D7.2

Definition of device system architecture and derivation scheme of mobile IDs

Document Identification	
Date	08.06.2017
Status	Final
Version	1.0

Related WP	WP 2, WP7	**Related Deliverable(s)**	D7.1
Lead Authors	F.-M. Kamm (G&D)	**Dissemination Level**	PU
Lead Participants	G&D	**Contributors**	Atos, Tubitak, TUG
Reviewers	FHG, EEMA		

This document is issued within the frame and for the purpose of the LIGHT^{est} project. LIGHT^{est} has received funding from the European Union’s Horizon 2020 research and innovation programme under G.A. No 700321.

This document and its content are the property of the *Lightest* Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the *Lightest* Consortium or the Partners detriment and are not to be disclosed externally without prior written consent from the *Lightest* Partners.

Each *Lightest* Partner may use this document in conformity with the *Lightest* Consortium Grant Agreement provisions.

1. Executive Summary

It is the goal of this deliverable to define a practical definition of a mobile ID scheme that can be embedded into the LIGHTest infrastructure and which leverages the advantages and features that LIGHTest offers. The objective of this deliverable is to develop further the general requirements of D 7.1 into a concrete proposal for a mobile ID scheme, based on existing elements and standards wherever possible. The FIDO protocol will be motivated as the primary choice for the LIGHTest use case.

To further support a practical implementation, the scheme definition has to take into account the existing reality of the mobile device landscape, including the large variety of different security environments present on mobile devices. It is therefore essential to understand the variety of security environments, their security properties and the ability to address them all. Based on these challenges a reference architecture will be laid out, both on a generic level as well as on a more concrete level taking into account the FIDO architecture.

When considering the integration of FIDO into a mobile ID scheme, two basic variants have to be taken into account. In the direct case the relying party supports the FIDO protocol and uses FIDO in a direct relationship to the end user. Since FIDO is designed as a user-centric model in which the user triggers the registration as well as the authentication, this model would reflect the standard use case of FIDO. In the other case the relying party uses a federation protocol like SAML or OpenID Connect and redirects the user to an ID provider who then performs the authentication. This deliverable focusses on the direct case and its integration into the mobile ID use case. Since even in the direct use case some elements of attribute assertion are required, a concrete SAML profile is proposed and potential further refinements outlined.

2. Document Information

2.1 Contributors

Name	Partner
Frank-Michael Kamm, Andreas Chalupar, Volker Stöhr, Ulrich Martini	G&D
Lorenzo Rosa, Miryam Villegas Jimenez	Atos
Edona Fasllija, Elif Ustundag Soykan, Muhammet Yildiz, Çağatay Karabat	TUBITAK
Peter Lipp	TUG

2.2 History

Version	Date	Author	Changes
0.1	03.05.2017	F.-M. Kamm	Initial Version
0.2	11.05.2017	F.-M. Kamm	First internal draft
0.3	29.05.2017	F.-M. Kamm	Integration of Atos and Tubitak contributions
0.4	30.05.2017	F.-M. Kamm	Update 6.2
0.5	01.06.2017	F.-M. Kamm	1st internal review version
1.0	08.06.2017	F.-M. Kamm	Integration of internal reviewer comments

3. Table of Contents

1. Executive Summary 2

2. Document Information 3
2.1 Contributors 3
2.2 History 3

3. Table of Contents 4
3.1 Table of Figures 5
3.2 Table of Acronyms 5

4. Introduction/Motivation 6
4.1 Mobile ID Requirements 6
4.2 FIDO Authentication Scheme 8
4.3 Challenges 13

5. Mobile Security Environments System Architecture 15
5.1 Security Environments in mobile Devices 15
5.1.1 Secure Element: 15
5.1.2 TEE 17
5.1.3 Software Security 17
5.2 Reference System Architecture for Credential Storage 18

6. FIDO Integration into Mobile ID Scheme 21
6.1 Federation vs. Direct FIDO 21
6.2 Registration/Identification Phase 23
6.3 Usage Phase 26
6.4 Organisational Policies 27
6.5 Choosing a Federation Protocol 27
6.6 SAML profile for LIGHTest 28

7. Summary/Conclusions 31

8. References 32

9. Project Description 34

3.1 Table of Figures

Figure 1: High-level architecture of the FIDO UAF scheme on the client and server side. From [5]. 9
Figure 2: High-level flow of registration step in the FIDO UAF protocol. From [5]. 10
Figure 3: Generic system architecture of security environments used for credential storage in mobile ID applications. 19
Figure 4: Mapping of generic system architecture onto the FIDO UAF architecture. 19
Figure 5: System architecture simplification using the OpnMobileAPI as universal transport layer. 20
Figure 6: Two basic variants of FIDO integration into a mobile ID scheme. The federation case is shown on the left, the direct FIDO case on the right. 21
Figure 7: Sequence diagram of the identity binding flow using a previously registered FIDO authenticator. 24
Figure 8: FIDO UAF and FIM protocols (from [7]). 28
Figure 9: Basic building blocks of the SAML 2.0 architecture [9]. 29

3.2 Table of Acronyms

ASiC	Associated Signature Container
eID	Electronic Identity
eSE	Embedded Secure Element
EU	European Union
FIDO	Fast Identity Online
GSMA	Global System for Mobile Communications Association
ID	Identity
IETF	Internet Engineering Task Force
IoT	Internet of Things
LoA	Level of Assurance
NFC	Nearfield Communication
OTA	Over the Air
OTI	Over the Internet
PKI	Public Key Infrastructure
SAML	Secure Assertion Markup Language
SE	Secure Element
SIM	Subscriber Identity Module
SSO	Single Sign On
SW	Software
TEE	Trusted Execution Environment
TPM	Trusted Platform Module
UICC	Universal Integrated Circuit Card

4. Introduction/Motivation

4.1 Mobile ID Requirements

One of the major use cases of the LIGHTest infrastructure can be found in the context of mobile IDs and mobile authentication. Since identity applications require to build up a chain of trust from the initial primary identity over the derived credentials to the relying party actually consuming the ID, the LIGHTest concept is ideally suited to provide and publish this trust information.

It is therefore the goal of this deliverable to define a practical definition of a mobile ID scheme that can be embedded into the LIGHTest infrastructure and which leverages the advantages and features that LIGHTest offers. Deliverable D 7.1 has defined the requirements for a mobile ID scheme on a more generic level. The objective of this deliverable is to develop further the general requirements into a concrete proposal for a mobile ID scheme, based on existing elements and standards wherever possible.

To further support a practical implementation, the scheme definition has to take into account the existing reality of the mobile device landscape, including the large variety of different security environments present on mobile devices. The storage of derived credentials or other authentication credentials related to the derived ID on the device will be influenced by this variety and will also impact the achievable trust level. It is therefore essential to have some kind of attestation scheme that allows the relying party to judge and understand the trust and security level of the mobile security environment.

While the generic requirements for mobile IDs can be found in more detail in D7.1, the most important ones shall be summarized here again to prepare the rationale of the following considerations:

ID Derivation Process:

- Functionalities to create, delete, revoke, renew, activate, and deactivate the mobile ID shall be provided,
- The secure storage of credentials depending on the desired LoA has to be provided,
- Local and remote provisioning of credentials shall be possible. In case of remote provisioning, end-to-end security has to be established,
- Proof-of-possession of derived credentials shall be possible.

Credential storage:

- The security level of the storage environment shall be well-known,

- An attestation mechanism/key shall be present to make an attestation of the type of authenticator/security environment,
- Secure key generation within the storage environment shall be possible. In cases where this cannot be achieved, end-to-end secured key provisioning shall be doable.
- A transfer of secret credentials shall not be possible to environments resulting in a lower Level of Assurance (LoA). If SW-based storage environments are used, no cloning/extraction of secret key material shall be possible.

Device attestation:

- A software- or hardware-level attestation scheme shall be present,
- A private key to sign attestation data is required,
- A mechanism for device-level revocation should be available.

Level of Assurance (LoA) Propagation:

- The secondary ID issuer scheme must include the LoA of the derivation process,
- A reference to the authenticator and attestation key used during ID derivation shall be included,
- Open standard protocols and data formats shall be used wherever possible,
- The user shall be informed when the LoA is reduced during propagation,
- If not otherwise specified, a combination of derivation, storage, and attestation trust elements should determine the LoA.

A practical definition and implementation of the scheme should of course try to realize as many of these requirements as possible. But beyond the technical and organisational requirements some more aspects have to be considered as well in order to achieve any practical relevance:

- **Open access to the scheme**: preferably, the mobile ID realisation shall be as open as possible regarding the type of authenticator and credential storage environment being used. This allows for large acceptance, optimum tailoring to the needs of the specific use case (not all use cases will require the highest LoA), future innovation, and easier realisation of widespread and suitable business cases by very heterogeneous players. If the scheme were bound to a specific security token (e.g. the SIM/UICC) this would restrict the realisation of a viable business case significantly for all parties not having access to the specific token. It could also drastically reduce the range of suitable mobile devices having all necessary technical requirements.

