POLITECNICO DI TORINO
Repository ISTITUZIONALE

On Delegation of Verifiable Presentations

Original
On Delegation of Verifiable Presentations / Flamini, A., Guglielmino, E., Orabona, V., Gangemi, A.. - (2025), pp. 26-38. (Trends in Digital Identity 2025 Bologna (IT) 3 Febbraio 2025).

Availability:
This version is available at: 11583/3001374 since: 2025-06-30T09:54:02Z

Publisher:
CEUR-WS.org

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

27 November 2026

On Delegation of Verifiable Presentations

Andrea Flamini[1,2], Andrea Gangemi[2], Enrico Guglielmino[2] and Vincenzo Orabona[3]

[1]Department of Mathematics, University of Trento, Povo, 38123 Trento, Italy
[2]Department of Mathematical Sciences, Politecnico di Torino, Corso Duca degli Abruzzi 24, 10129 Torino, Italy
[3]Eustema S.p.A, Via Carlo Mirabello, 7, 00195 Roma, Italy

Abstract

In this work, we study the problem of producing a delegation of verifiable presentations derived from verifiable credentials to enable a credential holder (the delegator) to securely authorize another party (the delegatee) to present a credential on their behalf. We define the notion of a *verifiable presentation delegation scheme*, with the core algorithms for delegation issuance, delegated presentation, and presentation verification, and formalize the security properties that such a scheme must satisfy, namely correctness and unforgeability. Then, we design a verifiable presentation delegation scheme that can be applied to the verifiable credentials used in the European Digital Identity Wallet Architecture Reference Framework (EUDI ARF) and we prove that our scheme satisfies the security properties under the assumption of a secure digital signature scheme. Finally, we briefly discuss and provide some insight on how to instantiate our scheme in the context of the European Blockchain Services Infrastructure (EBSI) and EUDI frameworks.

Keywords

Verifiable Credential, Verifiable Presentation, Delegation, eIDAS 2.0, EUDI ARF, EBSI

1. Introduction

The eIDAS 2.0 regulation [1] aims to regulate electronic identification and trust services in the EU's internal market and to improve cross-border interoperability across Europe. One of the main tools used to achieve these goals is the adoption of verifiable credentials (VC), which will be stored in a digital wallet called the EUDI Wallet [2]. VCs are the digital analogue of physical credentials, and their security relies on the use of cryptographic techniques. VCs are issued by *issuers* who sign them and deliver them to the credential *holders*. The holder of a VC stores it and can use it to prove specific claims (or statements) about their identity to a *verifier* by creating a verifiable presentation (VP). In general, roles are dynamic and context-dependent, allowing a user to function as a holder, verifier, or issuer based on the specific operations they wish (or are allowed) to perform. For example, an organization acting as a verifier in one transaction can assume the role of an issuer in a subsequent transaction. The structure and policies for managing Verifiable Credentials, including the specification of credential attributes and trust models, are detailed in frameworks such as EBSI [3] or EUDI ARF [2].

One of the main advantages of using VCs over their physical counterparts is the ability to selectively disclose a subset of the attributes included in a credential. This operation minimizes the amount of data shared with third parties, in accordance with the privacy principles required by the GDPR [4]. Current discussions on the design of the EUDI Wallet and the types of VCs it should support have highlighted two main approaches [5]. The first approach relies on hiding commitments and standard signatures [6], while the second utilizes anonymous credentials based on signature schemes that support Non-Interactive Zero-Knowledge Proofs (NIZKPs) following the framework of Camenisch and Lysyanskaya [7]. Both approaches enable VCs to support selective disclosure of attributes and are described in detail in [8].

TDI 2025: 3rd International Workshop on Trends in Digital Identity, February 3, 2025, Bologna, Italy
andrea.flamini@unitn.it (A. Flamini); andrea.gangemi@polito.it (A. Gangemi); enrico.guglielmino@polito.it (E. Guglielmino); v.orabona@eustema.it (V. Orabona)
0000-0002-3872-7251 (A. Flamini); 0000-0001-9689-8473 (A. Gangemi); 0009-0004-1394-4761 (E. Guglielmino)
© 2025 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR-WS.org/Vol-3968/paper2.pdf

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

1.1. Delegation of VPs

An important extension of VC schemes that would improve their usability is the ability to support delegation of VPs. In this context, a credential holder (the delegator), using its VC, can create a delegation that instructs another holder (the delegatee) to act on its behalf to execute a *specific and predetermined* operation while interacting with a verifier. VP delegation is developed as a mechanism allowing one to cover use cases such as: someone authorizes a family member to pick up a prescription from the pharmacy, an elderly person authorizes their adult child to represent them in a public online service, or a person delegates an intermediary to perform financial operations on its behalf.

For simplicity in this work, we consider a system in which every user is provided a "main credential", like an ID card, by a trusted issuer that every holder can use to delegate other holders. Our definition of a VP delegation scheme builds on top of such a VC scheme. In particular, we describe a delegation mechanism that allows a delegator to generate delegations that must specify:

- the *scope* for which the delegation is being created. This might include a period of validity, an identifier of the verifier to which it must be presented, and an identifier of the operation that the delegatee must perform;
- the *delegatee identifier*, a statement that the identity (or the VC) of the delegatee must satisfy. For example, the delegatee identifier could be "the delegatee is over 18", then any over 18 holder would be allowed to present such delegation, or more specific statements such as "the delegatee name is *Name* and their surname is *Surname*", in such a case only a holder named *Name Surname* would be allowed to present such delegation. This identifier is sent by the delegatee to the delegator before the latter produces the delegation;
- the *delegator payload*, a statement about the delegator's identity that contains (at least) the information the verifier would request from any holder to perform the operation specified in the scope;
- a *proof* that the delegator identity (or VC) satisfies the delegator payload.

The delegatee, with the delegation, interacts with the verifier to present the delegation and to execute the operations specified in the scope. The delegatee proves to be an authorized delegatee by showing, using its own VC, that it satisfies the delegatee identifier information specified in the delegation. Finally, the verifier checks that the delegation and the presentation of the delegation are valid and accepts (or rejects) the delegated presentation, allowing the delegatee to perform (or preventing the delegatee from performing) the operation specified in the scope.

1.2. Motivation, related works and contribution

With the upcoming enforcement of the eIDAS 2.0 regulation, the ability to delegate presentations has gained significant importance. This is confirmed by its inclusion in the EUDI Wallet Reference Implementation Roadmap [1]. In fact, EU citizens will soon be issued verifiable credentials that will be used to securely prove statements about their identity. In addition to that, VCs can be used to enhance the security of how users delegate specific, predetermined tasks to others, providing a more secure alternative to the paper-based methods too often used today.

A related problem, which has received significantly more attention in the literature, is the delegation of the issuance of anonymous credentials while concealing the delegation chain and revealing only the root issuer. This problem was firstly approached by Chase and Lysyanskaya [9] and since then many constructions have been proposed: some are [10, 11, 12, 13]. The use of anonymous credentials, combined with the ability to anonymously delegate the issuance of credentials, allows users to enjoy a very high level of privacy while lightening the workload on issuers. An interesting use case of delegatable anonymous credential is the issuance of ID cards to European citizens: the European authority delegates the member states to issue ID cards on its behalf, so that it does not have to carry out this burdensome operation on its own. The member states, using their delegated credentials, issue

[1] See the Delegation Attestation in the "Future" section at https://github.com/orgs/eu-digital-identity-wallet/projects/24

the ID cards to their citizens. With these credentials, the citizens can prove to verifiers to be European citizens without revealing the authority who issued their credential, and therefore their country.

