Formalising Linked-Data based Verifiable Credentials for Selective Disclosure

Dan Yamamoto
Internet Initiative Japan Inc.
Tokyo, Japan
dan@iij.ad.jp

Yuji Suga
Internet Initiative Japan Inc.
Tokyo, Japan
suga@iij.ad.jp

Kazue Sako
Waseda University
Tokyo, Japan
kazuesako@aoni.waseda.jp

***Abstract*—In this paper, we propose a formal definition for Linked-Data based verifiable credential to enable secure selective disclosure among one or multiple verifiable credentials a user has. Previous schemes considered using a single verifiable credential and could not hide the user's identifying information when performing selective disclosure. We propose the first Linked-Data based verifiable credentials that can perform selective disclosure free from the restrictions the previous scheme had, and prove its property. We also discuss a novel use of combining multiple certificates issued by independent issuers to still allow users to perform selective disclosure on the set of credentials. Our scheme has been implemented as an open source Web-based application that generates a verifiable presentation for a given selection of attributes. The performance evaluation is also provided in the paper.**

***Index Terms*—anonymous credentials, verifiable credentials, zero-knowledge proof**

1. Introduction

Current identity management systems are centralised, as there is a central authority who has all the attributes of all the users. On the contrary, the design being discussed at World Wide Web Consortium (W3C) based on Decentralised IDentifier (DID) [39] and Verifiable Credentials (VCs) [40] is gaining attention, as users are in charge of storing their own attributes and can control which of their attributes are to be presented to whom.

In a nutshell, the users are issued with verifiable credentials, which certify the user's set of attributes by a digital signature of the issuer. A verifiable credential can be issued not only to a natural person or legal entity but also to any subject. The adoption of standardised data structures for verifiable credentials has been seen in many practical applications, such as COVID-19 vaccination certificates [18] and the IATA Travel Pass [23].

What is attractive in this design is that we can combine anonymous credentials [11], [12], [15] and users can present verifiable credentials in a selective way. That is, the user can select a portion of the attributes and prove that they are certified by the issuer, without disclosing further information on the credentials, such as the signature or other remaining set of attributes. This property is called selective disclosure. For example, in vaccination credentials, a local government might have issued a certificate with a set of attributes, such as the name of the user and the vaccination dates. The user can present the credential in a way that discloses only the vaccination dates, but not the name, and it can still be verified.

Typically, verifiable credentials are implemented using JSON (JavaScript Object Notation) [8] and applying the digital signature on JSON objects. An example is to use JSON Web Signature (JWS) and JSON Web Token (JWT) [38]. To achieve the selective disclosure property using anonymous credentials, we need the messages to be signed by a special digital signature scheme that is not in the scope of JSON Web Signatures. Therefore, several alternative designs or "flavours" of verifiable credentials have been proposed [25], depending on the message structure and the digital signature scheme applied.

Among those different flavours of verifiable credentials, there are some that use JSON-LD [26] to adopt the concept of Linked Data [3]. By adopting JSON-LD, we can interpret JSON objects according to the publicly available JSON-LD context, which enhances the semantic interoperability of verifiable credentials. However, the current design that uses JSON-LD can offer a selective disclosure property in a limited manner.

Contribution. In this paper, we propose the first verifiable JSON-LD-based credentials that can perform selective disclosure that are free from the restrictions of previous schemes. We also propose a novel use of combining multiple certificates issued by independent issuers, which still allows users to perform selective disclosure on the set of credentials. We further propose a formal definition on the security and privacy of such selective disclosure schemes and prove the property of the proposed scheme. Our scheme has been implemented as open source and provides a Web-based application that generates a verifiable presentation for any given selection of attributes. Performance evaluation is also provided in the paper.

