Follow-up to Cryptographers' Feedback on the EUDI ARF: Integrating an Anonymous-Credential-based Solution to Hardware

July 2024

Abstract

Issuing digital identity documents to citizens at large scale so far has failed despite many attempts, mainly because citizens did not (want to) buy the hardware necessary to securely manage and use cryptographic keys. Today, for the first time in history, many users do have mobile phones that contain a piece of secure hardware that is able to manage such keys and their secure use part of by the operating system. This technology is generally referred to as Passkeys. Similarly, there is hardware and software systems available that allow one to build system to securely issue digital credentials. While there are many standards for cryptographic (signature) scheme for such systems, the most widely used is ECDSA, and for Passkeys that is the only scheme that is currently supported. The use of identity and personal information is very sensitive and requires sufficient protection, in particular in connection with any electronic media. That is, a digital identity scheme must allow users to have different identities with different parties and be able to selectively reveal attributes from a credential obtained. The goal of this document is to describe an identity system that provides privacy and can be implemented with the existing hardware and operating system and the algorithm that hardware supports. I.e., it can be realized solely in the application space and can be rolled out without requiring any change in hardware, firmware, or operating systems. In particular, the document describes a digital identity and credential scheme that addresses:

- a user's identity is bound to a passkey device that supports (only) ECDSA,
- an identity provider can issue credentials to a user's identity, the credentials containing a number of attributes (also here ECDSA is the algorithm of choice),
- users can present a credential to a relying party where they can decide to reveal only a selection of the attributes (and in such a way that the presentation transcript can not be linked to the original credential issuance transaction unless a unique attribute was revealed). This presentation is realized with zero-knowledge proofs.

For simplicity of the exposition and to initiate an iterative discussion, this document does not (yet) describe 1) how to present predicates over attributes, 2) describes only the very simple case of one issuer (and not a system where there is several issuers and users are not fully identified to all of them, and 3) does not consider revocation. These functionalities can be added but of course add some complexity. The scheme is described at a high-level only, in particular, 1) a number of cryptographic implementation details are left out and 2) no security proofs are given. Both of which is left as future work and will need to be done.

1 Introduction

The purpose of this document is to outline a basic construction that 1) is implementable in short time and 2) addresses the main requirements as we understand them (foremost hardware binding, efficiency, practical deployability, and privacy). It does however not offer all features that one would expect such as revocation and supporting multiple issuers where the user is not fully identified to the issuers. While these features can be added, we prefer to first have an iterative approach towards the final specification. Thus the document

also serves to clarify those requirements and to agree on a way forward to 1) agree on these requirements, 2) what specification(s) of the scheme is needed in 3) what time frame, and 4) forming a team to work on this specification.

With digital credentials (anonymous or not), the issue of preventing a user from sharing their credential is inherently difficult due to digital data being easy to copy. While we can mitigate this by adding some biometric aspect, requirement for presenting multiple credentials belonging to a single person, or disincentivize sharing through risk or policies, we give a method of adding a physical aspect that requires transferring the physical component to share the credential. Specifically, we require credential presentation to require interaction with trusted hardware on the user's phone, and therefore require a user to share their phone in order to allow others to use their credentials.

Construction Outline. Working with the constraint that secure elements on phones must be used as-is (i.e., the passkey technology supporting ECDSA signatures), we give a straightforward way of integrating the secure element with any credential.

The general idea is as follows. Let PK_Dbe the public key of a secure element of a user's phone. When the user is issued a credential from the Identity Provider, the user's device's public key $PK_D$is included as an attribute in its credential. Now, when a user wishes to present his credential to the Relying Party, he must also prove that he is in current possession of the secure element corresponding to the public key PK_Din his credential. This can be done by having the Relying Party send a unique string *ctxt* and the user returning a signature on that value produced by his phone's secure element. The string should bind the presentation to the context where the credential is presented and it must be ensure that it has not been used before. To do this without requiring the user reveal $PK_D$to the Relying Party, the user can use generic zero-knowledge techniques to prove they are 1) in possession of a signature on *ctxt* that verifies under an undisclosed attribute of their credential, 2) in possession of a signature on that credential that verifies under the issuer's public key PK_I, and 3) if an additional attribute is revealed, that attribute is also contained in the credential.

Using modern generic zero-knowledge proof system in such a construction are very efficient and hence it is no longer necessary to use a specific signature scheme (such as BBS+ or CL) that were designed to used discrete logarithm proofs systems because at that time these were the only practical proofs systems.