The same problem applied to non-anonymous credentials [6] can be trivially solved (losing most of the privacy-preserving features) using standard digital signature schemes by instructing the issuer to sign the public key of its delegatee, as is well explained in [12, Section 1].

In this work instead, we focus on the delegation of VPs, a problem surely related but with different applications from the delegation of VCs. In both cases the delegatee takes charge of carrying out operations on behalf of the delegator: for what concerns the delegation of VCs, such an operation is the issuance of VCs to users, and in our case such an operation is the showing of VPs to verifiers.

To the best of our knowledge, delegation of VPs has not been formalized in the existing literature, and for this reason, we fill this gap laying the ground to formalize the notion of VP delegation scheme. We do that by giving a formal definition of a VP delegation scheme and the security properties it must satisfy, namely correctness and unforgeability. Then, we describe a natural VP delegation scheme that is compliant with the VC schemes specified in the EUDI ARF, and we prove its security. Finally, we discuss the compatibility of our scheme in the context of the EBSI and EUDI frameworks.

1.3. Notation

Let λ denote the security parameter. The issuer's private and public keys are denoted as $\mathsf{sk}_{\mathsf{Iss}}$ and $\mathsf{pk}_{\mathsf{Iss}}$, respectively. The set of messages or attributes is denoted by $\mathbf{a} = (a_i)_{i\in[l]}$, where $[l]$ represents the set $\{1, \ldots, l\}$. When presenting a VC supporting selective disclosure, a subset of attributes, whose indices are denoted by Hid, can be hidden, while the set of indices of the disclosed attributes is $\mathsf{Rev} = [l] \setminus \mathsf{Hid}$. We denote by stmt the set of revealed attributes, and with $\mathcal{S}$ the set of possible statements. The *delegator*, *delegatee*, and *verifier* are indicated by D, Δ, and V, respectively. Each party is provided with a single VC. The delegator's credential is represented as $\mathsf{cred}_{\mathsf{D}}$, and the delegatee's credential is denoted as cred_{Δ}. The delegatee identifier is indicated by Δ_{ID}, the delegator payload is denoted as DP, while the delegation scope is referred to as scope. In the security definitions, we will define the following tables: a credential table CT, a presentation table PT, and a delegation table DT. Finally, we consider a function $f : \mathbb{N} \to \mathbb{R}$ to be *negligible* if, for every $c > 0$, there exists an integer $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, it holds that $f(n) = O(1/n^c)$. This notation will be used throughout the paper to formalize the schemes and algorithms presented.

2. Verifiable Credential Schemes

A verifiable credential scheme is defined by a set of algorithms and protocols that establish secure processes to issue, present and verify digital credentials [8].

Definition 1 (Verifiable Credential Scheme)**.** *A* verifiable credential scheme *is defined by the following algorithms:*

- ***Issuer Setup*** *(*$\mathsf{IssuerSetup}(\lambda) \xrightarrow{\$} \mathsf{pp}, (\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}})$*): this algorithm is executed by the issuer, and it generates the public parameters* pp *for the verifiable credential scheme together with the issuer key pair.*
- ***Credential Issuance*** *(*$\mathsf{CredIssuance}(\mathsf{sk}_{\mathsf{Iss}}, \{a_i\}_{i\in[l]}) \xrightarrow{\$} \mathsf{cred}$*): The issuer executes this algorithm to generate a verifiable credential* cred *for the attributes* $\{a_i\}_{i\in[l]}$*.*
- ***Credential Presentation*** *(*$\mathsf{CredPresentation}(\mathsf{cred}, \mathsf{stmt}) \xrightarrow{\$} \mathsf{pres}$*): This algorithm is executed by the holder to generate a proof (the presentation* pres*) that it possesses a credential* cred *that satisfies a statement* stmt[2]*.*

[2]The presentation can allow the holder to selectively disclose the attributes of the credential, revealing only the attributes necessary to prove stmt.

- ***Presentation Verification*** *(*$\mathsf{PresVer}(\mathsf{pres}, \mathsf{stmt}) \xrightarrow{\$} \{0,1\}$[3]*): The verifier executes this algorithm to verify the validity of the presentation provided by the holder, ensuring that it satisfies* stmt.

In this work, we focus on the class of VCs schemes that are currently adopted in the EUDI Wallet Architecture Reference Framework (ARF) [2, Annex 2, Topic 12]. The VC schemes considered in the EUDI Wallet ARF must have one of the following formats: mDOC, described in [14, 6], SD-JWT, described in [15], or W3C VCDM, described in [16]. These different verifiable credential formats are based on the same cryptographic mechanism that we abstract in this section. The cryptographic primitives used are (1) hiding commitments based on hash functions and (2) digital signature schemes $\mathcal{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$.

The hiding commitment based on a hash function H is defined as follows: to commit to a message m, sample a random $\mathsf{salt} \xleftarrow{\$} \{0,1\}^\lambda$ and compute $\mathsf{com} = H(m||\mathsf{salt})$. To open a commitment com, it is necessary to reveal the message-salt pair $(m||\mathsf{salt})$, and the verifier can check if $\mathsf{com} = H(m||\mathsf{salt})$. The digital signature scheme $\mathcal{DS}$ can be any UF-CMA secure digital signature scheme [4].
In the rest of this work, we omit the public parameters pp in the inputs of most of the algorithms described. Even $\mathsf{pk}_{\mathsf{Iss}}$ is omitted, since we assume for simplicity that there is only one issuer in the system.

Definition 2 (ARF-Compliant Verifiable Credentials)**.** *Let* $\mathcal{DS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Vf})$ *be a digital signature scheme and* H *a cryptographic hash function. The verifiable credentials which are* ARF-Compliant *can be described as follows:*

