-based cross-border authentication first needs to be made more user-friendly on mobile device for any digital identity wallet to succeed. [4]. Interfacing multiple SSCDs which are on different security level of assurance, will allow mobile device to create signatures with multiple assurance levels.

The main contributions of this paper are:

- introducing a way to interface different SSCDs to a smartphone application by developing a unified signature interface API.
- an open-source component released in Github at TRL 8. It uses two components, MUSAP Library, which is packaged natively with Android or iOS apps, and MUSAP Link service docker which is placed on back-end of the app
- extending the existing state-of-the-art identity systems. MUSAP Libraries will be used to implement a use-case driven identity management system deployment.

ENISA, European cybersecurity agency released Digital Identity Standards in 2023 where they highlighted a need to develop a API from mobile phone to be able to call secure elements (SSCDs). [3]. Such API is crucial for establishing interoperable identity systems.

The rest of this paper is structured as follows. Section II provides an overview of a related work dealing with the problem of digital signatures and signatures creation devices i.e. SSCDs or QSCDs. Section III proposes our approach and describes method used for building a unified signature API. Section IV provides implementation details of our MUSAP in different environments. Section V provides discussion of our method, and finally section VI concludes this paper.

II. Overview of Digital Signatures and SSCDs

A. Background

Secure Signature creation devices are usually a standardized devices relying on a combination of hardware and security controls, necessary to store the private key of the user and maintain it under user's sole control. SSCD securely stores the private key locally or remotely the private key which cannot be exported. When a user wants to sign a digital document, SSCD generates a digital signature using the private key and the document digest. Annex 2 of eIDAS [5] regulation lay conditions for signature creation with use of SSCDs.

In EU, an eIDAS dashboard maintains a list of SSCD/QSCD in EU. These devices are usually complying to certain standards such as CEN 419 241-2 CEN 419 221-5 for remote signature creation devices, CEN 419211-1 and CEN 419211-2 for SIM based local signatures, CEN 419211-3, CEN 419211-4, CEN 419211-5 etc. for ID card based devices. In addition, each member state has different SSCDs or QSCDs in play with their national eID schemes, which allows end-user to to use them for secure authentication and creating signature.

Since eIDAS 2 regulation is published, European Digital Identity Wallet security is based on asymmetric keys and qualified signatures, making wallet act as a QSCD [6].

ENISA mentions the need for a harmonied interface that allows direct access to the internal and external mobile device's cryptographic security which will allow wallets to perform cryptographic operations [3]. Such harmonised interface could be a unified signature API which streamlines the need for interfacing SSCDs with mobile apps.

B. Basic Requirements

With the maturity of eID MUSAPs or digital signatures, there is a consensus (add reference) on security requirements of creating signatures and agreed algorithms. Universally for a digital signature to be legally recognized and enforceable, several components and requirements must be met. Most common requirements listed in [7]–[9]. These typically include:

1) Signer's Identity: The Signer's Identity must be a valid identifier within the X.509 PKI certificate's subject field, providing sufficient information about the individual to ensure their uniqueness and recognition.
2) Digital Signature Algorithm: The digital signature must be generated using a cryptographic algorithm that provides integrity, authenticity, and non-repudiation. Common algorithms include RSA, DSA, and ECDSA, etc.
3) Digital Certificate: The signer must possess a valid digital certificate issued by a trusted Certificate Authority (CA) or Qualified Trust Service Provider (QTSP). The certificate binds the signer's identity to their public key and is used to verify the authenticity of the digital signature.
4) Private Key: The signer must have the private key that matches the public key in their digital certificate and the private key must be under Signer's sole control. Possession of the private key is no longer required. The private key is used to generate the digital signature and the control of the key must be secure enough to prevent unauthorized use.
5) Hash Function: A secure hash function is used to generate a unique fixed-size digest of the data being signed. The hash function must be recognized by the legislation. The digital signature is created by encrypting the hash value and optional extra information with the Signer's private key.
6) Timestamp: Optionally, a timestamp from a trusted Timestamp Authority (TSA) can be added to the digital signature to prove the time at which the signature was

created. This helps establish the temporal validity of the signature.