We strongly believe in the potential of applying the concept of Linked Data to verifiable credentials. For example, as shown in Figure 1, user John Smith, identified by the identifier `A`, has a verifiable credential issued by the issuer `I`, which asserts that he works for the company identified by an identifier `B`, such as the Legal Entity Identifier (LEI) [24]. Furthermore, this company has a verifiable credential issued by the issuer `J` that claims that the company is named ABC Inc. and has received the Top 100 award. John can combine these two credentials and claim that he works for a company that received the Top 100 award without disclosing his name, the name

DOI 10.1109/EuroS&PW55150.2022.00013

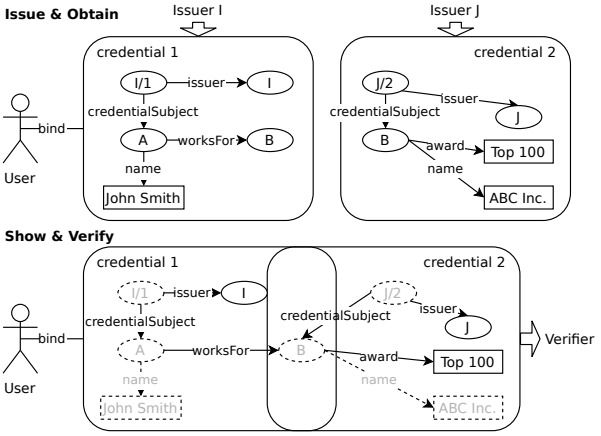

Figure 1. Example of Linked-Data based verifiable credentials

of the company, or any other identifiers included in the credentials.

Note that credential 1, identified by the identifier `I/1` in the figure, is classified as a *bound credential* because the credential is bound to the user secret key and no one can show the credential without such secret key. On the contrary, credential 2, identified by the identifier `J/2`, is called an *unbound credential* because there are no user secret keys bound to the credential and anyone can use it as a public signed document. This is an interesting example of combining bound and unbound credentials to augment attribute presentations of the user. By collecting public unbound credentials from governments, companies, etc., and linking them to private bound credentials, users can attest to various attributes of themselves which could not have been done with bound credentials alone, as well as claims that include the attributes of others. To our knowledge, this type of selective disclosure use case has not been discussed in previous work, and this would open up new possibilities for verifiable credentials.

We note that Fig. 1 exemplifies a graph where the arrows are labelled with informal properties such as `issuer` and `name` for simplicity, while in practice we can use standardised properties such as https://www.w3.org/2018/credentials#issuer from [40] and de facto standard ones such as http://schema.org/name from [36] to enhance semantic interoperability among distinct issuers.

Related work. The BBS+ signature is an extension of the Boneh, Boyen and Shacham (BBS) group signature [5] for anonymous credentials. It is capable of signing sequences of messages and has the features of generating signatures while hiding some messages and efficiently constructing discrete-logarithm-based zero-knowledge proofs for each message. It uses bilinear pairings, and security is shown under the q-SDH (Strong Diffie-Hellman) assumption. Signatures with similar characteristics exist, such as the Pointcheval-Sanders (PS) signature [34] and the Sanders redactable signature [35], but their security is reduced to a less general assumption than the q-SDH assumption.

Most conventional anonymous credentials treat attributes as simple sequences or sets of values. For example, [4], [10]–[12], [35] deals with a sequence of attribute values $(m_\ell)_\ell$, and [20] represents attribute values as a set $\{m_\ell\}_\ell$. For more complex data structures, anonymous credentials based on graph signatures are discussed in [21], [33], [42]. When encoding a graph in the input of a digital signature, Nakanishi et al. [33] use a pairing-based accumulator to achieve proof sizes and verification times that do not depend on the number of points or edges in the graph. However, it is not obvious how to convert Linked Data, the subject of this paper, into general graph information handled by these graph signatures or how to selectively link graphs signed by multiple issuers.

Related standardisation efforts. There are three standardisation efforts at W3C related to our work. The first is Verifiable Credentials Data Model specification [40], which describes a core data model, concepts and syntaxes of verifiable credentials. The specification is maintained by the Verifiable Credentials Working Group. It does not standardise on any single proof mechanism, but instead refers to the second specification effort, Data Integrity.

The specification of Data Integrity [30], which was formerly called Linked Data Proofs, is currently published as a Draft Community Group Report by the W3C Credentials Community Group. It provides specifications to ensure the authenticity and integrity of Linked Data documents, including Linked-Data based verifiable credentials. Data Integrity specification only provides high-level algorithms. Hence, we additionally require a specific canonicalisation algorithm, message digest algorithm, and proof algorithm to be used to obtain the intended result. These are considered in the third effort, LDP-BBS+ specification.