- ***Issuer setup:*** $\mathsf{pp}, (\mathsf{sk}_{\mathsf{Iss}}, \mathsf{pk}_{\mathsf{Iss}}) \xleftarrow{\$} \mathsf{IssuerSetup}(\lambda)$
  The issuer executes the issuer setup algorithm which consists in executing the Setup *and* KeyGen *algorithms of the underlying digital signature scheme* $\mathcal{DS}$.
- ***Credential issuance:*** $\mathsf{cred} \xleftarrow{\$} \mathsf{CredIssuance}(\{a_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{Iss}})$
  The issuer executes the following operations:
  - *sample uniformly at random salts* $\mathsf{salt}_1, \ldots, \mathsf{salt}_l \xleftarrow{\$} \{0,1\}^\lambda$*;*
  - *compute* $\mathsf{com}_i \leftarrow H(a_i||\mathsf{salt}_i)\ \forall i \in [l]$*;*
  - *generate* $(\mathsf{sk}_{\mathsf{cred}}, \mathsf{pk}_{\mathsf{cred}}) \xleftarrow{\$} \mathsf{KeyGen}(\mathsf{pp})$*;*
  - *sign the commitments and the public key of the credential* $(\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}})$ *computing* $\sigma \xleftarrow{\$} \mathsf{Sign}((\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \mathsf{sk}_{\mathsf{Iss}})$*;*
  - *set* $\mathsf{cred} \leftarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \{a_i\}_{i\in[l]}, \{\mathsf{salt}_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{cred}})$[5].
- ***Credential presentation****:* $\mathsf{pres} \xleftarrow{\$} \mathsf{CredPresentation}(\mathsf{cred}, \mathsf{stmt}, \mathsf{nonce})$
  The statement $\mathsf{stmt} = \{a_i\}_{i\in\mathsf{Rev}}$ *is given by the set of attributes that the holder wants to reveal* $\mathsf{Rev} \subseteq [l]$*, and the nonce* nonce *is sent by the verifier to the holder to guarantee the freshness of the presentation. The holder performs the following operations:*
  - *compute* $\mathsf{pres}' \leftarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \{\mathsf{salt}_i\}_{i\in\mathsf{Rev}}, \{a_i\}_{i\in\mathsf{Rev}}, \mathsf{nonce})$*;*
  - *sign* pres' *computing* $\sigma' \xleftarrow{\$} \mathsf{Sign}(\mathsf{pres}', \mathsf{sk}_{\mathsf{cred}})$*;*
  - *set* $\mathsf{pres} \leftarrow (\mathsf{pres}', \sigma')$.
- ***Presentation Verification****:* $\{0,1\} \xleftarrow{\$} \mathsf{PresVer}(\mathsf{pres}, \mathsf{stmt})$
  The verifier performs the following operations:
  - *parse* $\mathsf{pres} \rightarrow (\mathsf{pres}', \sigma')$ *and then* $\mathsf{pres}' \rightarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \{\mathsf{salt}_i\}_{i\in\mathsf{Rev}}, \{a_i\}_{i\in\mathsf{Rev}})$*;*
  - *verify the signature of the issuer:* $1 \leftarrow \mathsf{Vf}(\sigma, (\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}}), \mathsf{pk}_{\mathsf{Iss}})$*;*
  - *check that* $\mathsf{com}_i = H(a_i||\mathsf{salt}_i), \forall i \in \mathsf{Rev}$*;*
  - *verify the signature of the holder:* $1 \leftarrow \mathsf{Vf}(\sigma', \mathsf{pres}', \mathsf{pk}_{\mathsf{cred}})$.

  If the previous checks are satisfied, the verifier accepts and outputs 1, otherwise it outputs 0.

[3] In this context, $1 = \texttt{success}$, indicating that the presentation satisfies the statement, while $0 = \texttt{fail}$, indicating it does not.
[4] UF-CMA is a standard security model for digital signature which stands for "unforgeability under chosen message attacks".
[5] To ensure a secure binding between a credential and the device storing it, the secret key of the credential ($\mathsf{sk}_{\mathsf{cred}}$) is stored within a hardware security module (HSM) or a trusted execution environment (TEE) embedded in the device.

3. Verifiable Presentation Delegation Scheme

In this section we define the notion of *VP delegation scheme* which is built on top of a VC scheme according to Definition 1. A VP delegation scheme must define the algorithms (1) to allow the delegator D to create a delegation del, (2) to allow anyone to verify del, (3) to allow the delegatee Δ to present del to a verifier generating a presentation pres and (4) to verify the delegated presentation pres.

We define the input-output specifications of each algorithm that defines a VP delegation scheme.

Definition 3 (VP delegation scheme)**.** *A* VP delegation scheme $\mathcal{VPDS}$ *is defined by the the following algorithms:*

- Delegation issuance*:* $\mathsf{DelegIssuance}(\mathsf{cred_D}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}}) \xrightarrow{\$} \underbrace{(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})}_{\mathsf{del}}$ *where:*
  - DP *is the* delegator payload *containing the information that the delegator must disclose about its identity (e.g. it is entitled to withdraw a specific drug);*
  - scope *is the* delegation scope*, describing the operation that can be performed by the delegatee interacting with specified verifiers* [6]*;*
  - Δ_{ID} *is the* delegatee identity *which is a statement that must be satisfied by the delegatee presenting* del *(and its credential);*
  - π_{DP} *is a proof that the holder knows a credential* $\mathsf{cred_D}$ *for the claims in* DP*. The proof must be bound to* scope *and* Δ_{ID}*.*

  See Section 1.1 for a better intuition about the semantic meaning of these fields.
- Delegation verification*:* $\mathsf{DelegVer}(\mathsf{del}) \xrightarrow{\$} \{0, 1\}$
  This algorithm is executed by a delegatee (or by a verifier) to verify the validity of the delegation.
- Delegated presentation*:* $\mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_\Delta, \mathsf{nonce}) \xrightarrow{\$} \underbrace{(\mathsf{del}, \pi_{\mathsf{del}})}_{\mathsf{pres}}$

  The delegator contacts the verifier to which it wants to present the delegation. The verifier sends to the delegatee a random nonce *which is used to guarantee the freshness of the delegated presentation. After parsing* del *as* $(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})$*, the delegatee computes* π_{del} *which is a proof of knowledge of a credential* cred_Δ *satisfying the delegatee identity* Δ_{ID} *included in* del*.*
- Delegated presentation verification*:* $\mathsf{DelegPresVer}(\mathsf{pres}) \xrightarrow{\$} \{0, 1\}$
  This is the verification algorithm executed by the verifier. Parse pres *as* $(\mathsf{del}, \pi_{\mathsf{del}})$*. Upon receiving the proof* π_{del} *and the delegation* del*, the verifier checks that*
  - $\mathsf{DelegVer}(\mathsf{del}) \to 1$*;*
  - π_{del} *is a valid proof for the value* Δ_{ID} *in* del*;*
  - scope *included in* del*, which is part of* pres*, is satisfied.*

  If these checks pass, the delegated presentation is accepted, and the algorithm outputs 1.

Figure 1 describes the interaction flow between the delegator D, the delegatee Δ and the Verifier V.

4. Security Notions for VP Delegation Schemes

Correctness Property. The correctness property specifies that a delegation algorithm always allows a honest delegator (who generates a delegation for a DP consistent with its identity), to generate a delegation del and delegate another user. Then, if the user is a honest delegatee (its identity satisfies Δ_{ID}) it can always present del to a verifier satisfying scope.

[6] Note that these information must be encoded using a dictionary or schema which subsequently allows verifiers of delegated presentations to verify it. A more detailed analysis of this aspect is out of scope and will be treated in a future work.

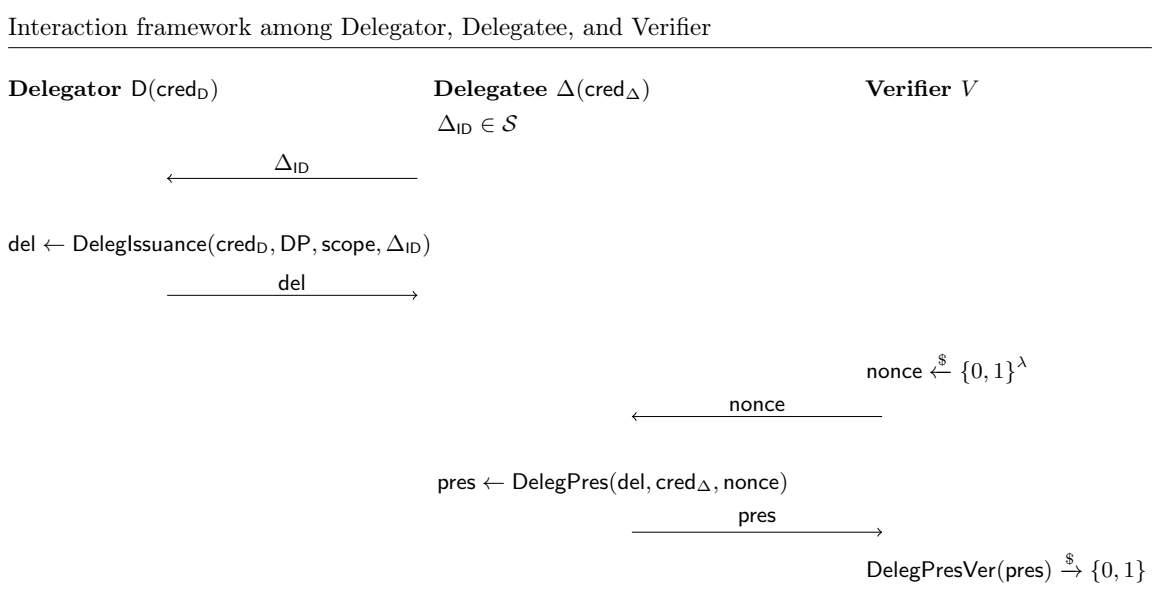

Figure 1: Interactions between the delegator D, the delegatee Δ, and the Verifier V.