7) Secure Signing Environment: The signing process should be performed in a secure environment to protect the private key from theft or tampering. This may involve using tamper-proof devices which provides a secure environment for key generation and storage. This may include hardware backed, secure elements (SE), hardware security modules (HSMs), etc. Such functionalities are provided by SSCD.
8) Legal Framework: Finally, the digital signature must be used in accordance with applicable laws, regulations, and standards governing electronic signatures and digital transactions in the relevant jurisdiction. This may include compliance with eIDAS regulations in the European Union or the Uniform Electronic Transactions Act (UETA) and Electronic Signatures in Global and National Commerce Act (ESIGN) in the United States to name a few.

By meeting these requirements, a digital signature can provide a legally binding mechanism for signing electronic documents, authentication and transactions. However, the specific legal validity of digital signatures may vary depending on the jurisdiction and the nature of the document or transaction. It's essential to consult legal experts familiar with electronic signature laws in particular jurisdiction to ensure compliance and enforce-ability.

C. Related Work

As SSCD is used with a mobile app, it provides protection and resistance to attacks, and maintains users key materials secure under their control. In this article we focus only those technologies which can be used for public services with existing open standards and commonly available smart phone devices. Hence, we have omitted Oma/ETSI Smart Card Web Services or GlobalPlatform Trusted Execution Environment, primarily utilized in enterprise applications.

Many APIs to interface keystores and SSCDs were studied. Typically, mobile apps that have an SSCD interfaced are either Hardware Security Module (HSM) like eIDAS remote signing, secure tokens or bank specific tools. These are usually provided only as a binary software. There are different individual APIs to interface one SSCD with mobile app, they are like:

- GlobalPlatform (GP) has published Open Mobile API (OMAPI), which enables mobile apps to access GP secure-elements in mobile devices [10].
- Cloud Signature Consortium (CSC) has published API to request Remote Signatures, but interfacing with mobile apps is not feasible [11].
- Github hosts various FIDO2 libraries and several signing workstations libraries, which relies either on keystore, USB token. These were at a lower technology readiness level

Current status is that there are no public implementations to interface multiple SSCDs with one device.

TABLE I
STATE-OF-THE-ART MAPPED TO GAPS ADDRESSED BY MUSAP

State-of-the-art	What MUSAP provides
1. Identifying key is dependent	Universal key identification
2. Key is platform dependent	Universal key selection
3. Verifying eligibility of the SSCD	Key attestation for SSCD keys
4. Mobile provides one LoA	LoA selection mechanism
5. DID not signed with SSCD	Defined DID signing mechanism

The Table 1 describes the current state of art of identity system APIs and how MUSAP provides a fix for these.

In EU, as eIDAS 2 is now officially in force, all mobile identity wallets need to provide capability for mobile app to create qualified electronic signature (QES), which possesses legal equivalence to a handwritten signature and is often considered even more robust in terms of security. These standards for QES rely on dedicated hardware-based security modules to store and generate keys [5], [6], [8], which is responsible for both generating and safeguarding digital signatures. All EDIWs needs to be interfaced with SSCDs for a 'substantial' security assurance level and QSCDs for the 'high' security assurance level [6].

Each MS will decide upon the authentication and onboarding phase – prior to the issuance, both for proximity and remote enrolment. Each MS will have liberty to define their SSCDs for Wallets and end-users to have liberty to choose them.

Moreover, ARF for European digital identity wallets discusses about two different type of configurations for end-user [12] i.e with Level of assurance (LoA) High as Type 1 and LoA Substantial as Type 2. Though in latest versions of ARF from 1.3 onwards, different configuration of wallet was taken out of scope [13], and mentioned in [6] that users having identity with SSCD at LoA Substantial can onboard to the wallet with some additional security mechanism.

III. Proposed Solution

Proposed MUSAP design will allow developers to build applications that can easily integrate with multiple systems without having to learn the details of each individual SSCD interface. MUSAP simplifies the development process, reduces costs, and accelerates time-to-market for new applications utilizing digital signatures with mobile apps. Making it particularly useful in the context of citizen's digital services, where multiple independent services need to interact with each other seamlessly.