The LDP-BBS+ specification [31] is also a Draft Community Group Report by the W3C Credentials Community Group. It defines specific algorithms to create, verify and derive proofs for BBS+ Signatures [1], [9], [12] with the Data Integrity specification. The construction proposed in our work extends LDP-BBS+ to overcome its limitation and realise the aforementioned use case. While our work has not yet been on any standardisation roadmap, we had some discussions with the editors of the specifications. Our work can be regarded as a candidate for an additional suite in conformance with Data Integrity.

Furthermore, we believe that this work can also be integrated into OpenID Connect for Self-Sovereign Identity specifications [28], [43], [44] discussed in the OpenID Foundation and the Decentralized Identity Foundation (DIF), as well as DIDcomm [16] discussed in DIF.

As for other ongoing standardisation activities related to anonymous credentials, Privacy Pass protocol [17] enables a user to obtain a blindly signed anonymous token from an issuer. The issuer issues anonymous tokens when the user passed some test, for example, CAPTCHAs, and the verifier uses the token for authorisation. This protocol is currently being standardised by the Internet Engineering Task Force (IETF) as Internet-Draft of Privacy Pass Issuance Protocol [13], using the blind RSA signature scheme [14] in publicly verifiable anonymous tokens. In contrast to our work, the scheme intends a different use case where the token is used once, and not multi-use as in our case. It is also unclear if other attributes of the user need to be verified besides passing a test, nor if the property of selective disclosure will be in need.

2. Preliminaries

2.1. Notation

For $a \leq b \in \mathbb{Z}$, $[a, b]$ denotes the set $\{x \in \mathbb{Z} : a \leq x \leq b\}$. We write $[1, n]$ as $[n]$. If $I \in \mathbb{Z}$, denote the sequence $(x_1, \ldots, x_I)$ by $(x_i)_{i\in[I]}$ or $(x_i)_{i=1}^{I}$, and $(x_{i,1}, \ldots, x_{i,J})_{i\in[I]}$ to denote the sequence $((x_{1,1}, \ldots, x_{1,J}), \ldots, (x_{I,1}, \ldots, x_{I,J}))$. Probabilistic Polynomial Time is abbreviated as PPT. Let $a \leftarrow \mathsf{f}(x)$ or $\mathsf{f}(x) \rightarrow a$ denote that the PPT algorithm f with x as input is executed and the output is a. We write $\langle a, b\rangle \leftarrow \langle \mathsf{f}(x), \mathsf{g}(y)\rangle$ or $\langle \mathsf{f}(x), \mathsf{g}(y)\rangle \rightarrow \langle a, b\rangle$ to indicate that the PPT algorithms f and g run interactively with x and y as input, respectively, and their outputs are a and b. We write $a \leftarrow \mathsf{f}(x, y, \ldots; \mathcal{O}_{\mathsf{oracle}_1}, \mathcal{O}_{\mathsf{oracle}_2}, \ldots)$ to indicate the operation of a PPT algorithm f with inputs $x, y, \ldots$ and access to oracles $\mathcal{O}_{\mathsf{oracle}_1}, \mathcal{O}_{\mathsf{oracle}_2}, \ldots$ and let a be the output. We omit the system parameter prm that is entered into the algorithm if it is clear from the context.

2.2. Verifiable Credentials

A verifiable credential is a set of attributes of a subject digitally signed by an issuer. For example, a verifiable credential could serve as a vaccination certificate for a user that is signed by the authority, e.g. a local government, that includes a set of attributes like user's name, address, date of birth, vaccination status together with dates of vaccinations and the names of each vaccine manufacturer.

The World Wide Web Consortium (W3C), in its Verifiable Credentials Working Group, defines the ecosystem of Verifiable Credentials consisting of the following roles [40]: issuer, subject, holder, verifier, and verified data registry, as depicted in Fig. 2.

The issuer issues a verifiable credential containing a set of claims on a Subject, and transfers it to a Holder, who is typically the subject themselves. The holder stores the verifiable credential received, and storage is often called an identity wallet. For simplicity, we use the term "User" to represent both a holder and a subject.