Having said this, we still need to consider how the attributes are encoded into the message signed by the issuer to achieve a very practical system. Let us explain. A cryptographic signature scheme (and ECDSA in particular) takes as input a message in the format of a binary string, hashes it to obtain an algebraic value that is then used in the signing and verification algorithms. The system sketched above needs to make some statements about attributes (i.e., the user's key $PK_D$and the revealed attributes) and hence needs to consider the encoding of the attributes into the binary string that is hashed and signed. In practice such a mapping from structured data (such as a credential) is done using encoding scheme such as JSON. While such encoding have advantages in terms of developer friendliness, they will make the zero-knowledge proofs slow. It is thus important that in the construction, a more low level encoding is used. That does not mean that at a higher level API encodings such JSON cannot, it just means that we need to specify an low level API encoding for the credential format that is less "bloated."

We note out that while the described scheme is tailored towards ECDSA because that is the scheme that is prevalent in the hardware in the population, the approach itself is "crypto agile": it can be instantiated with other cryptographic schemes including ones that are post-quantum secure. In fact, the approach presented here seems to be the best one for a post-quantum credential system. In that sense, the scheme proposed is "post-quantum ready."

Document Organization. We review basic preliminaries in Section 2 before going into the construction in Section 3. In Section 3.1 we give an outline of the construction with the exact signing algorithms and zero-knowledge protocols abstracted away. In Section 3.2 we give the protocol with the abstractions replaced by (ey: word)

2 Preliminaries

ECDSA. The Elliptic Curve Digital Signature Algorithm (ECDSA) is the elliptic curve analog of the Digital Signature Algorithm (DSA) and is standardized by NIST [oST23]. We let $\mathbb{G}$ denote the elliptic curve group, q the order of the group, and G a generator of the group. ECDSA additionally makes use of a hash function H, which we instantiate with SHA-256.
A public and private key pair for ECDSA is generated as:

1. Uniformly choose a secret key $\mathsf{sk} \leftarrow \mathbb{Z}_q^*$.
2. Calculate the public key as $\mathsf{pk} = \mathsf{sk} \cdot G$.
3. Output $(\mathsf{pk}, \mathsf{sk})$.

To sign a message m under a secret key sk, the signer does:

1. Uniformly choose an instance key $r \leftarrow \mathbb{Z}_q^*$.
2. Calculate $R = r \cdot G$ and let r_x be the x-coordinate of R, modulo q.
3. Calculate
$$s = \frac{H(m) + \mathsf{sk} \cdot r_x}{r}$$
4. Output the signature as (s, r_x).

To verify a signature (s, r_x) on a message m under a public key pk, the verifier does as follows:

1. Calculate
$$R' = \frac{H(m) \cdot G + r_x \cdot \mathsf{pk}}{s}$$
and let ${r_x}'$ be the x-coordinate of R', modulo q.
2. Output 1 if and only if ${r_x}' = r_x$. Output 0 otherwise.

Zero-Knowledge Proofs. Interactive Oracle Proofs (IOP) are a type of zero-knowledge proof (ZKP) system that aims to provide efficient and scalable proofs with strong security guarantees. IOPs enable a verifier to be convinced after having only queried the proof at random locations rather than reading the entire proof and require interaction between a prover and a verifier. Much work has been done in this space minimizing the computational and communication overhead typically associated with zero-knowledge proofs, making it well-suited for practical applications, especially in environments such as privacy-preserving protocols.

For the ZKP in our application, we also make use of a polynomial commitment scheme to commit to the witness for the proof. A polynomial commitment scheme allows a prover to commit to a polynomial and later prove that the evaluation of this polynomial f on a specific point is correct, i.e. $f(x) = v$ for public values v and x. In our context, the function is defined by the witness for our zero-knowledge proof Ligero [AHIV17] is a zero-knowledge argument of knowledge protocol based on the MPC-in-the-head method of zero knowledge proofs and only relies on collision-resistant hash-functions. On our case the function is defined by the witness, the image is 1 and the pre-image is the evaluation on the verification of the ECDSA signature by the issuer and by the user's device as specified below in the high-level description. When used as a polynomial commitment scheme, Ligero allows one to prove an inner-product with a public vector, prove linear constraints on committed values, and prove a small number of quadratic constraints between committed values.

3 Construction

In this section we give the basic construction for an issuer, a user, and a relying party. It has two phases: 1) the issuer issues a credential to a user and 2) the user presents the credential to a relying party. The second phase can be repeated many times to different relying parties.