- **Simple integration on relying party side**: A major aspect of practical acceptance will be the effort a relying party has to undergo to integrate the mobile ID scheme into their services. If the scheme is built upon known, proven and existing standards with a simple interface definition the adoption rate will be most likely higher. The integration costs always have to be balanced with economic gains the relying party can achieve when using this solution.
- **Avoiding the classical chicken-and-egg problem**: if technical requirements and reachable device range create too high adoption hurdles on several sides, the classical chicken-and-egg problem arises. For example, if end users have to invest into a certain token/device technology for using the scheme but would not take the invest unless a certain number of web services is available, while at the same time the web services wait until a certain user base is reachable, the scheme will never succeed. Several PKI and eID schemes have encountered exactly this problem. This is also related to the openness of the scheme (see above) since more flexibility will also increase the user base more quickly.
- **Broad cross-industry support:** mobile ID use cases will span across several application domains, from financial use cases to e-government, enterprise use cases, and presumably automotive and industrial use case. It is therefore desirable to rely upon standards that see a broad cross-industry adoption.

Taking the technical requirements into account as well as the considerations presented above, the FIDO Alliance concept and protocol seems to be a natural choice addressing most of the needs and requirements. The following section will provide a short overview of FIDO with special attention on the aspects above and a reasoning for the protocol choice. Chapter 6 will then discuss further how to practically use FIDO for a mobile ID scheme.

4.2 FIDO Authentication Scheme

The Fast Identity Online (FIDO) alliance is an industry specification group that was founded a few years ago and has now grown to more than 250 member companies. The goal of the alliance is to define an interoperable specification for mobile authentication and to overcome existing fragmentation and silos. More than 330 products have already been certified according to the FIDO specifications and several large roll-outs have been done, involving multi-million numbers of end users.

Technically, FIDO concentrates on authentication and explicitly excludes identity and ID federation. It can however be embedded into identity schemes and combined with ID federation, although not directly supported by the FIDO protocol. Steps like the initial identification and the identity or attribute attestation are not part of the FIDO protocol but can be performed on top of the FIDO authentication step (i.e. prior or afterwards).

FIDO comes with two flavours of the protocol, the U2F-protocol for two-factor authentication and the UAF-protocol for password-less authentication (e.g. using mobile device biometrics) and transaction signing. Both protocols are summarized under the FIDO 1.x specifications and will be unified in the future in the upcoming FIDO 2.0 specification. The approach of FIDO is to balance security on the one hand with usability on the other hand and to overcome existing technology silos by creating an open landscape for authenticators.

The principle of FIDO is based on simple challenge-response protocols using asymmetric keys. In contrast to previous PKI-based systems FIDO wants to explicitly reduce complexity by restricting PKI to the absolute minimum. As a consequence, the user-centric registration triggers the generation of the FIDO key pair and exports the public key to the service provider while the private key is kept on the user side. No further PKI is used in the authentication step. However, especially in the context of identity applications it would be possible to provide further PKI (e.g. a certificate over the public FIDO key) on top of the standard FIDO concept.

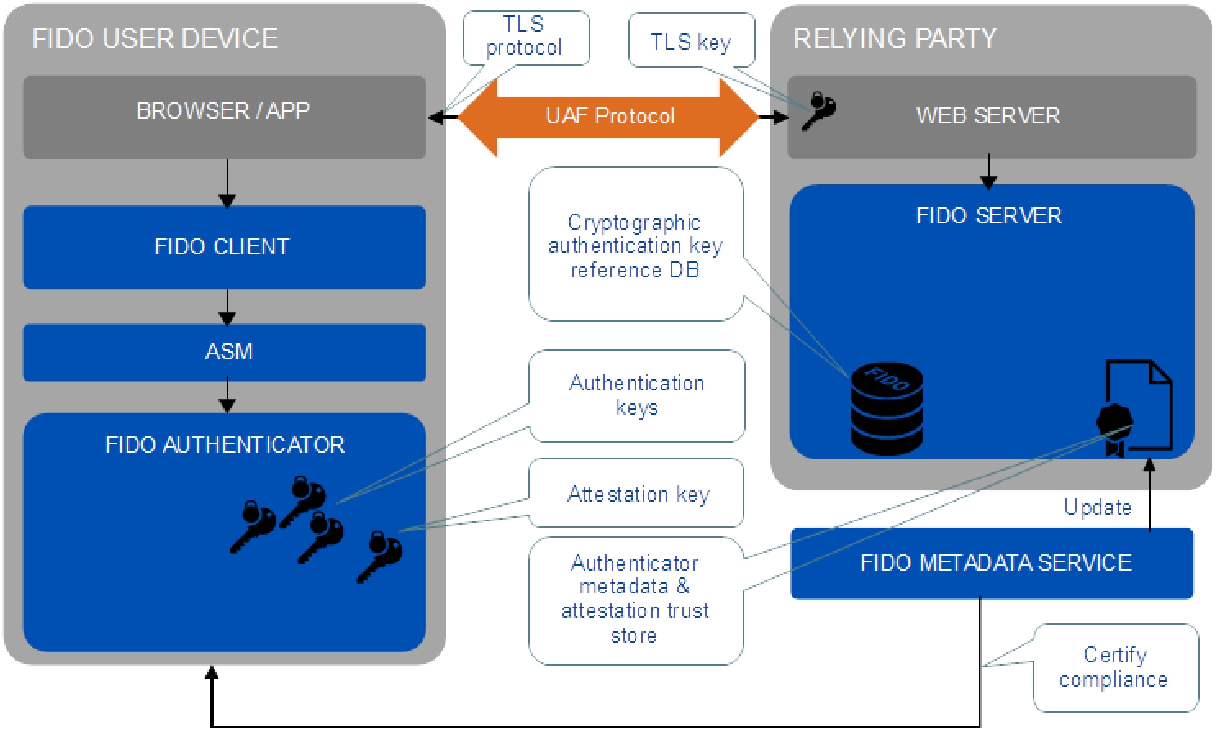
FIDO USER DEVICE
BROWSER / APP
FIDO CLIENT
ASM
FIDO AUTHENTICATOR
TLS protocol
TLS key
UAF Protocol
Cryptographic authentication key reference DB
Authentication keys
Attestation key
Authenticator metadata & attestation trust store
RELYING PARTY
WEB SERVER
FIDO SERVER
FIDO
Update
FIDO METADATA SERVICE
Certify compliance

Figure 1: High-level architecture of the FIDO UAF scheme on the client and server side. From [7].

The high-level architecture in case of FIDO UAF is shown in Figure 1. With the introduction of the Authenticator Specific Module (ASM) as abstraction layer on the client side, basically any suitable authenticator can be integrated if an according ASM is provided. The authenticator contains the private authentication key as well as the private attestation key. The public authentication key is stored on the server side while the attestation certificate can be found in the FIDO metadata service database.

According to the specifications, a lightweight PKI is only used for device attestation where the authenticator proves its integrity with a self-signed certificate of the authenticator manufacturer. The certificate is published in a metadata database and can be used by the FIDO server during the registration step to verify the type of authenticator and to enforce certain policies on the type of acceptable authenticators. The requirement for an attestation scheme is the consequence of the openness of the FIDO authenticator landscape. It adds more complexity but also allows for more flexibility on the usage of different authenticator types. In principle, every authenticator that complies with the FIDO protocol specifications can be used on the client side.

Therefore, FIDO needs to deal with a large variety of authenticators with significantly different security levels. The range can include pure software implementations as well as TEE-based authenticators or hardware-supported devices (smart cards, µSD cards, USB tokens…). In order to enforce certain security policies the relying party needs to know which type of authenticator is available and how trustworthy this can be. With the attestation certificates the relying party can restrict the range to only known authenticators.