In response to a request from the verifier, the holder retrieves one or more stored verifiable credentials from the identity wallet, extracts only the necessary attributes, and presents them to the verifier in a data set called a verifiable presentation. The verifier uses the issuer's public key to verify that the credentials contained in the verifiable presentation are indeed those of the issuer. In this process, the issuer's identifier and public key are exchanged through a verifiable data registry.

For the implementation of verifiable credentials, several methods have been proposed, including those that enhance user privacy by applying anonymous credentials and those that have features as Linked Data that facilitate linking between multiple data and assigning clear meanings to data.

The verifiable credential is based on the work of the anonymous credential [11], [12], [15] and privacy-enhancing attribute-based credential [10], providing a privacy enhancement mechanism that uses zero-knowledge proofs in addition to digital signatures. For example, it can be used to show that a person is qualified to drive a

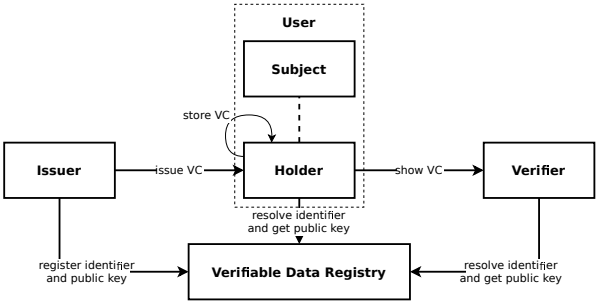

Figure 2. The roles and information flows related to verifiable credentials.

regular car while hiding their name and address or to show that a person is over 20 years old without revealing their date of birth. In addition, the unlinkability of the credential is considered so that the user's actions cannot be tracked between issuers, verifiers, or even between issuers and verifiers.

Verifiable credentials can be represented as JSON-LD [26] document, which has the aspect of Linked Data, linking data to data to form a "web of data" [1]. The verifiable credential described as JSON-LD enables scalable and interoperable data definition and also facilitates linking among multiple verifiable credentials.

The LDP-BBS+ [31] scheme has been proposed as a scheme that takes advantage of the features of anonymous credentials and Linked Data. This scheme applies the BBS+ signature [1], [9], [12] to verifiable credentials based on JSON-LD to allow selective disclosure of attributes. In recent years, the community has been working on standard specifications and test implementations of this method, as it requires less preparation than conventional methods, has a simple structure, and is easy to implement. However, there are some limitations in the scope of anonymisation and zero-knowledge proofs other than selective disclosure cannot be handled. In addition, there is a lack of features to take advantage of the characteristics of Linked Data, such as confidentiality of identifiers contained in credentials and selective disclosure of links between multiple credentials.

2.3. Anonymous Credentials

In this paper, we basically follow the (game-based security) definition of multi-use[2] anonymous credential (AC) from [35] with slight modifications. We modify the definitions to enable simultaneous presentations of multiple credentials and for proofs of equivalence of attributes, as in the simulation-based security definitions of privacy-enhancing attribute-based credential system [10]. Contrary to the previous definitions based on interactive proofs, our definitions are based on non-interactive proofs to suit for applications in concrete data exchange protocols such as OpenID Connect and DIDComm.

Definition 1. An *anonymous credential system* AC consists of the following PPT algorithms:

1. W3C Specification [40] also includes JWT-based verifiable credentials as well as those based on JSON-LD.
2. We require our ACs to support multiple unlinkable showing of credentials, rather than single-use showing such as U-Prove based on Brand's signature scheme [7] and anonymous credentials light [2].

$\mathsf{prmGen}(1^\lambda, L) \to \mathsf{prm}$**:** Given a security parameter 1^λ and an upper bound L for the number of attributes, it outputs public parameters prm.

$\mathsf{ikGen}(\mathsf{prm}) \to (\mathsf{isk}, \mathsf{ipk})$**:** Given public parameters prm, it outputs an issuer's secret key isk and public key ipk.

$\mathsf{sign}(\mathsf{isk}, (m_\ell)_\ell) \to \sigma$**:** Given an issuer secret key isk and attribute values $(m_\ell)_\ell$ to be signed, it outputs a signature σ.