We first describe the basic setting and notations of our protocols. Then, we expand on the construction outlined in Section 1, which requires a user to be in possession of their phone when presenting a credential by incorporating the secure element of a user's phone into the credential presentation procedure.

Public Parameters. Our constructions make use of ECDSA, which we briefly reviewed in Section 2. ECDSA requires specifying an elliptic curve group $\mathbb{G}$ of order q and a generator G for $\mathbb{G}$. It additionally requires a hash function H (e.g., SHA-256), which maps to bit strings of length q. For the zero-knowledge proofs, their are additional parameters that will be added in the next version.

Credential Format. As mentioned the construction requires an efficient mapping from credential format to the binary string that is input to the signature scheme's sign algorithm. As there will be many different credential formats, it is not possible a format that will fit all of them. We assume there is a public registry (standard) that defines mappings from a credentials' attributes to binary strings (and their length). There will be further work needed here. For now we assume that the first bits of the message encode PK_D, the next part a credential specifier *csp* (that defines how many attributes will follow and how much space they take and how they are encoded), and then followed by n attributes $a_1, \ldots, a_n$. The attributes will typically include an expiration date, a serial number of the purpose of revocation, and information about user.

User's Device. A user's device contains a secure component that is able to issue signatures under a public key that only the device's secure element knows the corresponding signing key for. ECDSA has widespread support among secure elements, and we give our constructions assuming the secure element issues ECDSA signatures. We will refer to the ECDSA public key associated with the user's device's secure element as PK_D.

Identity Provider. Our constructions consider the Identity Provider to be issuing ECDSA signatures for the credentials, but the techniques can applied to other types of signatures. We specify ECDSA signatures for efficiency reasons during the credential presentation phase since the user's device is also assumed to use ECDSA. We denote the ECDSA public key of the Identity Provider as PK_I.

Relying Party. We note that for our suggested instantiations , we do *not* require the Relying Party to have performed extra setup prior to the credential presentation. If a different zero-knowledge protocol than the one in Section 3.2 is used, this may require the Relying Party to perform a one-time setup prior to any credential presentations.

3.1 High level description

3.1.1 Credential Issuance

For an Identity Provider to issue a credential to a user, the two first establish a session. The user uses the public key of its secure element PK_D to authenticate the session. During this session, the user and Identity Provider perform any steps necessary to establish if a credential should be issued and what attributes should be included in the credential. Additional means might be required to establish that PK_D is indeed the public key of the specific user's device, such as being personal present at an office and using the device in the presence of supervising personnel. In the following figure we represent this process by "establish session" and "establish attributes."

Once the parties have determined which attributes should be included in the credential, the Identity Provider will then issue a credential of type *csp* with these attributes $a_1, \ldots, a_n$ to user that is bound to

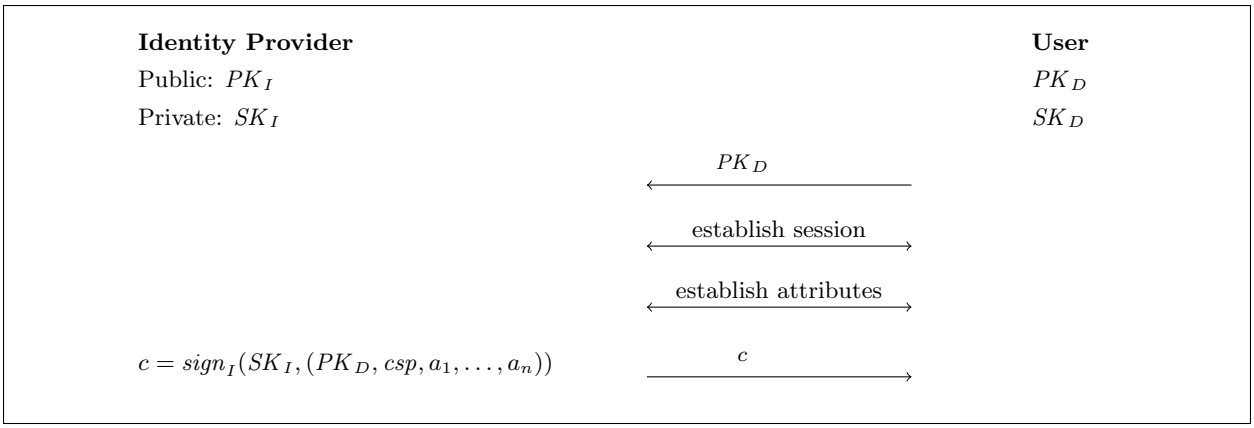

Figure 1: Credential Issuance.