Definition 4 (Correctness of VP delegation scheme)**.** *Given a VP delegation scheme*

$$\mathcal{VPDS} = (\mathsf{DelegIssuance}, \mathsf{DelegVer}, \mathsf{DelegPres}, \mathsf{DelegPresVer}),$$

we say that the scheme is correct if $\mathsf{DelegPresVer}(\mathsf{pres}) \to 1$ *whenever:*

- $\mathsf{del} \xleftarrow{\$} \mathsf{DelegIssuance}(\mathsf{cred_D}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$ *where* $\mathsf{cred_D}$ *satisfies the statements contained in* DP *(which is contained in* del*);*
- $\mathsf{pres} \xleftarrow{\$} \mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_\Delta, \mathsf{nonce})$*, where* cred_Δ *satisfies the statements contained in* Δ_{ID}*.*

Unforgeability Property. We consider two notions of unforgeability: the unforgeability of the delegation algorithm DelegIssuance, which means that an adversary cannot forge a delegation from a credential it does not possess, and the unforgeability of the delegation presentation algorithm DelegPres, which means that an adversary cannot present a delegation that is not issued to its identity (i.e., its identity does not satisfy Δ_{ID}). We capture these two notions of unforgeability in a single experiment.

The experiment captures that an adversary $\mathcal{A}$, after receiving the public key of the issuer $\mathsf{pk_{Iss}}$, the public parameters pp, and performing a training, cannot forge a delegation presentation. The training considers an adversary that can learn information from the system in the following ways: it can (1) corrupt a polynomial number of users, (2) receive a polynomial number of delegations from users it does not control, and (3) verify a polynomial number of delegated presentations (i.e., receive a polynomial number of presentations of its choice). These operations are modeled by allowing $\mathcal{A}$ to query a polynomial number of credentials to the oracle $\mathsf{O_{iss}}$, of delegations to the oracle $\mathsf{O_{del}}$ and of presentations of delegations to the oracle $\mathsf{O_{pres}}$.

Note that in a system supporting the delegation of VPs the adversary can see not only delegated presentations, but also presentations of verifiable credentials (generated using the algorithm $\mathsf{CredPresentation}(\mathsf{cred}, \mathsf{stmt})$). However, we omit the queries for credentials presentations because we assume that credentials presentations can be seen as delegated presentations associated to an empty delegation $(\mathsf{del}, \mathsf{nonce}) = (\perp, \perp)$.

Experiment 1 ($\mathsf{Exp}_{\mathcal{A}}^{\mathsf{DelPresUnforgeability}}(1^\lambda)$)**.** *The experiment is given by the following phases.*

Setup phase *$\mathcal{A}$ receives from the challenger of the experiment the public key* $\mathsf{pk_{Iss}}$ *of the issuer and the public parameters* pp *of the underlying VC scheme.*

Training phase *$\mathcal{A}$ can interact with random oracles $\mathsf{O_{iss}}$, $\mathsf{O_{del}}$ and $\mathsf{O_{pres}}$, to which it can send a polynomial number of issuing queries, delegation queries, and delegated presentation queries, respectively. Each query has the following input-output specification:*

1. *Issuing queries to $\mathsf{O_{iss}}$: $\mathcal{A}$ can query for the issuance of a credential for a set of attributes $\{a_i\}_{i\in[l]}$ of its choice; the oracle computes a credential* $\mathsf{cred} \leftarrow \mathsf{CredIssuance}(\{a_i\}_{i\in[l]}, \mathsf{sk_{Iss}})$*, adds* cred *to the* credential table CT *and sends it to $\mathcal{A}$.*
2. *Delegation queries to $\mathsf{O_{del}}$: $\mathcal{A}$ can query $\mathsf{O_{del}}$ for the issuance of a delegation for $(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP})$ of its choice; the oracle computes a delegation* $\mathsf{del} \leftarrow (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})$*, stores* del *in a* delegation table DT *and sends it to $\mathcal{A}$.*
3. *Delegated presentation queries to $\mathsf{O_{pres}}$: $\mathcal{A}$ can query for the presentation of a delegation for the values* $(\mathsf{del}, \mathsf{nonce})$*; the oracle $\mathsf{O_{pres}}$ generates a presentation* pres *for* $(\mathsf{del}, \mathsf{nonce})$*, adds* pres *to a* presentation table PT *and sends it to $\mathcal{A}$.*

Forgery phase *$\mathcal{A}$ eventually outputs a delegated presentation* $\mathsf{pres}^\star = (\mathsf{del}^\star, \pi_{\mathsf{del}^\star})$.

Winning conditions *Parse* $\mathsf{pres}^\star = (\mathsf{del}^\star, \pi_{\mathsf{del}^\star})$ *as* $(\Delta_{\mathsf{ID}}{}^\star, \mathsf{scope}^\star, \mathsf{DP}^\star, \pi_{\mathsf{DP}^\star}, \pi_{\mathsf{del}^\star})$.

The adversary wins the experiment if $\mathsf{DelegPresVer}(\mathsf{pres}^\star) \rightarrow 1$ *and one of the following conditions is satisfied:*

- forgery of the delegation*:* $\mathsf{del}^\star$ *is forged, i.e.*
  - *it is not a delegation queried by $\mathcal{A}$ to $\mathsf{O_{del}}$, i.e.* $\mathsf{del}^\star \notin \mathsf{DT}$*;*
  - $\mathsf{del}^\star$ *was not generated using the credentials issued to $\mathcal{A}$.*
- forgery of the delegated presentation*:* $\pi_{\mathsf{del}^\star}$ *is forged, i.e.*
  - $\forall \mathsf{pres} \in \mathsf{PT}, \mathsf{pres} \neq \mathsf{pres}^\star$*;*
  - $\pi_{\mathsf{del}^\star}$ *was not generated using the credentials issued to $\mathcal{A}$.*

Definition 5 (Unforgeability of VP delegation scheme)**.** *We say that a VP delegation scheme is unforgeable if for any PPT adversary $\mathcal{A}$ executing* $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{DelPresUnforgeability}}(1^\lambda)$*,*

$$\Pr\left[\mathcal{A} \text{ wins } \mathsf{Exp}_{\mathcal{A}}^{\mathsf{DelPresUnforgeability}}(1^\lambda)\right] \leq \nu(\lambda),$$

where $\nu(\lambda)$ is negligible in the security parameter λ.

In this security model, we assume that every interaction between holders/delegatees and verifiers is identified by a unique session identifier. This prevents an adversary from replaying messages exchanged during different sessions. When the protocol is instantiated, the protocol designer must take care of issues related to the creation of the communication sessions between the parties involved. To test the security of the approach used, it is advised to use automated tools like Tamarin or ProVerif; nonetheless, addressing this issue is beyond the scope of this paper.