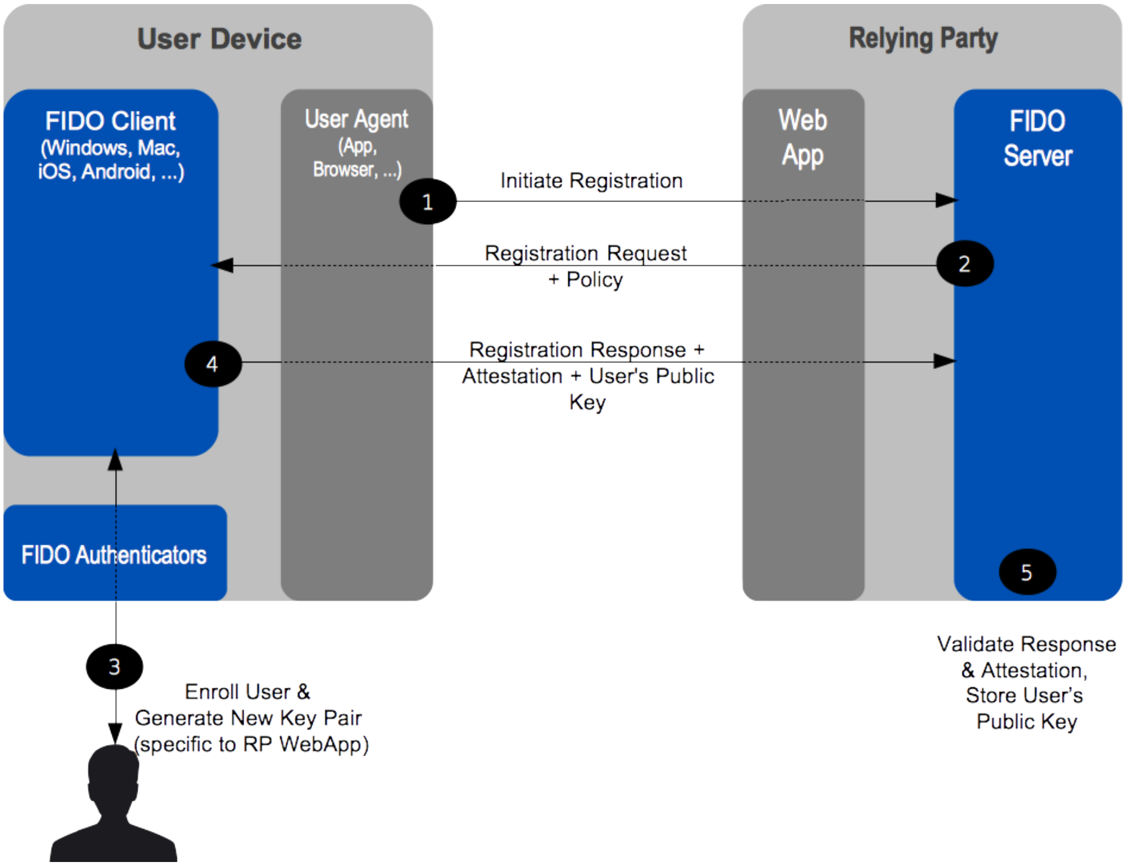
User Device
FIDO Client (Windows, Mac, iOS, Android, ...)
User Agent (App, Browser, ...)
FIDO Authenticators
Relying Party
Web App
FIDO Server
1
Initiate Registration
2
Registration Request + Policy
4
Registration Response + Attestation + User's Public Key
3
Enroll User & Generate New Key Pair (specific to RP WebApp)
5
Validate Response & Attestation, Store User's Public Key

Figure 2: High-level flow of registration step in the FIDO UAF protocol. From [7].

Figure 2 shows a high-level flow of the FIDO UAF registration step. After the user has initiated the registration the server sends the policy containing a list of acceptable authenticators. In the context of ID applications this step could be used to guarantee a certain minimum LoA by blocking all types of authenticators that would not reach this LoA. After user enrolment and credential generation the response is sent containing a signature with the attestation key. Attestation only takes place during the registration step, not during authentication.

In summary, FIDO addresses all of the non-technical requirements discussed above. With its open authenticator architecture it is not restricted to a certain security token or type of authenticator. With its simple server API it is easy to integrate for the relying party. In addition, it can be embedded into federation protocols if this is the preference of the relying party (see section 6.1). It has gained large cross-industry support and with the ability to address many types of devices and authenticators it also reduces the likelihood of a chicken-and-egg problem, although it also has to overcome the initial acceptance hurdles.

Regarding the technical requirements, the following mapping table based on the technical and authenticator related requirements of D 7.1 shall give an overview of the suitability of the FIDO protocol:

Table 1: Mapping of requirements from D7.1 related to the authenticator with capabilities of the FIDO scheme.

Identifier	Description	With FIDO?
R_MID_IDev_1	The Secondary ID Derivation Service MUST provide functionalities for the Creation/ Deletion/ Revocation of the derived ID credentials.	✓ (Revocation outside the FIDO scheme, e.g. via certificate validity).
R_MID_IDev_2	The ID Derivation Service of the Secondary ID Issuer MUST provide functionalities for the Activation/ Deactivation of the derived ID credentials.	(✓) (Indirectly via validity or revocation).
R_MID_IDev_6	The Secondary ID Derivation Service MUST provide means for the renewal of the validity period of the derived ID credential.	(✓) (Indirectly via validity or revocation).
R_MID_IDev_8	The derived ID credentials MUST be valid only for a limited period of time.	(✓) (Indirectly via certificate validity).
R_MID_IDev_9	The derived ID credentials SHOULD be suitable to be used as replacement of the primary eID credentials for identification, authentication and authorization steps depending on the LoA of the primary eID credential.	✓ (Identification in conjunction with certificates or federation attestation).
R_MID_IDev_10	The derived ID credentials MUST be stored securely on the mobile device.	✓ (Addressed by attestation

Identifier	Description	With FIDO?
	The level of security may depend on LoA requirements set by the corresponding trust scheme.	scheme).
R_MID_IDev_14	The Secondary ID Derivation Service MUST synchronize the lifecycle and status of the derived ID credential with the primary ID credential.	(✓) (Indirectly via validity or revocation).
R_MID_IDev_15	The Secondary ID Derivation Service SHOULD allow for both remote and local derived ID credential provisioning schemes.	✓ (External attestation and public key import possible. Provisioning service outside FIDO).
R_MID_IDev_17	A proof of possession of the derived ID credential by the owner MUST be ensured.	✓ (See chapter 6.2).
R_MID_IDev_18	The Secondary ID Derivation Service SHOULD be able to choose among different available options of security environments on mobile devices, based on the required security level.	✓ (FIDO middleware architecture allows for integration of several environments.).
R_MID_CrSt_1	The mobile device MUST have at least one storage environment with well-known security properties that is part of a device attestation scheme.	✓ (Attestation scheme independent of type of security environment).
R_MID_CRSt_2	The storage environment MUST contain at least one cryptographic key that can be used for the attestation of the type of security environment.	✓ (Addressed by attestation scheme).
R_MID_CRSt_4	If a mobile device has several credential storage environments, each of these environments MUST be integrated into the same attestation scheme.	✓ (Addressed by attestation scheme).
R_MID_DevAt_3	The private key of the device MUST be used to sign data during the attestation process.	✓ (Addressed by attestation scheme).
R_MID_DevAt_4	The private key of the device SHOULD be unique.	✓ (Addressed by DAA attestation option).
R_MID_DevAt_5	The attestation scheme used SHOULD provide privacy, e.g. by using a Privacy CA or a scheme like ECDAA	✓ (Addressed by DAA attestation option).
R_MID_DevAt_6	The device MUST provide a mechanism for device level revocation.	✓ (Via metadata database).
R_MID_DevAt_8	A policy on how old versions are handled SHOULD be provided.	✓ (Via metadata database. Policy

Identifier	Description	With FIDO?
		definition outside FIDO scope).
R_MID_TrPr_1	The derived ID credentials MUST contain the LoA that can be reached by the derived ID within the trust scheme of the Secondary ID Issuer.	✓ (In conjunction with certificates or federation attestation).
R_MID_TrPr_2	The derived ID credentials SHOULD contain the LoA of the primary ID that was achieved in the primary ID issuer trust scheme.	✓ (In conjunction with certificates or federation attestation).
R_MID_TrPr_3	The derived ID credentials SHOULD contain a reference to the authenticator used during ID derivation and to the attestation key of the authenticator.	✓ (In conjunction with certificates or federation attestation).
R_MID_TrPr_4	The LoA propagation to the relying party MUST employ open standard protocols and data formats to ensure interoperability.	✓ (FIDO as openly available specification).

Requirements not listed in the table above are not related to the authenticator part and are thus not directly related to the FIDO scheme. They can be realized outside or on top of the FIDO protocol.

4.3 Challenges

While FIDO addresses most of the requirements for a mobile ID scheme and allows for large flexibility regarding the choice of authenticators there are also some remaining challenges when trying to embed FIDO into an identity scheme:

- **Scoped credentials**: Since FIDO credentials are scoped, i.e. bound to a specific relying party, each new FIDO registration will result in a new key pair. While this supports privacy it also contradicts the idea of a universally usable mobile ID. The challenge will therefore be to find an appropriate binding mechanism between the identification and the FIDO credentials taking into account the nature of the scoped credentials. In the worst case this would result in asking the user to perform an identification session each time a new FIDO key pair is generated.
- **User centricity**: FIDO is a user centric model in which the registration and authentication are triggered by the user. This also means that the key pair generation only takes place after the user has triggered the registration step (see Figure 2). In certain use cases this may be circumvented by an external key pair generation followed by a provisioning step (e.g. during production and personalisation of a smart card when using the card as authenticator) but this would already be beyond the FIDO standard protocol. As a

consequence it has to be considered carefully when and how to integrate the registration step with the identification step of the secondary ID issuer without breaking the FIDO flow. Specifically the question will be whether a FIDO key pair scoped to the relying party is already available when the user performs the ID derivation step with the secondary ID provider.

- **Unlimited validity**: As already indicated in the requirements mapping table (Table 1) the validity of the FIDO credentials is basically not limited unless the user performs a de-registration (user centricity). Thus, life cycle synchronizations as well as revocations have to be solved outside/beyond the FIDO scope, e.g. via certificate validity.

The challenges listed above will have to be taken into account when designing a FIDO based mobile ID scheme and will be part of the concept outlined in chapter 6.2 and 6.3.