As each member state of EU needs to provide wallet to their residents, MUSAP addresses the complexities of SSCD technologies, offering a unified API Library for developers to use MUSAP library in their Android or iOS applications.

In order to address the challenges we need to specify and develop a robust and unified API from the mobile signature app to the security anchor provided by SSCDs. Using the MUSAP with mobile app, allows end-user to have multiple options to choose where they store their private keys.

MUSAP addresses security and convenience, offering a resilient and adaptable implementation for mobile app(s) requiring high level of trust. MUSAP offers end-users methods to diversify their key storage and use existing SSCD (from already deployed Digital ID system). Eventually avoiding the concentration of all keys in a single basket.

Moreover, MUSAP never shares actual cryptographic keys of the SSCD with mobile app. With user's consent, only SSCD metadata is shared to the app.

A. *Architecture and APIs*

While writing MUSAP specification and designing the architecture, team took guidelines from Framework for Designing Cryptographic Key Management Systems (CKMS) outlined in NIST Special Publication 800-130 [14], and eIDAS (910/2014). MUSAP specifications, alongwith APIs and source code is already published in Github with Apache license 2.0.

MUSAP is composed of two components:

1) MUSAP Library → Client-side library for Android and iOS to be packages with end-user app (mobile app)
2) MUSAP Link service → A docker component deployed on server side of end-user app or seperately

Detailed architecture and components of MUSAP is presented in Figure 1.

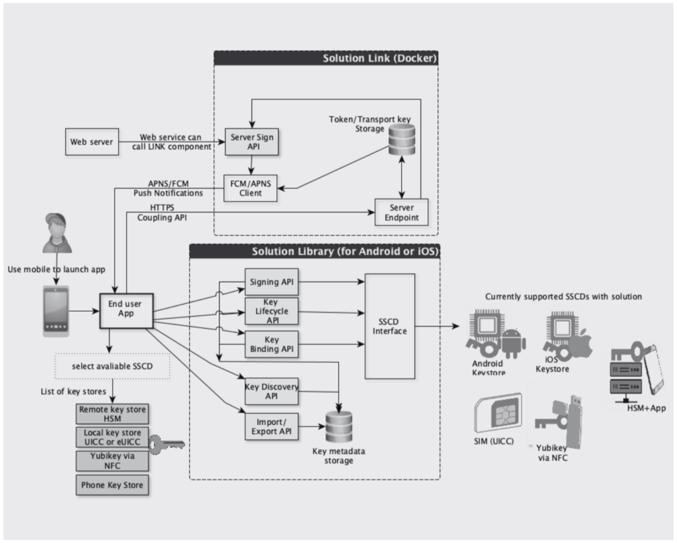

Fig. 1. MUSAP architecture, APIs and components

MUSAP is designed in a way that allows easy integration with mobile apps and offloads key management operations. App can call MUSAP API to call the library in the phone. Later app using the MUSAP Library is linked (initialized/enrolled) with Link through Coupling API. APIs in detail are listed in Specification and API repository.

MUSAP is utilizing 3 APIs, they are:

- Native API (MUSAP API for Android or iOS) which can be used by mobile app to call the library
- JSON API (Coupling API between MUSAP Library (Android/iOS) and Link service)
- REST API (LINK API between Server/WebApp and service) which can be used by web services to request signature.

Native API in mobile apps is used to request key operations and interact with the MUSAP library is core API which links mobile device and library. Native API is further divided into multiple APIs such as Sign, Key lifecycle, Key binding, Key Discovery and import/export. All these APIs can be called by mobile app for required function.

- Sign API: send to sign challenges and other data (for example VC, X.509, DID, etc.
- Key lifecycle API: allows the user to generate new keypairs with specified parameters
- Key discovery API: lists all available keys from SSCDs which can be reached through MUSAP and app.
- Key binding API: used to bind existing SSCD to library and app. Has endpoints to enable SSCD and bind keys
- Key import/export API: used to import/export key metadata data between apps, and for backing up the keys.

These APIs are well documented and explained in MUSAP specification in Github. MUSAP project Github contains links to all API repositories and documentation [15].

B. *MUSAP Library for mobile apps*

MUSAP library can be called in the mobile app. Developers can import MUSAP Android library or MUSAP iOS Library in their Android/iOS app projects. Defines what kind of SSCDs they want to support and its configurations.