A tuple $(\mathsf{ipk}, (m_\ell)_\ell, \sigma, \mathsf{b} = 0)$ consisting of a public key ipk corresponding to isk, attribute values $(m_\ell)_\ell$, a signature σ, and an unbound indicator $\mathsf{b} = 0$ is called *unbound credential*.

$\mathsf{sigVf}(\mathsf{ipk}, (m_\ell)_\ell, \sigma) \to b$**:** Given an issuer's public key ipk, attribute values $(m_\ell)_\ell$, and a signature σ, it outputs 1 (accept) or 0 (reject).

$\mathsf{uskGen}(\mathsf{prm}) \to \mathsf{usk}$**:** It takes prm as input and outputs a user's secret key usk.

$\langle \mathsf{obtain}(\mathsf{usk}, \mathsf{ipk}, (m_\ell)_\ell), \mathsf{issue}(\mathsf{isk}, (m_\ell)_\ell) \rangle \to \langle \sigma, \top \rangle$**:** It is an interactive protocol run by a user and an issuer, where the user runs obtain with its secret key usk, the issuer's public key ipk, and attribute values $(m_\ell)_\ell$ to be signed, whereas the issuer runs issue with its secret key isk and $(m_\ell)_\ell$. If the protocol is successful, the user gets the signature σ and the issuer gets $\top$. Otherwise, both parties get $\bot$.

We name the issued $(\mathsf{ipk}, (m_\ell)_\ell, \sigma, \mathsf{b} = 1)$ as *bound credential*.

$\mathsf{show}(\mathsf{usk}, (\mathsf{ipk}_i, (m_{i,\ell})_\ell, \sigma_i, \mathcal{D}_i, \mathsf{b}_i)_i, \mathcal{E}, \mathsf{m}) \to \pi$**:** Given one or more pairs of credentials $(\mathsf{ipk}_i, (m_{i,\ell})_\ell, \sigma_i, \mathsf{b}_i)$ and reveal indices $\mathcal{D}_i$, a user's secret key usk, indices $\mathcal{E}$ to specify the target of the equivalence proof, and nonce $\mathsf{m} \in \{0,1\}^*$, it outputs a proof π.

$\mathcal{D}_i$, is an index to specify the $(m'_{i,\ell})_\ell$ to be presented to the verifier from the entire attribute value $(m_{i,\ell})_\ell$.

$\mathcal{E} = \{E_1, E_2, \ldots\}$ is the set of $E \subseteq [I] \times [L]$, which instructs us to prove that $(i,\ell), (i',\ell') \in E\mathcal{E}$ then $m_{i,\ell} = m_{i,\ell'}$ without hiding the actual value. Here, $(i,\ell) \in E \in \mathcal{E}$ implies $\ell \notin \mathcal{D}_i$.

If there exists a pair $(i,\ell), (i',\ell') \in E \in \mathcal{E}$ such that $m_{i,\ell} \neq m_{i',\ell'}$, the algorithm show outputs $\bot$ since in this case $\mathcal{E}$ contains inappropriate indices.

$\mathsf{verify}((\mathsf{ipk}_i, (m'_{i,\ell})_\ell, \mathsf{b}_i)_i, \mathcal{E}, \mathsf{m}, \pi) \to b$**:** Given more than one triple of an issuer's public key ipk_i, revealed attribute values $(m'_{i,\ell})_\ell$, and an indicator b_i of bound / unbound, as well as indices $\mathcal{E}$ to specify the target of equivalence proof, nonce $\mathsf{m} \in \{0,1\}^*$, and the proof π, it outputs 1 (accept) or 0 (reject).

When $\ell \in \mathcal{D}_i$ (subject to disclosure), $m'_{i,\ell} = m_{i,\ell}$, and when $\ell \notin \mathcal{D}_i$ (subject not to disclosure), $m'_{i,\ell} = \bot$ using the non-disclosure symbol $\bot \notin \mathcal{M}$.

Correctness requires that if all I credentials with at most L attributes each are correctly issued to a user by K honest issuers, then any presentation, with selective disclosure and proof of equivalence indicated by $\mathcal{D}_i$ and $\mathcal{E}$, correctly computed by the user from those credentials will be accepted by the verifier. The formal definition is as follows.