5. Mobile Security Environments System Architecture

5.1 Security Environments in mobile Devices

One of the major challenges when dealing with mobile device based security solutions is the large diversity of device types and security environments that they provide. Every solution that requires a specific security environment (like an embedded Secure Element or a Trusted Execution Environment) automatically limits the device range significantly. It is therefore essential to understand the variety of security environments, their security properties and the ability to address them all (e.g. via an abstraction layer). The following overview summarizes the main properties of the most common mobile security environments.

5.1.1 Secure Element:

Hardware based secure elements are available in various form factors (see below) and provide the highest possible security level of all security environments. They offer a large set of security and trust properties, especially:

- **Tamper-proof hardware**: The security IC is typically tamper-proof and provides a larger set of functions to prevent or at least detect hardware attacks.
- **Side channel leakage resistance**: The combination of hardware and operating system is optimized with respect to reduced side channel leakage. Via side channel leakage secret information could be obtained without actually attacking the direct channel. Side channel attacks can be timing attacks, power analysis attacks or fault injection attacks.
- **Secure Storage**: The combination of tamper-resistant hardware and access control allows for secure storage of secret information like cryptographic keys.
- **Secure Authentication**: With cryptographic secrets and appropriate authentication protocols, the card allows for secure authentication. A successful authentication can be the requirement for accessing certain data or performing certain functions on the card.
- **System Integrity Verification**: While booting, the secure elements can verify the integrity of their firmware by calculating a cryptographic checksum over the code.
- **PIN/PW Protection**: A PIN or password can be requested to protect information on the card. A PIN retry counter allows to limit the number of allowed false attempts for a PIN or password entry.
- **Secure Messaging**: Allows to protect the communication between card and interface device with respect to integrity and/or confidentiality.
- **Cryptographic Services**: A cryptographic key pair can be generated on the card, while the private key never leaves the card environment.

The following Secure Element form factors are the most important ones for mobile devices. They all share the same fundamental security properties but have some differences regarding their interface and/or their functionality:

SIM/UICC:

The Subscriber Identity Module (SIM) is the most frequently used secure element in mobile devices and is mainly used for authentication to the mobile network. Over the recent years, this card has evolved to a UICC (universal IC card) which can host several applications in parallel. In this case the SIM functionality is one of several applications on the card. New applications can be loaded onto the UICC via the internet (“over-the-internet”, OTI) or via the cellular network (“over-the-air”, OTA) allowing the support of additional applications after field rollout. Appropriate card management functions have been defined to allow a secure managing of card applications and to avoid interference or information leakage between the applications. For mobile devices supporting Nearfield Communication (NFC), the NFC interface can securely connect directly to the UICC via the Single Wire Protocol (SWP). Thus, NFC-based applications, like mobile payment can rely on the trust and security functionality of a secure element.

Embedded Secure Element:

While the SIM/UICC is owned by the network operator and is interchangeable, the same functionality can be located on an embedded secure IC which is owned by the device manufacturer. Since the element is not removable, all applications and credentials stored on it need to be manageable from remote. An embedded secure element can support additional security applications independent of the mobile network operator.

Secure microSD Card:

The secure microSD card is a smartcard chip combined with a flash memory chip inside a conventional microSD memory card. As microSD cards are removable elements, they are neither bound to the mobile network operator nor to the mobile phone manufacturer, and can thus be controlled and used by an independent third party. Generally these cards are available on any phone with a microSD card slot. On many Android devices, secure microSD cards can be accessed in the same way as SIM cards and embedded secure elements using the Open Mobile API.

Contactless NFC Smartcards:

As the NFC interface is compatible to ISO 14443, it can provide access to external contactless smartcards. This allows using for example eID cards as secure elements in a convenient way, by tapping them to the phone and without requiring a dedicated smartcard terminal.

5.1.2 TEE

The Trusted Execution Environment (TEE) for mobile devices is an execution environment completely separated from the normal ("rich") operating system. Mechanisms are implemented to protect the integrity of the TEE software itself with a secure boot process and the integrity of applications running within the TEE context. The separation of the TEE and the rich OS is supported by the mobile device processor architecture (e.g. ARM TrustZone or Intel TXT). The processor provides a dedicated hardware section to run trusted applications and to establish a trusted environment. Additionally, the access to shared resources (e.g. memory) is controlled and strictly separated.

In contrast to a Secure Element, the TEE does not provide hardware security in terms of tamper resistance against hardware attacks and side channel leakage. In the current form, it also does not have access to dedicated hardware resources - like a dedicated memory – other than the separate processor execution environment. Nevertheless, the TEE can store information in the main memory (or other storage resources) in encrypted form and can control the encryption and decryption by a Trusted Application. Therefore, this information is never accessible in plain text for applications running in the rich OS.

5.1.3 Software Security

All the security environments listed above are only available on a subset of mobile devices and in some cases even restricted to a specific mobile operating system. For example, currently dual interface cards cannot be accessed by iOS phones due to the restrictions of the NFC interface. TEEs are about to penetrate the market further but are still available only on a limited number of device types. Other secure elements may be available but not accessible since they are managed by an external party. This is especially the case for the SIM card or UICC. As a consequence, pure software-based solutions are a typical fall-back option since they can be used on basically any device type.

On the other hand, software environments do not provide the same security and protection level as hardware Secure Elements since they are missing most of their security properties as listed above. Especially side channel resistance and resistance to cloning/extraction of credentials are typical issues. For this reason, software-secured environments require additional security measures compared to plain software implementations, typically comprising:

- White-Box Cryptography (WBC): this is a method to hide secret/private key material in code and random matrices so that an attacker cannot distinguish between random data and actual key material.
- Root detection: checks if a device is rooted or contains a custom OS with unknown security properties or with root level permissions that could circumvent OS security policies.
- Device fingerprinting: collects characteristic device data to allow for a binding of WBC credentials to a specific device.
- Obfuscation: by obfuscating the code, re-engineering is made more difficult, though not impossible.

As all these environments listed above provide a different security level they will typically result in a different LoA of the derived ID once they are used for storage of authentication credentials. Therefore it will be the challenge to deal with these different properties and to make them transparent to the relying party consuming the mobile ID. Fortunately, the FIDO scheme offers an excellent opportunity to address this challenge with the attestation scheme, allowing to obtain more information about the authenticator and its type.

5.2 Reference System Architecture for Credential Storage

Based on the considerations above a reference architecture shall be laid out here, both on a generic level as well as on a more concrete level based on FIDO. Three aspects have to be taken into account, resulting from the different properties of the security environments:

- **Data format/API**: the security environments differ significantly in terms of API definition and data format use. While the hardware based Secure Elements typically rely upon ISO/IEC 7816 interfaces and APDUs [1], the TEE is rather based on the Global Platform API [2] and software-based libraries are basically unspecified regarding APIs and data formats.
- **Transport layer**: another major difference can be the transport layer, depending on whether the secure environment is embedded, or connected via USB, BLE or NFC. Especially removable tokens or smart cards fall into the latter category.
- **Security level**: as described above the security level of the hardware-based, TEE-based and SW-based environments differs and thus results in a different LoA in the identity use case. It is therefore essential to have an indication or better a cryptographic proof for the relying party about which environment was used.

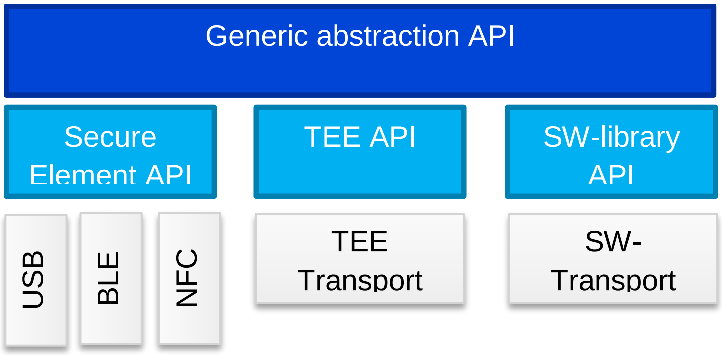
Generic abstraction API
Secure Element API
TEE API
SW-library API
USB
BLE
NFC
TEE Transport
SW-Transport

Figure 3: Generic system architecture of security environments used for credential storage in mobile ID applications.

Thus, on a generic level, the reference architecture will typically look like shown in Figure 3. A generic API would serve as the interface to the mobile application and provide an abstraction of the underlying security environment. For each security environment, a separate layer would be required below to translate into the logic and data format of the specific environment. One or more transport layers would then be needed to address the security environment, especially for those hardware environments that share the same logical interface but are available in different form factors.

As shown in Figure 1 on page 9, the FIDO UAF middleware architecture already offers a structure that can be nicely mapped onto the generic architecture shown above. In the FIDO model, the FIDO client acts as a generic API that provides abstraction to the underlying layers. To connect this layer to the specific authenticator environment, the ASM (Authenticator Specific Module) is introduced, which acts as an interface layer as well as a transport layer.