MUSAP library consists of the following main components (as illustrated in Figure 1):

- Native API (MUSAP APIs)
- SSCD Interface(s)
- Metadata storage

Moreover, one key feature which MUSAP library provides is to provide validity of SSCD key. In a scenario when MUSAP is able to interface SSCDs, it also provides key attestation to verify the key origin. It is trivial to verify public key origin if they are sourced from mobile operator system via local signatures though SIM card having java Card applet, or from a provider of CEN standards based remote signing. However, if the public key originates from phone's keystore, the registrar cannot validate SSCD's validity without a key attestation. Hence, a unified API attestation proves validity.

C. *Link service for mobile apps*

Link service establishes a connection between web applications and the mobile app using MUSAP library. Link service URL is configured on the MUSAP library (in mobile apps to enable the connectivity with the link service). Registering or linking your mobile app the link service is always optional and a user's choice.

Link composes of a straightforward REST API for seamless integration with external web services. Along with push notification support integration for both Apple's and Google's notification service.

Additionally Link service serves as an essential component, as it facilitates decentralized identity management by providing a secure mechanism that eliminates the necessity for centralized identity management. As decentralized identity components become more mainstream and part of the identity ecosystem, interoperability between x509 and decentralized identity based on Decentralized identifiers (DIDs) and Verifiable Credentials (VC)and digital signatures can be mentioned as factor of utmost importance for its success [6].

Link service flow can be described iteratively as:

1) Link service sends a push notification to smartphone which awakes the mobile app with MUSAP library.
2) MUSAP Native APIs requests Link service for the signature request data.
3) Lastly, MUSAP library sends the signature request data to SSCD interfaced with mobile app.

For additional security there is a transport security layer between MUSAP library and Link service. When enabled, all message payloads between them are encrypted with AES-CBC-PKCS7 Padding cipher with a random *IV*. Messages also have a message authentication code to verify message integrity and authenticity. The message payloads contain a *nonce*, and a *timestamp*. Link service has a configurable duration range to reject old messages and messages that have an already used nonce. Such mechanism prevent replay attacks on Link service, and ensures that each payload is unique. MAC uses encrypt-then-mac scheme. Library encrypts the payload, and then calculates a HMACSha256 of the payload, user identifier, message type, and AES encryption IV. When Link service, or MUSAP library receives a message, it first checks if MAC is correct by comparing the calculated and received MACs. They reject messages with a wrong MAC. The MAC proves the authenticity of the message. Moreover, the encryption between two components is established during the enrollment.

D. Use cases

MUSAP is designed in a way to provide unique use cases for both centralized and decentralized technologies. In centralized technology, MUSAP eases the interfacing of in-scope SSCDs with mobile apps, allowing them to sign any data format. As ARF of EDIW mentions wallet mobile app to support two configurations. Configurations are sets of specific rules and methods that define how EDIW will be built and operate. These configuration are wallets of different types i.e. Type 1 (LoA High) and Type 2 (LoA Substantial) [12]. MUSAP has been used (explained in Section IV about Implementation) to demonstrate a use case where different assurance level SSCDs were interfaced with mobile app. Figure 2 explains high level overview of MUSAP use with identity wallet apps to provide interface with multiple SSCD technologies.

Similarly for decentralized technology, MUSAP can be used to sign DIDs with different assurance levels. As decentralized identity standards get developed and mature, MUSAP paves