5. Our Construction

We create an instance of a VP delegation scheme according to Definition 3 that is built on top of verifiable credential schemes supporting selective disclosure which are compliant with the EUDIW ARF [2], as described in Definition 2. Note that in this particular case all the proofs introduced in Definition 3 are digital signatures.

Definition 6 (ARF-Compliant VP Delegation Scheme)**.** *Given a verifiable credential scheme as defined in Definition 2, which uses the digital signature scheme $\mathcal{DS}$ and the hash function H, we can define the following delegation scheme (see Definition 3).*

- Delegation issuance: $\mathsf{DelegIssuance}(\mathsf{cred_D}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}}) \xrightarrow{\$} \underbrace{(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})}_{\mathsf{del}}$

  On input $(\mathsf{cred_D}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$ *with* $\mathsf{cred_D} = \big((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred_D}}), \{a_i\}_{i\in[l]}, \{\mathsf{salt}_i\}_{i\in[l]}, \mathsf{sk}_{\mathsf{cred_D}}\big)$, *the delegator* D *computes* π_{DP} *as a variant of a presentation of* $\mathsf{cred_D}$ *which is bound to* DP *but also to* scope *and* Δ_{ID}*:*

  - $\mathsf{pres}' \leftarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred_D}}), \{\mathsf{salt}_i\}_{i\in\mathsf{DP}}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$*;*
  - $\sigma' \xleftarrow{\$} \mathsf{Sign}(\mathsf{pres}', \mathsf{sk}_{\mathsf{cred_D}})$ *and* $\pi_{\mathsf{DP}} \leftarrow (\mathsf{pres}', \sigma')$*;*
  - *return* $\mathsf{del} \leftarrow ((\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}}))$.

- Delegation verification: $\mathsf{DelegVer}(\mathsf{del}) \xrightarrow{\$} \{0,1\}$
  To verify the delegation, parse:

$$\mathsf{del} \rightarrow (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}}), \quad \pi_{\mathsf{DP}} \rightarrow (\mathsf{pres}', \sigma'),$$

$$\mathsf{pres}' \rightarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred_D}}), \{\mathsf{salt}_i\}_{i\in\mathsf{DP}}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}}).$$

  Then, perform the following checks:

  - *Verify the signature of the issuer:* $1 \xleftarrow{?} \mathsf{Vf}(\sigma, (\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred_D}}), \mathsf{pk}_{\mathsf{Iss}})$. *This means that the delegation is created from a valid VC issued by* $\mathsf{pk}_{\mathsf{Iss}}$*;*
  - *Check that* $\mathsf{com}_i = H(a_i||\mathsf{salt}_i), \forall i \in \mathsf{DP}$. *This means that the delegation has a* DP *consistent with the credential used to generate it;*
  - *verify the signature* σ' *of* pres' *using the public key* $\mathsf{pk}_{\mathsf{cred_D}}$*:* $1 \xleftarrow{?} \mathsf{Vf}(\sigma', \mathsf{pres}', \mathsf{pk}_{\mathsf{cred_D}})$. *This means that the delegation (for* scope *and* Δ_{ID}*) was created by the holder of the associated credential.*

- Delegated presentation: $\mathsf{DelegPres}(\mathsf{del}, \mathsf{cred}_\Delta, \mathsf{nonce}) \xrightarrow{\$} \underbrace{(\mathsf{del}, \pi_{\mathsf{del}})}_{\mathsf{pres}}$

  After parsing $\mathsf{del} \rightarrow (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})$*, the delegatee computes* π_{del} *as a presentation of* Δ_{ID} *using* cred_Δ *which is bound to* del*:*

  - *compute* $\mathsf{pres}'' \leftarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta}), \{\mathsf{salt}_i\}_{i\in\Delta_{\mathsf{ID}}}, \mathsf{del}, \mathsf{nonce})$*;*
  - *the delegatee signs* pres'' *computing* $\sigma'' \xleftarrow{\$} \mathsf{Sign}(\mathsf{pres}'', \mathsf{sk}_{\mathsf{cred}_\Delta})$*, sets* $\pi_{\mathsf{del}} \leftarrow (\mathsf{pres}'', \sigma'')$ *and returns* $\mathsf{pres} \leftarrow (\mathsf{del}, \pi_{\mathsf{del}})$.

- Delegated presentation verification: $\mathsf{DelegPresVer}(\mathsf{pres}) \xrightarrow{\$} \{0,1\}$
  Parse $\mathsf{pres} \rightarrow (\mathsf{del}, \pi_{\mathsf{del}})$*, where* $\pi_{\mathsf{del}} \rightarrow (\mathsf{pres}'', \sigma'')$ *and* $\mathsf{pres}'' \rightarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta}), \{\mathsf{salt}_i\}_{i\in\Delta_{\mathsf{ID}}}), \mathsf{del}, \mathsf{nonce})$. *The verifier checks that*

  - *the delegation is valid, i.e.* $\mathsf{DelegVer}(\mathsf{del}) \rightarrow 1$*;*
  - π_{del} *is a valid presentation of* cred_Δ *for the attributes in* Δ_{ID} *specified in* del*, i.e.:*
    1. *the signature* σ'' *of* pres'' *is valid using* $\mathsf{pk}_{\mathsf{cred}_\Delta}$*:* $1 \leftarrow \mathsf{Vf}(\sigma'', \mathsf{pres}'', \mathsf{pk}_{\mathsf{cred}_\Delta})$*;*
    2. $\mathsf{com}_i = H(a_i||\mathsf{salt}_i)\, \forall i \in \Delta_{\mathsf{ID}}$*;*
    3. *the signature of the issuer is valid:* $1 \leftarrow \mathsf{Vf}(\sigma, (\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_\Delta}), \mathsf{pk}_{\mathsf{Iss}})$.
  - *the value* scope *included in* del *is satisfied according to the associated verification procedure.*

  If these checks pass, the delegated presentation is accepted, and the algorithm outputs 1.

6. Security Analysis

In this section we prove that the VP delegation scheme described in Definition 6 satisfies the security properties of correctness (Definition 4) and unforgeability (Definition 5).

6.1. Correctness

We prove the following theorem.

Theorem 1. *The delegation scheme described in Definition 6 instantiated using the digital signature $\mathcal{DS}$ is correct, assuming that the digital signature $\mathcal{DS}$ is correct and that H is modeled as a random oracle.*

Proof. It is easy to see that the VP delegation scheme is correct, in fact the delegator always computes a proof π_{DP} to include in del, and since the digital signature $\mathcal{DS}$ is correct del is a valid delegation. Then, if the delegation is presented by a delegatee whose identity matches with the identity Δ_{ID} included in del, the delegatee can always successfully present the delegation, again because $\mathcal{DS}$ is correct. □

6.2. Unforgeability

In the security proof of the unforgeability, we must introduce a technicality which simplifies the description of our reduction. In particular, we show that the adversary $\mathcal{A}$ to the unforgeability of the VP delegation scheme can be used to build a reduction to a *variant* of the unforgeability under chosen message attacks (UF-CMA) of the digital signature $\mathcal{DS}$. In this variant, which we refer to as *v-UF-CMA*, the reduction can open a polynomial number of sessions of standard UF-CMA experiments (each associated with a different public key with the same public parameters pp), and the reduction wins the v-UF-CMA experiment if it produces a forgery for at least one of the public keys it is provided by the challenger of the experiment. Informally, the queries that the adversary can make are queries for the opening of a new session i, for which it receives a public key pk_i, and signing queries specifying a message and one of the public keys it has previously received (m, pk_j), for which it receives a valid signature σ of m under the public key pk_j.