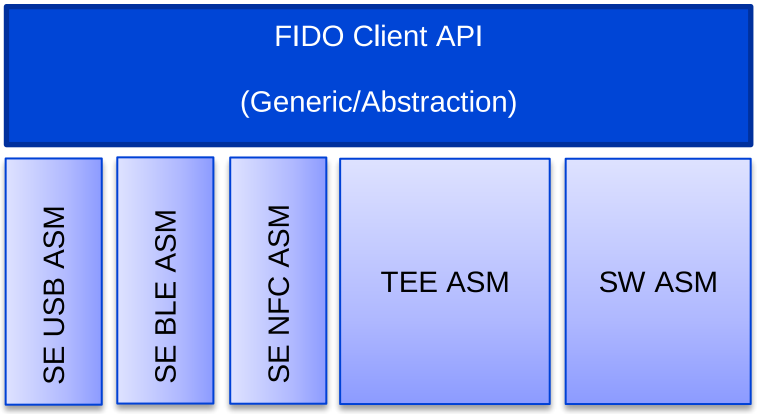
FIDO Client API
(Generic/Abstraction)
SE USB ASM
SE BLE ASM
SE NFC ASM
TEE ASM
SW ASM

Figure 4: Mapping of generic system architecture onto the FIDO UAF architecture.

Thus, mapping the generic architecture of Figure 3 onto the FIDO architecture in Figure 1 results in the architecture shown in Figure 4. According to the FIDO specifications, the client would allow the user to choose an authenticator and thus a security environment in case that several ASMs are present on the device. Since during the registration step (Figure 2) an authenticator attestation is provided with an attestation key embedded into the chosen authenticator security environment, the relying party would also have a cryptographic proof of the type of authenticator/security environment used. Depending on the chosen overall architecture (see section 6.1) it will be a challenge to make this cryptographic proof also available to the secondary ID issuer who performs the ID derivation.

To simplify the architecture in Figure 4 further, the variety of transport layers can be simplified by using the OpenMobile API with add-on terminals for each (non-SIM/UICC) transport channel [3]. This concept was already demonstrated in the FutureID project and is describe further in FutureID deliverable D31.2 - Interface and Module Specification and Documentation [4]. If the OpenMobile API concept is applied to the FIDO architecture, the simplification shown in Figure 5 can be achieved. The OpenMobile API now acts as a common transport layer for each security environment so that the ASM only has to provide the translation of the FIDO flow into the respective language of the security environment.

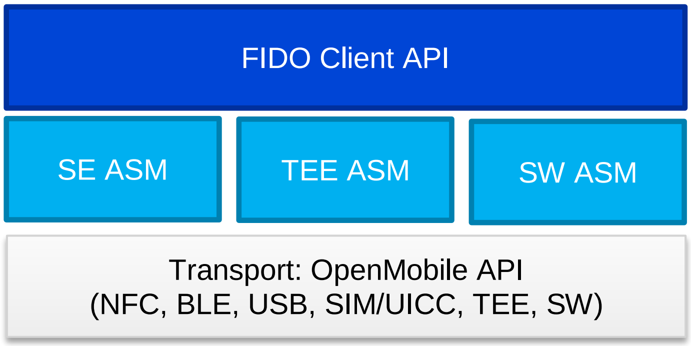

Figure 5: System architecture simplification using the OpnMobileAPI as universal transport layer.

Overall, it is demonstrated here that the FIDO concept and architecture nicely addresses the existing variety of security environments as well as the needs of trust propagation in the envisioned LIGHTest mobile ID use case. The concrete integration of the FIDO concept into the mobile ID concept will be discussed further in the next section.

6. FIDO Integration into Mobile ID Scheme

6.1 Federation vs. Direct FIDO

When considering the integration of FIDO into a mobile ID scheme, two basic variants have to be taken into account. Both schemes are depicted in Figure 6. In the direct case (Figure 6, right) the relying party (i.e. the web service consuming the mobile ID) supports the FIDO protocol and uses FIDO in a direct relationship to the end user. Since FIDO is designed as a user-centric model in which the user triggers the registration as well as the authentication, this model would reflect the standard use case of FIDO.

In the other case (Figure 6, left) the relying party uses a federation protocol like SAML or OpenID Connect and redirects the user to an ID provider who then performs the authentication. The ID provider could use FIDO as a strong authentication protocol and would afterwards issue an assertion to the relying party. As a consequence, the relying party would not directly see the FIDO authentication but trust the attestation of the ID provider regarding the success of the authentication and LoA that has been achieved based on the primary ID and the security level of the authenticator.

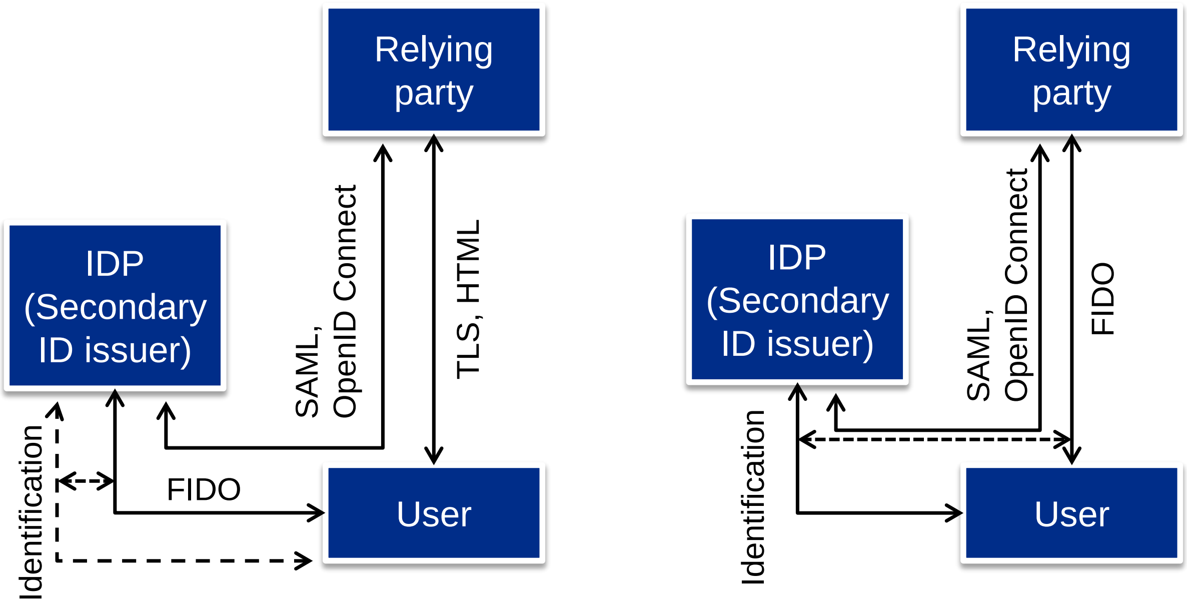
Relying party
IDP (Secondary ID issuer)
SAML, OpenID Connect
TLS, HTML
Identification
FIDO
User
Relying party
IDP (Secondary ID issuer)
SAML, OpenID Connect
FIDO
Identification
User

Figure 6: Two basic variants of FIDO integration into a mobile ID scheme. The federation case is shown on the left, the direct FIDO case on the right.

There are some major differences between both integration schemes with regard to the timing of events and also with regard to the role of the ID provider. For the **federation scheme**, the basic flow would be as follows:

1. The user opens the relying party web site and chooses "Login/Registration with ID provider".
2. The relying party constructs a SAML or OpenID Connect request and redirects the request to the ID provider.
3. If the user visits the ID provider for the first time, an identification session has to be done. This can be for example an eID card identification or a video chat session (VideoIdent) in which the user presents his/her ID cards. Part of this identification session would also be the registration of a FIDO authenticator. The ID provider then performs a binding between the public FIDO key of the user and the identity credentials. This can be done by issuing a certificate over the public key or by including the public key into the SAML or OpenID Connect attestation.
4. If the user visits the ID provider repeatedly the ID provider asks for a FIDO authentication. Based on the public key certificate that was issued during step 3 or based on a user database entry the ID provider confirms that the user has already been identified.
5. The ID provider issues a SAML or OpenID Connect attestation about the user identity or merely the success of the authentication.

As a consequence, the ID provider would be contacted every time the user wants to authenticate to the relying party. If this shall be avoided, it would also be possible to place an authentication service in between the user and the ID provider. This authentication service would only contact the ID provider if an identification of the user is needed. Once this has been done and the identity has been linked to the FIDO credentials the authentication service could operate without the ID provider for pure authentications.

In the **direct FIDO** scheme the flow looks somewhat different:

1. The user contacts the relying party web site and triggers a FIDO registration. During this registration the FIDO key pair is generated.
2. If the relying party needs an identity attestation in addition to the registration it would forward the user to the ID provider and provide the FIDO public key to the ID provider. This can be done similar to the federation scheme using a federation protocol like SAML or OpenID Connect.
3. The ID provider performs the identification and verifies the identity data.
4. The ID provider asks for a proof of possession to verify that the user is holder of the FIDO private key. To do this, the ID provider triggers the relying party to perform a FIDO authentication and verifies it with the public key he has obtained from the relying party.