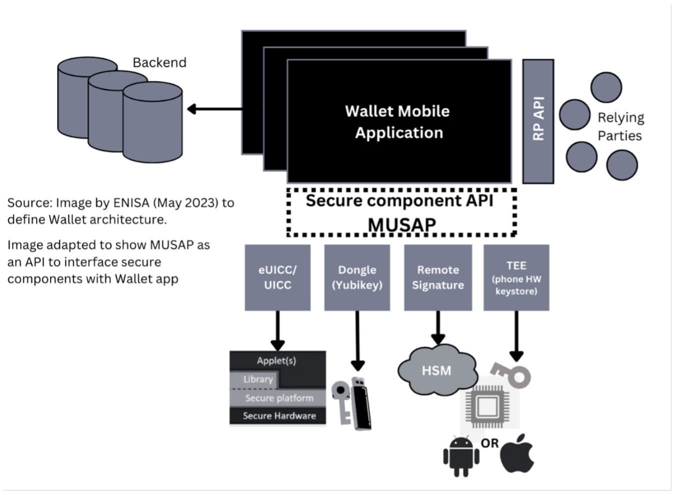
Backend
Wallet Mobile Application
RP API
Relying Parties
Source: Image by ENISA (May 2023) to define Wallet architecture.
Image adapted to show MUSAP as an API to interface secure components with Wallet app
Secure component API
MUSAP
eUICC/ UICC
Dongle (Yubikey)
Remote Signature
TEE (phone HW keystore)
Applet(s)
Library
Secure platform
Secure Hardware
HSM
OR

Fig. 2. MUSAP role as a secure component API for identity wallets to interface different SSCDs

a way for client-secret mode for DIDs and manage cryptographic operations. In this mode of DID operations will ask the client to perform cryptographic operations such as generating keypair(s), signatures, etc. [16].

Especially as decentralized identity services become the main trend of identity services, MUSAP offers an interoperable route from centralized identities towards decentralized model.

IV. Implementation

Main goal of the first implementation version is to define and demonstrate how MUSAP library constructs the sign request. Software implementation goals were to develop a simple and native code (Android/iOS) implementation as an example, which has limited number of dependencies to 3rd party components, it has a clear API interface, and it can be integrated with the client app easily as a native component. Integrator can replace also included 3rd party libraries with their preferred tools.

While the implementation is an open source code, the integrator can see the whole library and its structure, which make it easy to contribute the implementation, extend it or write completely own implementation.

Multiple API call versions are needed for various signing requirements of the client. It is clearly different to just sign a document, sign PAdES LTA, sign Verifiable Credential etc. Additionally, we considered that integrator should be able to add more signing mechanisms when needed.

Complete MUSAP libraries and reference implementation apps have been released in Github with extensive documentation [15]. Moreover, as described in Figure 2, four different SSCDs were interfaced with mobile app in reference implementation. Demo video [17] also demonstrates interfacing 4 different keystores with mobile app.

V. Discussion

Although SSCDs have been around for long, but with emergence of eIDAS2 and the European Digital Identity Wallet has brought renewed focus to digital wallets and their architecture. Since all wallets at highest assurance level need to act as a QSCD and with substantial assurance level as a SSCD [6]. There seems remote SSCD and local SSCD which can be interfaced for wallets.

To ensure maximum user adoption, extensive user study needs to be carried out what end-users prefer as their SSCD technology to ensure their private key stays safe. ENISA has been releasing recommendations related to a need for harmonized interface that allows access to different secure components or SSCDs to allow cryptographic operations for the identity wallet [15]. Currently, eIDAS expert group with participation from each member state and team of experts developing architecture and reference framework for European digital identity wallet is collaborating for an interoperable architecture. Indeed it is a big task which will impact user adoption.

Developed MUSAP library paves a way, which allows mobile apps like identity wallet apps to call different SSCDs of different assurance levels, allowing end-user of apps to have a choice which SSCD technology they feel comfortable to use. Simply put providing a choice to end-users to choose what they trust more to ensure safety of their private key.

VI. Conclusion

In this paper we presented a novel approach to create a unified signature API through proposed MUSAP libraries, allowing mobile apps to easily interface one or more SSCD. Our MUSAP compliments the with requirements of the eIDAS 2 , eIDAS 1 and legacy directive 1999/93/EC of the European Parliament.

As Digital identity space evolves, a Unified Signature API, which interfaces multiple different SSCDs can help with the mass adoption of users. As success of eIDAS 2 hinges upon its user- friendliness, security, and convenience. The feasibility of achieving widespread acceptance is closely tied to the system's adoption, ease of use and its seamless operation on a global scale.

MUSAP provides a standardized and simplified way for developers to request multiple LoA signatures, regardless of the programming language, SSCD platform, or technology (centralized or decentralized) used. This ensure consistency and interoperability across different systems.