It is easy to see that an adversary who can win the UF-CMA experiment can turn into an adversary of the v-UF-CMA experiment, and the success probability remains the same. However, it is also easy to see that an adversary of v-UF-CMA of $\mathcal{DS}$ can be used as a subroutine for a reduction to the UF-CMA of $\mathcal{DS}$ with a loss in the probability of success proportional to the number of sessions the adversary of v-UF-CMA can open.

6.2.1. Unforgeability Proof

Theorem 2. *The delegation scheme described in Definition 6 is unforgeable according to Definition 5 assuming that the digital signature $\mathcal{DS}$ is (v-UF-CMA) unforgeable, that H is modeled as a random oracle and that the commitment scheme is binding.*

We provide a description of the reduction omitting some details, for example the way the random oracle is programmed. Moreover, since we prove the security in the random oracle model and the commitment is binding [7], the adversary cannot open a commitment to two different values because this would imply that it has found a collision, which happens with negligible probability.

Proof Sketch. We prove that if there exists an adversary $\mathcal{A}$ of Experiment 1 who can win the experiment with non-negligible probability, then we can use $\mathcal{A}$ to build a reduction $\mathcal{B}$ to the v-UF-CMA experiment of the underlying digital signature scheme $\mathcal{DS}$, which wins the experiment with non-negligible probability. The reduction $\mathcal{B}$ interacts with the challenger $\mathcal{C}$ of the v-UF-CMA experiment and simulates the challenger of Experiment 1.

Setup $\mathcal{B}$ sends a query for a new public key in the v-UF-CMA experiment. $\mathcal{C}$ sends to $\mathcal{B}$ a public key pk and public parameters pp. $\mathcal{B}$ sets $\mathsf{pk}_{\mathsf{lss}} \leftarrow \mathsf{pk}$, and forwards $(\mathsf{pp}, \mathsf{pk}_{\mathsf{lss}})$ to the adversary $\mathcal{A}$.

Training phase We show how $\mathcal{B}$ answers to the queries $\mathcal{A}$ can send during the training phase.

[7] The output in bits of the random oracle is sufficiently large

- Queries to $\mathsf{O_{iss}}$: $\mathcal{A}$ sends in input a set of attributes $\{a_i\}_{i\in[l]}$. $\mathcal{B}$ performs the following operations:
  1. it generates a key pair $(\mathsf{sk_{cred}}, \mathsf{pk_{cred}})$, the salts $\{\mathsf{salt}_i\}_{i\in[l]}$, computes a commitment $\mathsf{com}_i = \mathsf{RO}(a_i||\mathsf{salt}_i)$ (where RO is a random oracle programmed by $\mathcal{B}$) to $a_i, \forall i \in [l]$;
  2. it sends a signing query to $\mathcal{C}$ with input $((\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk_{cred}}), \mathsf{pk_{Iss}})$: $\mathcal{C}$ returns a signature σ of $(\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk_{cred}})$ using $\mathsf{sk_{Iss}}$.
  3. sends to $\mathcal{A}$ the credential $\mathsf{cred} \leftarrow ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk_{cred}}), \{a_i\}_{i\in[l]}, \{\mathsf{salt}_i\}_{i\in[l]}, \mathsf{sk_{cred}})$ and adds cred to CT.
- Queries to $\mathsf{O_{del}}$: $\mathcal{A}$ can query for a delegation for the values $(\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP})$ of its choice. Upon receiving a query $\mathcal{B}$ performs the following operations:
  1. generates a set of random attributes $\{a_i\}_{i\in[l]}$ which satisfies DP and computes the commitments com_i as before generating the salts and programming the random oracle;
  2. opens a new session with of UF-CMA with $\mathcal{C}$ from which it receives a public key $\mathsf{pk_{cred}}$ and sends a signing query $((\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk_{cred}}), \mathsf{pk_{Iss}})$ for a signature with $\mathsf{sk_{Iss}}$ of $(\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk_{cred}})$, and receives from $\mathcal{C}$ the signature σ;
  3. sends a signing query $(\mathsf{pres}', \mathsf{pk_{cred}})$ to $\mathcal{C}$ for a signature with $\mathsf{sk_{cred}}$ of $\mathsf{pres}' = ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk_{cred}}), \{\mathsf{salt}_i\}_{i\in\mathsf{DP}}, \mathsf{DP}, \mathsf{scope}, \Delta_{\mathsf{ID}})$. $\mathcal{C}$ returns the signature σ';
  4. $\mathcal{B}$ sets $\pi_{\mathsf{DP}} = (\sigma', \mathsf{pres}')$ sends to $\mathcal{A}$ the delegation $\mathsf{del} = (\Delta_{\mathsf{ID}}, \mathsf{scope}, \mathsf{DP}, \pi_{\mathsf{DP}})$ and adds del to DT.
- Queries to $\mathsf{O_{pres}}$: $\mathcal{A}$ can query for a delegated presentation giving in input $(\mathsf{del}, \mathsf{nonce})$ of its choice. Upon receiving a query $\mathcal{B}$ performs the following operations:
  1. $\mathcal{B}$ generates a credential with the help of $\mathcal{C}$ as it is done for queries to $\mathsf{O_{del}}$ up to Item 2, but instead of choosing the attributes that satisfy DP, they satisfy Δ_{ID} included in del received in input. The credential is associated to the public key $\mathsf{pk_{cred}}$, whose secret counterpart is not known to $\mathcal{B}$;
  2. $\mathcal{B}$ sends a query $(\mathsf{pres}'', \mathsf{pk_{cred}})$ to $\mathcal{C}$ for a signature of $\mathsf{pres}'' = ((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk_{cred}}), \{\mathsf{salt}_i\}_{i\in\Delta_{\mathsf{ID}}}, \mathsf{del}, \mathsf{nonce})$ using $\mathsf{sk_{cred}}$. $\mathcal{C}$ returns σ''.
  3. $\mathcal{B}$ sends $\mathsf{pres} = (\mathsf{del}, (\sigma'', \mathsf{pres}''), \mathsf{nonce})$ to $\mathcal{A}$ and stores pres in PT.

Note that $\mathcal{B}$ knows only the secret keys associated to the credential issued to $\mathcal{A}$ that can not be used in the generation of the forgery.

Forgery Phase $\mathcal{A}$ sends to $\mathcal{B}$ a delegated presentation $\mathsf{pres}^\star = (\mathsf{del}^\star, \pi_{\mathsf{del}^\star})$ as its forgery.

We argue that this presentation is a forgery, i.e. it satisfies at least one of the winning conditions of Experiment 1.

Instantiation and analysis of the winning condition of Experiment 1 We now focus on the winning condition of Experiment 1, and we rewrite it considering the experiment instantiated with the ARF-Compliant VP delegation Scheme in Definition 6.

The adversary's output is a presentation $\mathsf{pres}^\star = (\mathsf{del}^\star, \pi_{\mathsf{del}^\star})$, with $\mathsf{del}^\star = (\Delta_{\mathsf{ID}}{}^\star, \mathsf{scope}^\star, \mathsf{DP}^\star, \pi_{\mathsf{DP}^\star})$. We recall the structure of the proofs $\pi_{\mathsf{DP}^\star}$ and $\pi_{\mathsf{del}^\star}$.