5. After successful identification and proof of possession the ID provider issues an attestation to the relying party. Alternatively he could also issue a certificate over the FIDO public key.
6. For the following authentications the ID provider is not needed anymore. Now, the relying party and the user follow the standard FIDO approach and perform standard FIDO authentications.

As a consequence, both schemes have their advantages and disadvantages. The federation scheme only requires one protocol interface to the ID provider and thus makes the integration for the relying party easier. As a drawback, the ID provider is involved even for every normal authentication. This could be circumvented by introducing an authentication service in between but this adds complexity to the overall architecture. The direct FIDO scheme requires a somewhat more complex integration on the relying party side since besides the standard FIDO protocol the relying party would also need to support the redirect to the ID provider and the proof of possession. On the other hand, the ID provider would not be needed anymore for a simple authentication.

Since the federation scheme will be predominantly covered by the FutureTrust project (due to the partner structure comprising an already existing ID provider with federation interface), the LIGHTest project will concentrate on the direct FIDO case. The following sections will outline in more detail, how the integration will look like.

6.2 Registration/Identification Phase

The usual FIDO registration flow (as shown in Figure 2) occurs based on the user intention to register on the relying party web site. For the LIGHTest reference flow it is assumed that this registration is done first by the user, following the standard FIDO protocol. This is preferable since the FIDO credentials (public/private key pair) as well as the key ID do not exist before this registration is done. It is therefore assumed, that the actual identity binding occurs after a FIDO user registration has been performed successfully.

Figure 7 shows the sequence diagram of the identity binding flow. In a more detailed description the different steps are the following:

1. The user has performed a successful FIDO registration at the relying party according to the FIDO standard protocol. In case the registration and identification occur at different points in time, the user would start with a FIDO authentication with the previously registered authenticator instead.
2. The relying party requests a challenge from the ID provider through an API that the ID provider publishes. This challenge will later be signed by the FIDO private key as proof-of-possession and can be a session ID of the ID provider or any other unique reference. The challenge is returned to the relying party.

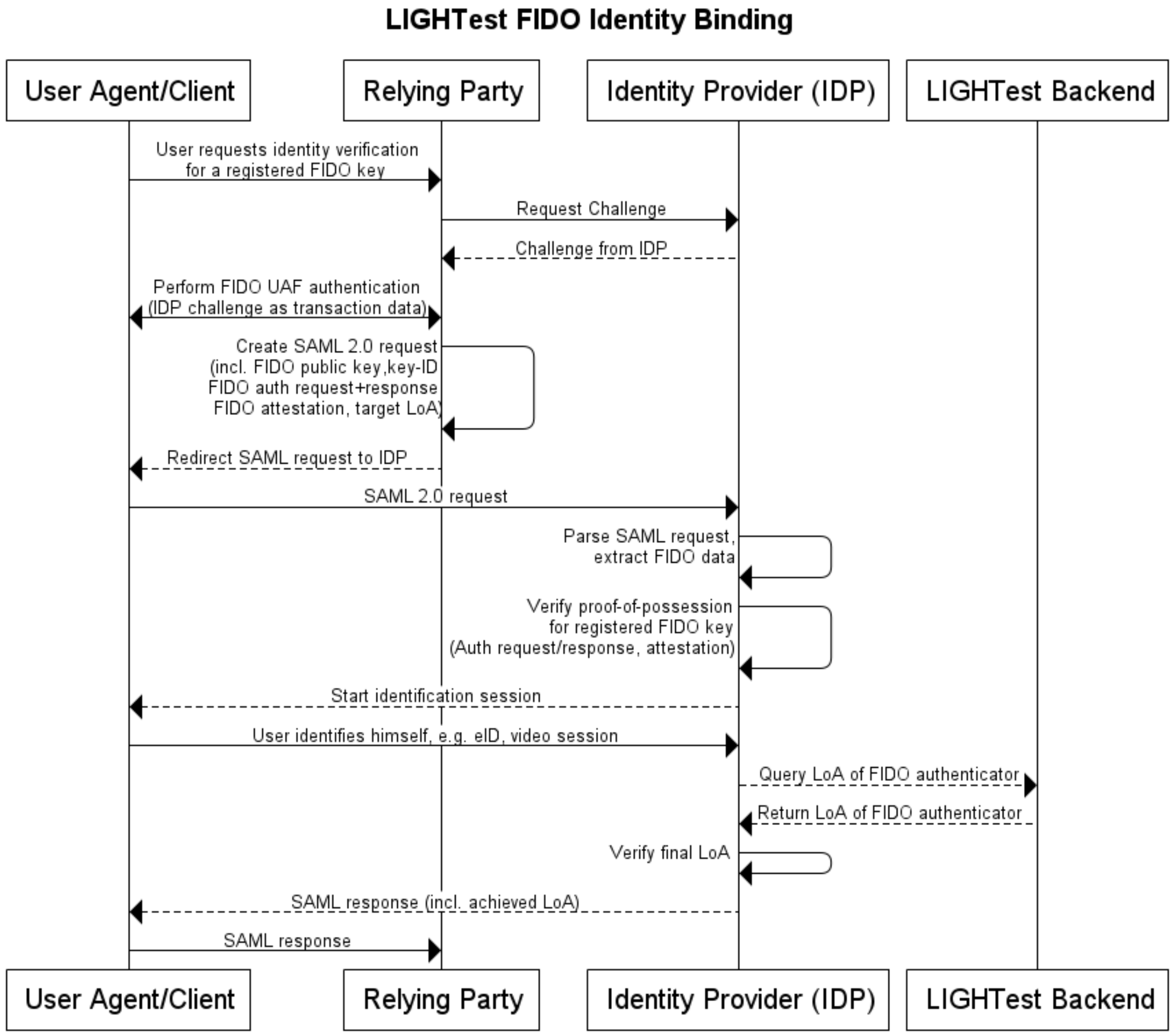
LIGHTest FIDO Identity Binding
User Agent/Client
Relying Party
Identity Provider (IDP)
LIGHTest Backend
User requests identity verification for a registered FIDO key
Request Challenge
Challenge from IDP
Perform FIDO UAF authentication (IDP challenge as transaction data)
Create SAML 2.0 request (incl. FIDO public key,key-ID FIDO auth request+response FIDO attestation, target LoA)
Redirect SAML request to IDP
SAML 2.0 request
Parse SAML request, extract FIDO data
Verify proof-of-possession for registered FIDO key (Auth request/response, attestation)
Start identification session
User identifies himself, e.g. eID, video session
Query LoA of FIDO authenticator
Return LoA of FIDO authenticator
Verify final LoA
SAML response (incl. achieved LoA)
SAML response
User Agent/Client
Relying Party
Identity Provider (IDP)
LIGHTest Backend

Figure 7: Sequence diagram of the identity binding flow using a previously registered FIDO authenticator.

3. The relying party performs a FIDO authentication, using the IDP challenge as transaction data to be signed (FIDO UAF).
4. The relying party constructs a SAML 2.0 (or OpenID Connect) request, containing at least the FIDO public key of the user to which the identity shall be bound, the key-ID, the attestation of the FIDO authenticator (done during registration), the authentication request and response and the target LoA that the relying party needs.
5. With a HTTP redirect the SAML/OpenID Connect request is forwarded to the identity provider (secondary ID issuer).
6. The ID provider extracts the FIDO data and verifies the proof-of-possession, based on the FIDO public key, the FIDO request and the FIDO response.

7. The ID provider requests an identification of the user. The actual mechanism of identification depends on what the ID provider supports and what is required by the target LoA. Typical methods could be an identification using an eID card or a video web session.
8. The user performs the identification according to the requested method. The ID provider verifies the validity of the user identity.
9. The ID provider can facilitate the LIGHTest trust publication infrastructure to verify that the corresponding authenticator is sufficient for the target LoA. The authenticator can be identified based on the FIDO attestation that the relying party has forwarded to the ID provider.
10. The ID provider determines the final LoA based on the provided user ID verification and the user authenticator.
11. The ID provider issues a SAML 2.0 (or OpenID Connect) attestation, containing the requested identity attributes, the FIDO public key and the achieved LoA.
12. The SAML (or OpenID Connect) response is forwarded to the relying party. If the relying party and ID provider agree upon issuing a certificate over the FIDO public key, the relying party would import the certificate into the relying party FIDO server.

The proof-of-possession part of this flow (step 6) requires some further considerations. For a proof-of-possession the ID provider has to obtain the assurance that the user is actually in possession of the corresponding private key to which the public key belongs. For this reason, the ID provider could request an authentication of the user which is then verified with the public key that the ID provider has obtained from the relying party. However, since FIDO credentials are scoped credentials this step is not straightforward.

The FIDO model wants to ensure that no third party can request an authentication with a private key that was originally issued for a specific purpose, i.e. a specific relying party. For this reason, FIDO introduced the AppID and Facet ID concept which provides a binding of the FIDO key pair to a certain origin [5]. The AppID is usually a https-URL of the relying party. If the relying party uses several instances of its service (e.g. a web URL, an Android app, and an iOS app) it can publish a list of trusted facets. In case of apps the facet would be the hash of the apk signing certificate (Android) or the BundleID URI of the app (iOS). The AppID would consist of the URL in which the facet list is published.