The advancements in building the MUSAP were marked by tangible demonstrations, active feedback from end users, constructive business meetings on common principle/goal for future collaboration, and room for future advancements in the MUSAP. As MUSAP is released in Github on TRL 8, it remains open for future work. As EDIW Reference Apps have recently been released, integrating MUSAP to reference apps, or adding more SSCDs in scope (like eID card, eSIM based Mobile ID,etc.) of API will make the service more lucrative for use with mobile apps.

Acknowledgment

MUSAP is funded through European Union's Horizon 2020 research and innovation program through the NGI TRUSTCHAIN program under cascade funding agreement No. 101093274.

References

[1] "Etsi en 419 241-1 trustworthy systems supporting server signing part 1: General system security requirements," https://www.sis.se/en/produkter/information-technology-office-machines/it-security/ss-en-419241-12018, 2018.

[2] "Etsi en 419 241-2 trustworthy systems supporting server signing - part 2: Protection profile for qscd for server signing," https://www.sis.se/en/produkter/information-technology-office-machines/it-security/ss-en-419241-22019/, 2019.

[3] and European Union Agency for Cybersecurity, I. Alamillo, S. Mouille, A. Röck, N. Soumelidis, M. Tabor, and S. Gorniak, *Digital identity standards – Analysis of standardisation requirements in support of cybersecurity policy – July 2023*. European Union Agency for Cybersecurity, 2023.

[4] R. Czerny, C. P. Kollmann, B. Podgorelec, B. Prünster, and T. Zefferer, "Smoothing the ride: Providing a seamless upgrade path from established cross-border eid workflows towards eid wallet systems," in *20th International Conference on Security and Cryptography: SECRYPT 2023*. SciTePress, 2023, pp. 460–468.

[5] European Commission: eIDAS Regulation, "eidas 910/2014 regulation," https://digital-strategy.ec.europa.eu/en/policies/eidas-regulation, Accessed 2024-03-22.

[6] European Commission eIDAS Regulation, "eidas 2 regulation 2024/1183," https://eur-lex.europa.eu/eli/reg/2024/1183/oj, Accessed 2024-04-25.

[7] T. ETSI, "119 432: Electronic signatures and infrastructures (esi)," *Protocols for remote digital signature creation*, 2019.

[8] J. Dumortier, "The european directive 1999/93/ec on a community framework for electronic signatures," *Edirectives: guide to European Union law on ecommerce: commentary on the directives on distance selling, electronic signatures, electronic commerce, copyright in the information society, and data protection, edited by AR Lodder & HWK Kaspersen The Hague*, 2002.

[9] "Organization for the advancement of structured information standards(oasis), "digital signature service core protocols, elements, and bindings version 2.0"," https://groups.oasis-open.org/higherlogic/ws/public/document?document_id=64707, Accessed on 2024-01-22.

[10] "Open mobile api specification v3.0," https://globalplatform.org/wp-content/uploads/2018/04/GPD_Open_Mobile_API_Spec_v3.2.0.13_PublicReview.pdf, Accessed on 2024-04-30.

[11] "Cloud signature consortium (csc) api v2.0," https://cloudsignatureconsortium.org/wp-content/uploads/2023/04/csc-api-v2.0.0.2.pdf, Accessed on 2024-05-04.

[12] "European digital identity wallet architecture reference framework v1.2," https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/releases/tag/v1.2.0, Accessed on 2024-02-04.

[13] "European digital identity wallet architecture reference framework v1.3," https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/releases/tag/v1.3.0, Accessed on 2024-03-04.

[14] Elaine Barker (NIST), Miles Smid (Orion Security MUSAPs), Dennis Branstad, Santosh Chokhani (Cygnacom MUSAPs), "A framework for designing cryptographic key management systems," https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-130.pdf, Accessed 2024-02-14.

[15] "Musap project," https://github.com/NGI-TRUSTCHAIN/MUSAP_project, Accessed on 2024-05-10.

[16] "Client-secret mode for did registration standard being developed by dif," https://identity.foundation/did-registration/client-managed-secret-mode, Accessed on 2024-03-14.

[17] "Musap demo video with multiple keystores," https://www.youtube.com/watch?v=IHl4WDTJY34, Accessed on 2024-05-15.