1. $\pi_{\mathsf{DP}^\star}$ is a presentation of a credential for the attributes specified in $\mathsf{DP}^\star$ containing also $\mathsf{scope}^\star$ and $\Delta_{\mathsf{ID}}{}^\star$:

$$\pi_{\mathsf{DP}^\star} = \left(\sigma' = \mathsf{Sign}(\mathsf{pres}', \mathsf{sk_{cred_D}}), \underbrace{((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk_{cred_D}}), \{\mathsf{salt}_i\}_{i\in\mathsf{DP}^\star}, \mathsf{DP}^\star, \mathsf{scope}^\star, \Delta_{\mathsf{ID}}{}^\star)}_{\mathsf{pres}'}\right),$$

with σ' being the signature of pres' using the secret key associated to $\mathsf{pk_{cred_D}}$ and σ is the signature of $(\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk_{cred_D}})$ (part of $\mathsf{cred_D}$) using $\mathsf{sk_{Iss}}$.

2. $\pi^{\star}_{\mathsf{del}}$ is a presentation of a credential cred_{Δ} for the attributes in Δ_{ID} bounded to $\mathsf{del}^{\star}$ and $\mathsf{nonce}^{\star}$, in particular:

$$\pi_{\mathsf{del}^{\star}} = \left(\sigma'' = \mathsf{Sign}(\mathsf{pres}'', \mathsf{sk}_{\mathsf{cred}_{\Delta}}), \underbrace{((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\Delta}}), \{\mathsf{salt}_i\}_{i\in\Delta_{\mathsf{ID}}{}^{\star}}, \mathsf{del}^{\star}, \mathsf{nonce}^{\star})}_{\mathsf{pres}''}\right)$$

with σ'' being the signature of pres'' using the secret key associated to $\mathsf{pk}_{\mathsf{cred}_{\Delta}}$ and σ is the signature of $(\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\Delta}})$ (part of cred_{Δ}) using $\mathsf{sk}_{\mathsf{Iss}}$.

We make the following observation.

Remark 1. *The challenger of the unforgeability experiment (Experiment 1) generates public keys* $\mathsf{pk}_{\mathsf{cred}}$ *and signs them (together with a set of commitments* $\{\mathsf{com}_i\}_{i\in[l]}$*) using* $\mathsf{sk}_{\mathsf{Iss}}$ *as a consequence of:*

1. *issuing queries: in this case, the challenger also gives to the adversary the associated secret key and adds the credential to* CT*;*
2. *delegation queries (delegated presentation queries): in this case, the issuer also signs* pres' *in* π_{DP} *(*pres'' *in* π_{del}*) using the secret key* $\mathsf{sk}_{\mathsf{cred}}$ *associated to* $\mathsf{pk}_{\mathsf{cred}}$*, adds the delegation to* DT (PT)*, and does not reveal* $\mathsf{sk}_{\mathsf{cred}}$ *to the adversary.*

Note that the challenger of Experiment 1 never signs public keys it has not generated [8]

The adversary wins the experiment if $\mathsf{DelegPresVer}(\mathsf{pres}^{\star}) \to 1$ and one of the following conditions is satisfied:

- *forgery of the delegation*: $\mathsf{del}^{\star}$ is forged, i.e.
  - it is not a delegation queried by $\mathcal{A}$ to $\mathsf{O}_{\mathsf{del}}$, i.e. $\mathsf{del}^{\star} \notin \mathsf{DT}$;
  - $\mathsf{del}^{\star}$ was not generated using the credentials issued to $\mathcal{A}$.

  Remark 2. *The public key* $\mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}$ *in* $\mathsf{del}^{\star}$ *has been generated either by the challenger or by* $\mathcal{A}$*. If it is generated by the challenger, since the delegation* $\mathsf{del}^{\star}$ *does not correspond to any credential issued to* $\mathcal{A}$*, then* $\mathcal{A}$ *does not know the secret counterpart* $\mathsf{sk}_{\mathsf{cred}_{\mathsf{D}}}$*. Also, since* $\mathsf{del}^{\star} \notin \mathsf{DT}$*, the challenger has never signed with* $\mathsf{sk}_{\mathsf{cred}_{\mathsf{D}}}$ *the message* $((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}), \{\mathsf{salt}_i\}_{i\in\mathsf{DP}^{\star}}, \mathsf{DP}^{\star}, \mathsf{scope}^{\star}, \Delta_{\mathsf{ID}}{}^{\star})$*, therefore* σ'*, included in* $\pi_{\mathsf{DP}^{\star}}$*, is a forgery of* $\mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}$ [9]*. If* $\mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}$ *is generated by* $\mathcal{A}$ *(who would know the value* $\mathsf{sk}_{\mathsf{cred}_{\mathsf{D}}}$*), then the signature* σ *of* $(\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}})$ *must be a forgery of* $\mathsf{pk}_{\mathsf{Iss}}$*.*

- *forgery of the delegated presentation*: $\pi_{\mathsf{del}^{\star}}$ is forged, i.e.
  - it is not a presentation queried by $\mathcal{A}$, i.e. $\mathsf{pres}^{\star} \notin \mathsf{PT}$;
  - $\pi_{\mathsf{del}^{\star}}$ was not generated using the credentials issued to $\mathcal{A}$.

  Remark 3. *The public key* $\mathsf{pk}_{\mathsf{cred}_{\Delta}}$ *is created either by the challenger or by* $\mathcal{A}$*. If it is created by the challenger,* $\mathcal{A}$ *does not know the associated secret key since* $\pi_{\mathsf{del}^{\star}}$ *is not generated using a credential issued to* $\mathcal{A}$*. But also* $\mathsf{pres}^{\star} \notin \mathsf{PT}$*, therefore it means that the issuer has never signed the message* $((\sigma, \{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\Delta}}), \{\mathsf{salt}_i\}_{i\in\Delta_{\mathsf{ID}}{}^{\star}}, \mathsf{del}^{\star}, \mathsf{nonce}^{\star})$*, therefore* σ'' *(in* $\pi_{\mathsf{del}^{\star}}$*) is a forgery of* $\mathsf{pk}_{\mathsf{cred}_{\Delta}}$*. If* $\mathsf{pk}_{\mathsf{cred}_{\Delta}}$ *is generated by* $\mathcal{A}$*, then* $(\{\mathsf{com}_i\}_{i\in[l]}, \mathsf{pk}_{\mathsf{cred}_{\Delta}})$ *must be a forgery of* $\mathsf{pk}_{\mathsf{Iss}}$*.*

This means that at least one of the public keys that $\mathcal{B}$ has received by $\mathcal{C}$, corresponding to the sessions opened in the v-UF-CMA experiment, has been forged. Therefore, $\mathcal{B}$ would also win the v-UF-CMA experiment, and this concludes the security proof. □

[8] For the sake of clarity, note that in the security proof, the challenger of Experiment 1 is the reduction $\mathcal{B}$. The reduction does not know the secret key $\mathsf{sk}_{\mathsf{Iss}}$ but can ask for signatures of messages of its choice using $\mathsf{sk}_{\mathsf{Iss}}$ sending queries to $\mathcal{C}$, the challenger of the v-UF-CMA experiment. Again, the public keys $\mathsf{pk}_{\mathsf{cred}}$ that get signed using $\mathsf{sk}_{\mathsf{Iss}}$ have been generated by $\mathcal{B}$.