FIDO has certain restrictions on the origin of the authentication request with respect to the AppID or Facet ID of the corresponding key pair. For example, the origin has to be in the same DNS domain as the AppID. This restricts the access of third parties like an ID provider who are typically located in another DNS domain. For this reason, the ID provider cannot directly ask for a user authentication but has to make this request to the relying party instead.

In the mobile app model, the flexibility is a little bit higher. In this case, the relying party can add the ID provider app to the trusted facet list. The ID provider app could then directly request a FIDO authentication with the corresponding key pair. The FIDO client would obtain the trusted facet list from a relying party URL and would verify that the ID provider app is eligible to access the key pair.

For the browser model there may be options to work with cross-origin resource sharing concepts however they would be quite complex, make the relying party integration more difficult and would potentially stretch the FIDO security concept close to its limits. Therefore, the easiest is to request a challenge from the ID provider which is later signed as transaction data during the authentication step. This challenge is also a unique reference for the identification session with the user which is performed afterwards. This is the approach proposed above.

6.3 Usage Phase

Once the identity binding has been done as described in the previous section, the usage phase will follow the standard FIDO protocol for authentication. The relying party has basically two options to store the identity data of the user in order to maintain the ID binding:

- **Certificate**: if the ID provider has issued a certificate over the public FIDO key, containing the requested attributes the relying party can import this certificate into the FIDO server database. This has also the advantage that for each authentication the relying party can verify that the FIDO public key is the correct one and has not been exchanged by an attacker. An additional optional security improvement could be a certificate request using a PKCS #10 signed certificate request message over a TLS channel. With this method, the ID provider ensures that the public key is delivered to the claimed relying party.
- **User database**: if no certificate is available, the relying party would simply store the user ID data in the normal identity management database, preferably with a reference to the FIDO key ID. This cross-linking would make it harder for an attacker to exchange public keys on the server.
- **ASiC Container:** If no certification functionality is available on the ID Provider side, the relying party can form an ASiC Container [5] so that it can manage the binding of the FIDO public key and the identity attributes without sacrificing their integrity. "The ASiC is a data container holding a set of file objects and associated digital signatures and/or time assertions using the ZIP [6] format." There are two variants of ASiC Container, an ASiC Simple (ASiC-S) container that associates one single signed file object and an ASiC Extended (ASiC-E) container that associates one or more signed file objects. Within the scope of the LIGHTest scenario, ASiC-S meets the requirements for binding of the FIDO key.

6.4 Organisational Policies

One of the challenges of derived mobile IDs is the lifecycle synchronisation between the primary ID and the secondary (derived) ID. Since the FIDO credentials are scoped with respect to a web service they are in principle not scoped with respect to time. The user-centric FIDO model assumes that the user triggers a de-registration if the user changes the mobile device. If the web service terminates the relationship with the user, then it would delete the user public key instead.

For this reason, the lifetime limitations need to be covered by organisational policies of the ID provider ensuring that a certificate over the public key or a SAML/OpenID Connect attestation contain a lifetime limitation that does not go beyond the lifetime of the original primary ID.

Revocations of the primary ID have to be handled by the secondary ID issuer as well. For this reason, the secondary ID provider shall operate (or co-operate with) an OCSP responder that allows the relying party to verify the current certificate status.

6.5 Choosing a Federation Protocol

As described above, the LIGHTest mobile ID scenario needs more attributes for personalization and authorization than simply the FIDO public key. Since FIDO is an authentication specification, and not an identity standard, we will use some existing federation protocol initiatives that can provide more attributes.

As already outlined, SAML or OpenID Connect are good candidates for this: the authentication based on FIDO takes place first, and then those identity protocols would *leverage* that authentication event into other events between the identity provider and the relying party, see Figure 8.

Especially in the mobile world, OpendID Connect has some advantages compared to SAML due to the JSON based messages (whereas SAML uses XML-based messages). The bigger message size of XML-based message leads to a transmission overhead. SAML has powerful features but can be complex when implementing. On the contrary, OpenID Connect offers an easy way of implementing functionality. However, we recommend SAML, since it is a base protocol used in eIDAS, more precisely for the exchange of messages as laid down in the eIDAS Interoperability Architecture [8].

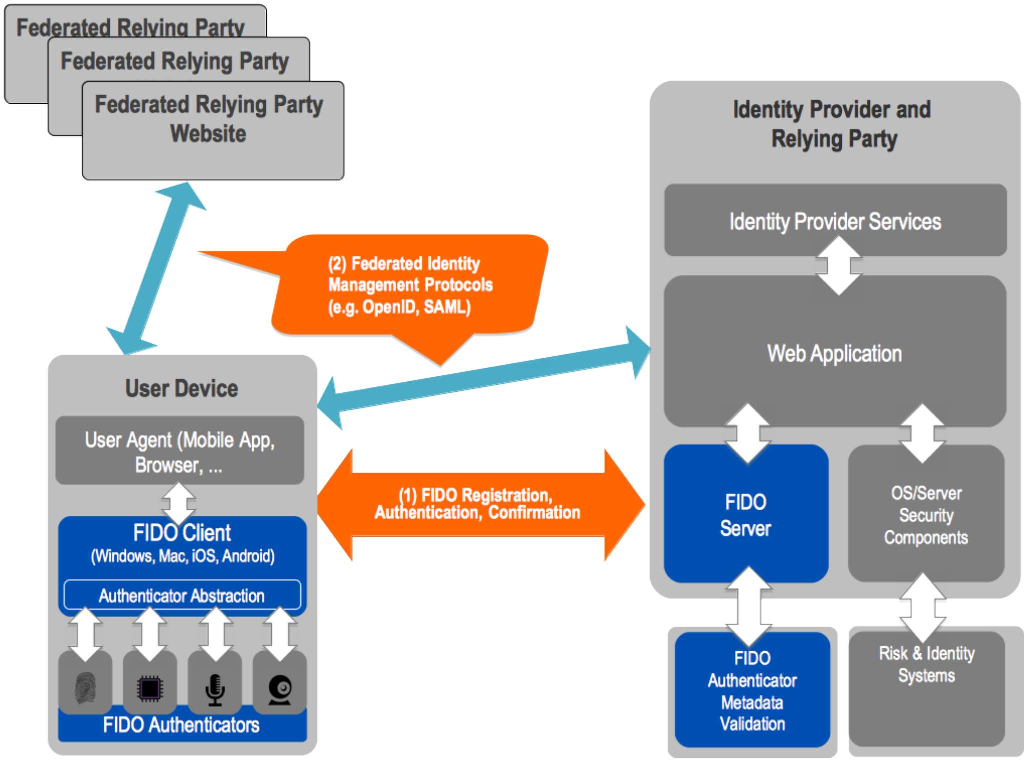
Federated Relying Party
Federated Relying Party
Federated Relying Party Website
(2) Federated Identity Management Protocols (e.g. OpenID, SAML)
Identity Provider and Relying Party
Identity Provider Services
Web Application
User Device
User Agent (Mobile App, Browser, ...
FIDO Client (Windows, Mac, iOS, Android)
Authenticator Abstraction
FIDO Authenticators
(1) FIDO Registration, Authentication, Confirmation
FIDO Server
OS/Server Security Components
FIDO Authenticator Metadata Validation
Risk & Identity Systems

Figure 8: FIDO UAF and FIM protocols (from [7]).

6.6 SAML profile for LIGHTest

The SAML architecture is split into various components which are shown in . SAML Profiles specify how SAML assertions, protocols and bindings are combined. Moreover, a SAML profile defines extensions and constraints of the usage of SAML for a specific application.

The following profiles are defined in SAML 2.0:

- SSO Profiles
    - Web Browser SSO Profile
    - Enhanced Client or Proxy (ECP) Profile
    - Identity Provider Discovery Profile
    - Single Logout Profile
    - Name Identifier Management Profile
- Artifact Resolution Profile
- Assertion Query/Request Profile
- Name Identifier Mapping Profile
- SAML Attribute Profiles
    - Basic Attribute Profile