the user's device key PK_D by computing the signature on these values and then sending this signature to the user's device. Specifically, the credential is computed as $c = sign_I(SK_I, (PK_D, csp, a_1, \ldots, a_n))$, where $sign_I$ is the signature scheme of the Identity Provider (e.g., ECDSA), SK_I is the Identity Provider's signing key, and the message signed is the concatenation of the user's device's public key PK_D with the credential specification and attributes. Note that a valid credential's specification *csp* will be required to define the number of attributes in the credential and each attribute's length.

3.1.2 Credential Presentation

For a user to present a credential to a Relying Party, the two must first establish a session, e.g., using the webauthn standard. We note that as the user very likely will be pseudonymous during this interaction, as webauthn generates a fresh public key for every url/origin. However, the protocol presented here can also be used in sessions that are established differently. Regardless of how a session is establish, the credential presentation should be bound this the session. To this end, we require the user and Relying Party to jointly define a session context *ctxt* that is unique to the interaction. This context can for instance be defined as the transcript of the key exchange protocol used to setup the end-to-end encryption between the user and the Relying Party, or as the session key that was established, as long as it is ensured that *ctxt* is a unique value with very high probability. Once the session and context *ctxt* have been established, the user produces the proof of its credential as follows.

The user produces a signature on *ctxt* using its device's secure element's public key as $s = sign_D(SK_D, ctxt)$, where $sign_D$ is the signature scheme of the secure element. Then, the user produces a polynomial commitment com to the function $f(PK_I, csp, ctxt) = ver_I(PK_I, (PK_D, csp, a_1, \ldots, a_n), c) \wedge ver_D(PK_D, ctxt, s)$ where ver_I, ver_D are the verification algorithms of the Identity Provider and device's secret element, respectively, and secret values PK_D, c, $a_1, \ldots, a_n$, and *ctxt* define f. Note this function corresponds to the user holding (1) a credential c that is a valid signature on PK_D, attributes $a_1, \ldots, a_n$, and specification *csp* with respect to the Identity Provider's public key PK_I and (2) a signature s on the unique context *ctxt* that verifies with respect to the device's public key PK_D. The user then sends this commitment com, the public key of the Identity Provider PK_I, and its credential specification *csp* to the Relying Party. Finally, the two parties execute an IOP, with the user acting as the prover and the Relying Party as the verifier, for the statement $f(PK_I, csp, ctxt) = 1$, where the function f is unknown to the verifier but is the value committed to in com. At the conclusion of the interaction, if the Relying Party accepts the proof, then the Relying Party can be convinced the user holds a valid credential. Otherwise, if the proof fails, then the Relying Party can conclude that either the user does not hold a valid credential or the user is not in possession of the device which is associated the credential it attempted to present.