[9] Also σ might be a forgery of $\mathsf{pk}_{\mathsf{Iss}}$ (because the adversary might have changed the set of commitments), but even if it is the case, σ' would be a forgery of $\mathsf{pk}_{\mathsf{cred}_{\mathsf{D}}}$.

7. Instantiation in EBSI and EUDI Frameworks

The protocol we have described and analyzed can be integrated into existing ecosystems such as EBSI or in the EUDI Wallet context without defining new data structures, only new verification procedures. We sketch some considerations describing possible solutions for integrating our scheme into the aforementioned VC frameworks.
The delegation del can be a *VC issued by the delegator* that has as attributes the components we have described, namely $\mathsf{scope}, \Delta_{\mathsf{ID}}, \mathsf{DP}$ and π_{DP}. The delegator signs this credential (as an issuer) using the same secret key used to generate the π_{DP} as specified in Definition 6. This additional signature on the whole delegation is needed for compatibility reasons, because every VC must be signed by the issuer, in this case the delegator. This special credential, which does not require the use of a key pair $(\mathsf{pk}_{\mathsf{cred}}, \mathsf{sk}_{\mathsf{cred}})$ to guarantee the binding to the device that receives it, will be fully disclosed by the delegatee when the time comes to present the delegation.

When the delegatee creates a delegated presentation, it presents del (which is structured as a VC) disclosing all its attributes, namely, $\mathsf{scope}, \Delta_{\mathsf{ID}}, \mathsf{DP}$ and π_{DP}; the delegatee then creates a verifiable presentation π_{del} using its own VC, revealing the attributes included in Δ_{ID}. The outcome of the delegated presentation is derived from the combination of these two presentations. The only modification to the verification protocol is that the verifier must check that π_{DP} is indeed a valid presentation of the statement DP and that the presentation π_{del} created by the delegatee using $\mathsf{cred}_{\mathsf{del}}$ is a valid presentation of Δ_{ID}.

In EBSI, the only entities entitled to issue credentials are legal persons whose DID (a reference to, at least, their public key) is registered in the Trusted Issuer Registry (TIR)[10]. This means that credential holders who are physical persons are not allowed to issue credentials and therefore cannot generate the delegation as we have mentioned above. If the delegator is only a physical person we must consider two cases:

- the delegatee is a legal person: in this case, the delegator can generate the attributes for the delegation VC $\mathsf{scope}, \Delta_{\mathsf{ID}}, \mathsf{DP}$ and π_{DP} and sends them to the delegatee who generates the VC containing this information and signs it in turn. Note that this signature is only useful to guarantee the compatibility and create a VC issued by an entity registered in the TIR. The delegatee can choose not to sign it, which is perfectly fine, but cannot change the information sent by the delegator because π_{DP} is cryptographically bound to $\mathsf{scope}, \Delta_{\mathsf{ID}}$ and DP.
- if the delegatee is also only a physical person, the delegator must have the delegation VC signed by a third party registered in the TIR, which may be the issuer of the credential used to generate the delegation or a third party with this specific role.

Acknowledgements

We thank Giada Sciarretta for the discussion about the application of our protocol in the EUDI framework. Andrea Gangemi is a member of GNSAGA of INdAM. Andrea Flamini, Andrea Gangemi, and Enrico Guglielmino are also members of CrypTO, the group of Cryptography and Number Theory of the Politecnico di Torino. Andrea Flamini acknowledges support from Eustema S.p.A. through the PhD scholarship. Andrea Gangemi and Andrea Flamini are supported by the QUBIP project, funded by the European Union under the Horizon Europe framework program [grant agreement no. 101119746]. Vincenzo Orabona is Blockchain Specialist for Eustema S.p.A and works on large-scale pilots developed by EBSI.

[10] https://hub.ebsi.eu/apis/pilot/trusted-issuers-registry/v4

References

[1] EU, Amendments by the European Parliament to the Commission proposal for a Regulation of the European Parliament and of the Council amending Regulation (EU) no 910/2014 as regards establishing a framework for a European Digital Identity, https://www.europarl.europa.eu/doceo/document/A-9-2023-0038_EN.html, 2023. URL: https://www.europarl.europa.eu/doceo/document/A-9-2023-0038_EN.html.

[2] DG CONNECT, The European Digital Identity Wallet Architecture and Reference Framework, version 1.4.1, 2024. URL: https://eu-digital-identity-wallet.github.io/eudi-doc-architecture-and-reference-framework/.

[3] European Blockchain Services Infrastructure (EBSI), Policies for verifiable credentials presentation and request, 2023. URL: https://hub.ebsi.eu/vc-framework/trust-model/policies.

[4] EU, Consolidated text: Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 (General Data Protection Regulation) (Text with EEA relevance), http://data.europa.eu/eli/reg/2016/679/2016-05-04, 2016. URL: http://data.europa.eu/eli/reg/2016/679/2016-05-04.

[5] C. Baum, O. Blazy, J. Camenisch, J.-H. Hoepman, E. Lee, A. Lehmann, A. Lysyanskaya, R. Mayrhofer, H. Montgomery, N. K. Nguyen, et al., Cryptographers' feedback on the eu digital identity's arf (2024).

[6] AAMVA, Mobile Driver's License (mDL) implementation guidelines, version 1.2, https://www.aamva.org/topics/mobile-driver-license, 2023. URL: https://www.aamva.org/topics/mobile-driver-license.

[7] J. Camenisch, A. Lysyanskaya, A signature scheme with efficient protocols, in: SCN 2002, volume 2576 of *LNCS*, 2002, pp. 268–289. doi:10.1007/3-540-36413-7_20.

[8] A. Flamini, G. Sciarretta, M. Scuro, A. Sharif, A. Tomasi, S. Ranise, On cryptographic mechanisms for the selective disclosure of verifiable credentials, Journal of Information Security and Applications 83 (2024) 103789.

[9] M. Chase, A. Lysyanskaya, On signatures of knowledge, in: Advances in Cryptology-CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings 26, Springer, 2006, pp. 78–96.

[10] J. Camenisch, M. Drijvers, M. Dubovitskaya, Practical uc-secure delegatable credentials with attributes and their application to blockchain, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 683–699.

[11] J. Blömer, J. Bobolz, Delegatable attribute-based anonymous credentials from dynamically malleable signatures, in: International Conference on Applied Cryptography and Network Security, Springer, 2018, pp. 221–239.

[12] E. C. Crites, A. Lysyanskaya, Delegatable anonymous credentials from mercurial signatures, in: Cryptographers' Track at the RSA Conference, Springer, 2019, pp. 535–555.

[13] S. Griffy, A. Lysyanskaya, O. Mir, O. P. Kempner, D. Slamanig, Delegatable anonymous credentials from mercurial signatures with stronger privacy, Cryptology ePrint Archive (2024).

[14] ISO/IEC 18013-5, ISO/IEC 18013-5 personal identification - ISO-compliant driving licence - part 5: Mobile driving licence (mDL) application, 2021. URL: https://www.iso.org/standard/69084.html.

[15] D. Fett, K. Yasuda, B. Campbell, Selective Disclosure for JWTs (SD-JWT), Internet-Draft draft-ietf-oauth-selective-disclosure-jwt-12, Internet Engineering Task Force, 2024. URL: https://datatracker.ietf.org/doc/draft-ietf-oauth-selective-disclosure-jwt/12/, work in Progress.

[16] M. Sporny, D. Longley, D. Chadwick, Verifiable credentials data model, 2022. URL: https://www.w3.org/TR/vc-data-model/.