Definition 2 (correctness)**.** We define the game $\mathsf{Exp}^{\mathsf{corr}}_{\mathsf{AC}}$ as in Fig. 3. If it always holds that $\mathsf{Exp}^{\mathsf{corr}}_{\mathsf{AC}}(\lambda, I, K, L, (k_i, L_i, (m_{i,\ell})_{\ell=1}^{L_i}, \mathcal{D}_i, \mathsf{b}_i)_{i\in[I]}, \mathcal{E}, \mathsf{m}) = 1$ for any $\lambda \in \mathbb{N}$, $I \geq 1$, $K \geq 1$, $L \geq 1$, $k_i \in [K]$ $(i \in [I])$, $L_i \leq L$, $(m_{i,\ell})_{\ell=1}^{L_i} \subseteq \mathcal{M}^{L_i}$, $\mathcal{D}_i \subseteq [L_i]$, $\mathsf{b}_i \in \{0,1\}$, $\mathcal{E}$ such that $m_{i,\ell} = m_{i,\ell'}$ holds for any $(i,\ell), (i',\ell') \in E$ for all $E \in \mathcal{E}$, $\mathsf{m} \in \{0,1\}^*$, we say that AC is correct.

$\mathsf{Exp}^{\mathsf{corr}}_{\mathsf{AC}}(\lambda, I, K, L, (k_i, L_i, (m_{i,\ell})_{\ell=1}^{L_i}, \mathcal{D}_i, \mathsf{b}_i)_{i\in[I]}, \mathcal{E}, \mathsf{m})$

1: $\mathsf{prm} \leftarrow \mathsf{prmGen}(1^\lambda, L)$; $\mathsf{usk} \leftarrow \mathsf{uskGen}(\mathsf{prm})$
2: **for** $k \in [K]$: $(\mathsf{isk}_k, \mathsf{ipk}_k) \leftarrow \mathsf{ikGen}(\mathsf{prm})$
3: **for** $i \in [I]$:
4: **if** $\mathsf{b}_i : \langle \sigma_i, b_{1,i} \rangle$
$\leftarrow \langle \mathsf{obtain}(\mathsf{usk}, \mathsf{ipk}_{k_i}, (m_{i,\ell})_\ell), \mathsf{issue}(\mathsf{isk}_{k_i}, (m_{i,\ell})_\ell) \rangle$
5: **else** : $\sigma_i \leftarrow \mathsf{sign}(\mathsf{isk}_{k_i}, (m_{i,\ell})_\ell)$; $b_{1,i} \leftarrow \mathsf{sigVf}(\mathsf{ipk}_{k_i}, (m_{i,\ell})_\ell, \sigma_i)$
6: **for** $m_{i,\ell} \in (m_{i,\ell})_\ell$:
7: **if** $\ell \in \mathcal{D}_i : m'_{i,\ell} \leftarrow m_{i,\ell}$ **else** : $m'_{i,\ell} \leftarrow \bot$
8: $\pi \leftarrow \mathsf{show}(\mathsf{usk}, (\mathsf{ipk}_{k_i}, (m_{i,\ell})_\ell, \sigma_i, \mathcal{D}_i, \mathsf{b}_i)_{i\in[I]}, \mathcal{E}, \mathsf{m})$
9: $b_2 \leftarrow \mathsf{verify}((\mathsf{ipk}_i, (m'_{i,\ell})_\ell, \mathsf{b}_i)_{i\in[I]}, \mathcal{E}, \mathsf{m}, \pi)$
10: **return** $b_{1,1} \wedge \cdots \wedge b_{1,I} \wedge b_2$

Figure 3. Correctness game for anonymous credentials

The security notions of an anonymous credential here are not novel; they are slightly modified versions of the definitions in [35]. The differences between ours and [35] are that we adopt a static corruption model for the brevity of the analysis, we add oracles to model the issuance of unbound credentials, and we allow users and adversaries to present multiple credentials at the same time.