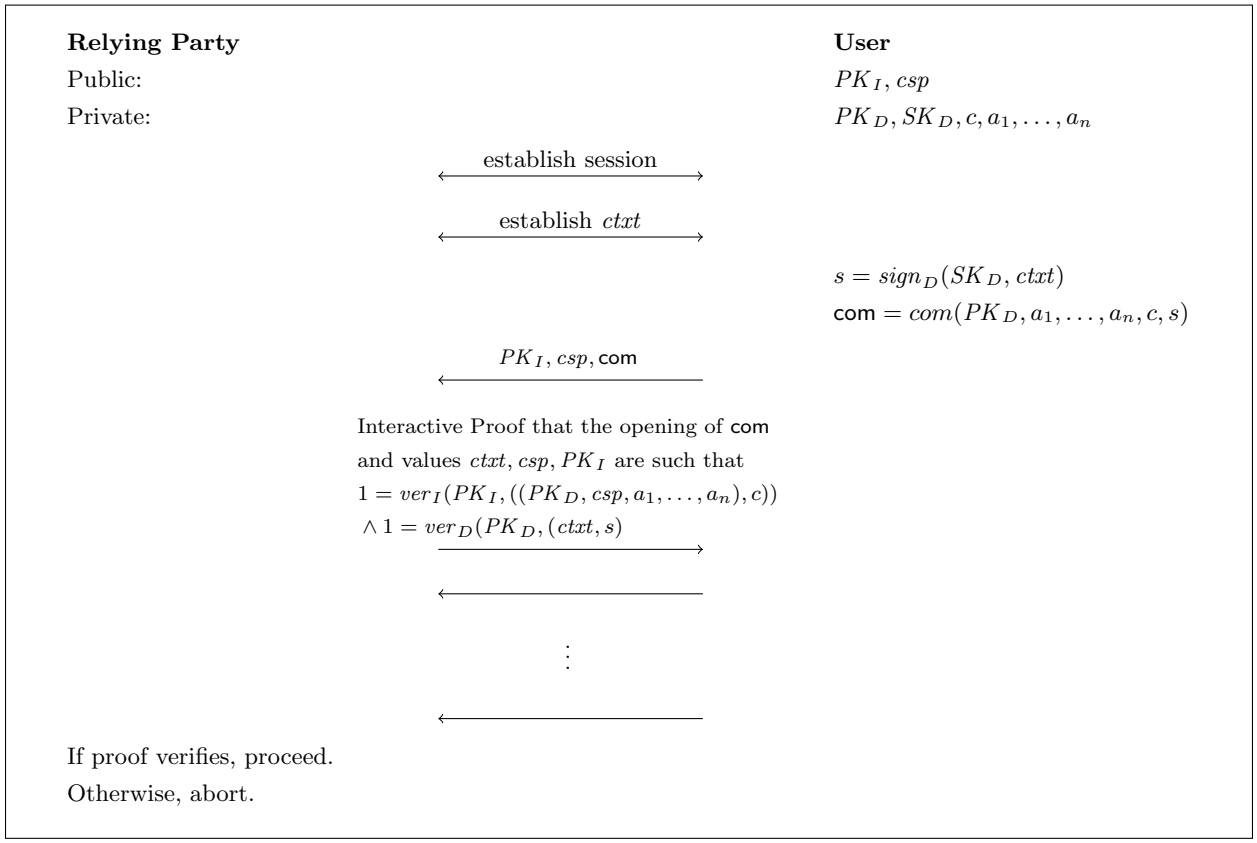

Figure 2: Credential Presentation.

3.2 Concrete instantation

One can use an arithmetic circuit based proof system to implement this zero-knowledge proof system. One modern, generic recipe is to choose a IOP system and a compatible polynomial commitment scheme. While there are many possible choices, one specific choice is to use a sumcheck-based protocol for the IOP, and a hash-based polynomial commitment scheme. Sumcheck, for example, when used in a circuit-based protocol, has a linear-time prover, and a careful implementation can achieve a very small constant overhead. In particular, the sumcheck prover does not use an FFT internally like many other proof systems. This approach is taken by a recent unpublished work by one of the authors. Their Hyrax implementation used a discrete-log based commitment scheme and a bulletproof to prove the opening. For certain small proofs, this solution is performant. The Ligero polynomial commitment also has size that grows asymptotically as square-root of the witness size, but has a larger implementation constant. However, Ligero only requires hashing, and this circuit exceeds the cross-over point—therefore, they chose to use it instead.

A paper describing the full system is planned to be submitted to a conference, but this work was done under a company arrangement, and thus must be internally reviewed.

4 Considerations

As mention already, the protocol presented is very bare bones and will need to be extended for meet all requirements.

For instance, one might require that is it possible to revoke credentials. There are many approaches to this including short-lived credentials and regularly publishing signatures on all ranges of all valid serial numbers or a hash-tree of them. At first sight, the latter seems to be the best approach for our setting.

Next, one might require predicates over attributes such as proving that one is older than 18 given a certified birth date. With proper encoding of attributes, proving that such predicates are true will be easy due to the use of generic zero-knowledge proofs.

We all have several credentials in our wallet. While these typically all include our full names as attributes, in an online setting, we have many more credentials which hardly ever contain our name but user some email address. Thus it might be important that it is possible to issue credentials to users that are pseudonym and then use different credentials together. Such scenarios can also be addressed with extensions to the protocol. The question here is what scenarios are envisioned which merits more discussion.

References

[AHIV17] Scott Ames, Carmit Hazay, Yuval Ishai, and Muthuramakrishnan Venkitasubramaniam. Ligero: Lightweight sublinear arguments without a trusted setup. pages 2087–2104, 2017. doi:10.1145/3133956.3134104.

[oST23] National Institute of Standards and Technology. Digital signature standard (dss). Technical report, U.S. Department of Commerce, Washington, D.C., 2023. doi:10.6028/NIST.FIPS.186-5.