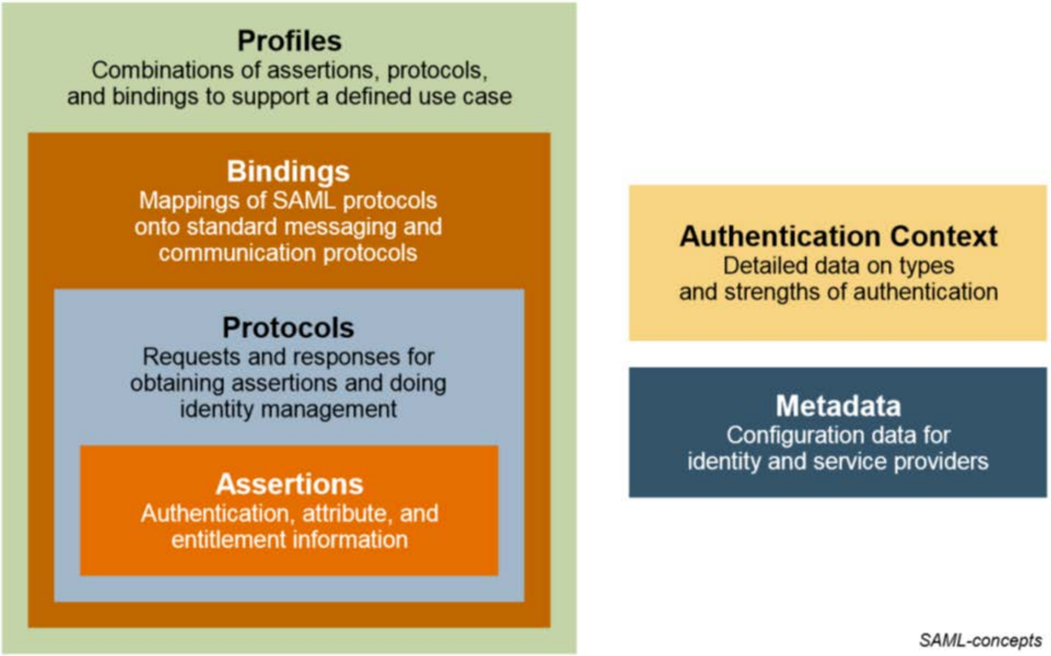
Profiles
Combinations of assertions, protocols, and bindings to support a defined use case
Bindings
Mappings of SAML protocols onto standard messaging and communication protocols
Protocols
Requests and responses for obtaining assertions and doing identity management
Assertions
Authentication, attribute, and entitlement information
Authentication Context
Detailed data on types and strengths of authentication
Metadata
Configuration data for identity and service providers
SAML-concepts

Figure 9: Basic building blocks of the SAML 2.0 architecture [9].

- X.500/LDAP Attribute Profile
- UUID Attribute Profile
- DCE PAC Attribute Profile
- XACML Attribute Profile

For the LIGHTest application we propose the Web Browser SSO Profile, since the mobile user (using a web app) wants to access a resource at a service provider, or to access an identity provider such that the service provider and that resource are understood. The web user authenticates (or has already authenticated) to the identity provider, which then produces an authentication assertion (possibly with input from the service provider) and the service provider consumes the assertion to establish a security context for the web user [9].

In order to allow interoperability between two entities, a further refinement of the Web Browser SSO profile is the SAML2int profile [10] , which narrows down the deployment options of the Web Browser SSO profile further.

To develop these existing profile definitions further towards a “LIGHTest SAML attribute profile” the eIDAS SAML Attribute Profile V1.1 can be taken as a reference. [8] The profile defines the SAML attributes to be used for the assertion of natural and legal person identity between eIDAS node. For the LIGHTest mobile ID trust propagation use case, the following attributes would be needed in the profile:

- Original ID:
    - LoA of the original ID (the source ID)
    - ID provider of this original ID
    - expiration date
    - a pointer to the certificate provided by the ID provider for validation (if needed).
- Derived ID (stored in your mobile device):
    - LoA (after the derivation)
    - the expiration date
    - the public key
    - the certificate
- Credential storage:
    - LoA (depending on how secure is the storage)
    - type of storage environment
- Device attestation:
    - LoA of the attestation scheme (provided by the manufacturer)
    - the authenticator
    - the attestation key

It will be the scope of the future tasks in WP 7 to refine this list further, based on the ongoing refinement of the LIGHTest pilot use cases.

7. Summary/Conclusions

In summary, this deliverable has outlined a concept of a strong link between FIDO credentials and an identity derived by a secondary ID issuer. Due to the special properties of scoped credentials in the user-centric and privacy friendly FIDO concept some technical hurdles had to be overcome. While other alterative are possible as well, the proposed concept is a balance between usability (number of authentications required by the user) and ease of integration for the relying party and ID provider (number of protocols and interfaces).

The technical aspects of implementation will have to be refined further during the implementation phase and will be tested with a demonstrator. In case that during this phase some smaller adaptions of the concept have to be made, they will be documented in the upcoming deliverables.

8. References

[1] ISO/IEC, *Identification cards - Integrated circuit cards,* International Standard, ISO/IEC 7816, Part 1 - 15, 1999 - 2011.

[2] G. Platform, „TEE Client API Specification V 1.0,“ 01 07 2010. [Online]. Available: https://www.globalplatform.org/specificationform.asp?fid=7339. [Zugriff am 05 05 2017].

[3] SIM Alliance, „OpenMobile API Specification V2.5,“ 28 01 2014. [Online]. Available: http://simalliance.org/wp-content/uploads/2015/03/SIMalliance_OpenMobileAPI2_05_release-Feb142.pdf. [Zugriff am 05 05 2017].

[4] FutureID, „D31.2 - Interface and Module Specification and Documentation,“ 01 05 2013. [Online]. Available: http://futureid.eu/data/deliverables/year1/Public/FutureID_D31.02%20_WP31_v1.0_Interface%20and%20module%20specification%20and%20documentation.pdf. [Zugriff am 05 05 2017].

[5] FIDO Alliance, „FIDO AppID and Facet Specification,“ 02 02 2017. [Online]. Available: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-appid-and-facets-v1.1-id-20170202.html. [Zugriff am 29 05 2017].

[6] ETSI, „ETSI EN 319 162-1 V1.1.1 (2016-04); Electronic Signatures and Infrastructures (ESI); Associated Signature Containers (ASiC); Part 1: Building blocks and ASiC baseline containers.,“ 2016.

[7] PKWARE, „Application Note: "APPNOTE.TXT - .ZIP File Format Specification", PKWARE® Inc.,“ 2012.

[8] FIDO Alliance, „FIDO UAF Specifications V1.1 - Architectrual Overview,“ 07 12 2016. [Online]. Available: https://fidoalliance.org/specs/fido-uaf-v1.1-id-20170202/fido-uaf-overview-v1.1-id-20170202.html. [Zugriff am 05 05 2017].

[9] E. Commission, „eID eIDAS profile,“ 16 12 2016. [Online]. Available: https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eID+eIDAS+profile. [Zugriff am 29 5 2017].

[10] OASIS, „Security Assertion Markup Language (SAML) V2.0 Technical Overview,“ 2008.

[11] SAML2int, „SAML 2.0 Interoperability Deployment Profile,“ [Online]. Available: http://saml2int.org/profile/current/ . [Zugriff am 29 05 2017].

9. Project Description

LIGHTest project to build a global trust infrastructure that enables electronic transactions in a wide variety of applications

An ever increasing number of transactions are conducted virtually over the Internet. How can you be sure that the person making the transaction is who they say they are? The EU-funded project LIGHTest addresses this issue by creating a global trust infrastructure. It will provide a solution that allows one to distinguish legitimate identities from frauds. This is key in being able to bring an efficiency of electronic transactions to a wide application field ranging from simple verification of electronic signatures, over eProcurement, eJustice, eHealth, and law enforcement, up to the verification of trust in sensors and devices in the Internet of Things.

Traditionally, we often knew our business partners personally, which meant that impersonation and fraud were uncommon. Whether regarding the single European market place or on a Global scale, there is an increasing amount of electronic transactions that are becoming a part of peoples everyday lives, where decisions on establishing who is on the other end of the transaction is important. Clearly, it is necessary to have assistance from authorities to certify trustworthy electronic identities. This has already been done. For example, the EC and Member States have legally binding electronic signatures. But how can we query such authorities in a secure manner? With the current lack of a worldwide standard for publishing and querying trust information, this would be a prohibitively complex leading to verifiers having to deal with a high number of formats and protocols.

The EU-funded LIGHTest project attempts to solve this problem by building a global trust infrastructure where arbitrary authorities can publish their trust information. Setting up a global infrastructure is an ambitious objective; however, given the already existing infrastructure, organization, governance and security standards of the Internet Domain Name System, it is with confidence that this is possible. The EC and Member States can use this to publish lists of qualified trust services, as business registrars and authorities can in health, law enforcement and justice. In the private sector, this can be used to establish trust in inter-banking, international trade, shipping, business reputation and credit rating. Companies, administrations, and citizens can then use LIGHTest open source software to easily query this trust information to verify trust in simple signed documents or multi-faceted complex transactions.

The three-year LIGHTest project starts on September 1st and has an estimated cost of almost 9 Million Euros. It is partially funded by the European Union's Horizon 2020 research and innovation programme under G.A. No. 700321. The LIGHTest consortium consists of 14 partners from 9 European countries and is coordinated by Fraunhofer-Gesellschaft. To reach out beyond Europe, LIGHTest attempts to build up a global community based on international standards and open source software.

The partners are ATOS (ES), Time Lex (BE), Technische Universität Graz (AT),EEMA (BE), G&D (DE), Danmarks tekniske Universitet (DK), TUBITAK (TR), Universität Stuttgart (DE), Open Identity Exchange (GB), NLNet Labs (NL), CORREOS (ES), IBM Danmark (DK) and Globalsign (FI). The Fraunhofer IAO provides the vision and architecture for the project and is responsible for both, its management and the technical coordination.
The Fraunhofer IAO provides the vision and architecture for the project and is responsible for both, its management and the technical coordination.