Fig. 4 shows the definitions of oracles to be used in the security definitions of anonymous credentials. The $\mathcal{O}_{\mathsf{obtiss}}$ is an oracle that allows an honest issuer and an honest user u to issue and obtain a signature σ for attribute values $(m_\ell)_\ell$. Signature σ is assigned an identifier cid and is recorded in CRED_u (a wallet of credentials issued to the user u) with the public key ipk^* of the issuer, the attribute value $(m_\ell)_\ell$, and a bound/unbound flag b. $\mathcal{O}_{\mathsf{iss}}$ is an oracle that causes an honest issuer to issue a bound signature associated with the adversary for the attribute values $(m_\ell)_\ell$. $\mathcal{O}_{\mathsf{sign}}$ is an oracle that causes an honest issuer to issue an unbound signature for the attribute values $(m_\ell)_\ell$. $\mathcal{O}_{\mathsf{obt}}$ and $\mathcal{O}_{\mathsf{obt}'}$ are oracles that allow an honest user u to obtain bound and unbound signatures, respectively. As in $\mathcal{O}_{\mathsf{obtiss}}$, the signature and other information is recorded in CRED_u, which is associated with the user u. $\mathcal{O}_{\mathsf{show}}$ is an oracle that causes an honest user u to present their credentials. It specifies one or more credentials issued to u by $\mathcal{O}_{\mathsf{obtiss}}$, $\mathcal{O}_{\mathsf{obt}}$ and $\mathcal{O}_{\mathsf{obt}'}$ by the identifier cid_i, and performs selective disclosure based on reveal indices $\mathcal{D}_i$ and proof of equivalence based on the equivalence index $\mathcal{E}$ using the nonce m. Information on the disclosed credentials is recorded in PRES with the nonce.

We define the unforgeability of anonymous credentials as follows.

Definition 3 (Unforgeability)**.** We say that AC is *unforgeable* if $\Pr\left[\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{AC}}(\lambda, L, \mathcal{A}) = 1\right]$ is negligible for any $\lambda \in \mathbb{N}$, $L \geq 1$ and adversary of the PPT $\mathcal{A}$, where $\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{AC}}$ is defined in Fig. 5.

With an unforgeable anonymous credential, for a honest issuer with $(\mathsf{isk}^*, \mathsf{ipk}^*)$ and a group of honest users with $((\mathsf{usk}^*_u)_{u\in[U]})$, the adversary cannot make a forgery $((\mathsf{ipk}_i, (m_\ell)_{\ell\in\mathcal{D}_i}, \mathcal{D}_i, \mathsf{b}_i)_{i\in[I^*]}, \mathcal{E}^*, \mathsf{m}^*, \pi^*)$ that is accepted by the verifier (b^*_1). Here, the forgery must be

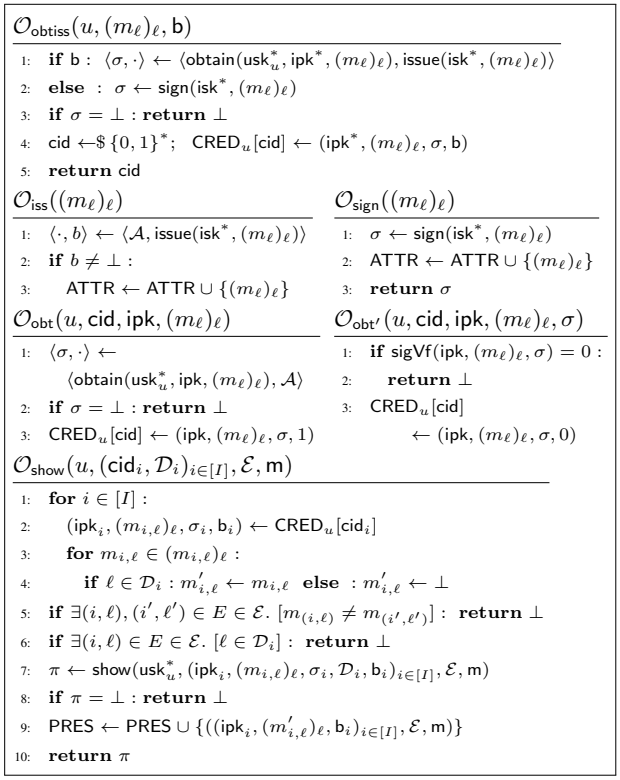

Figure 4. Oracles to be used in the security definitions of anonymous credentials

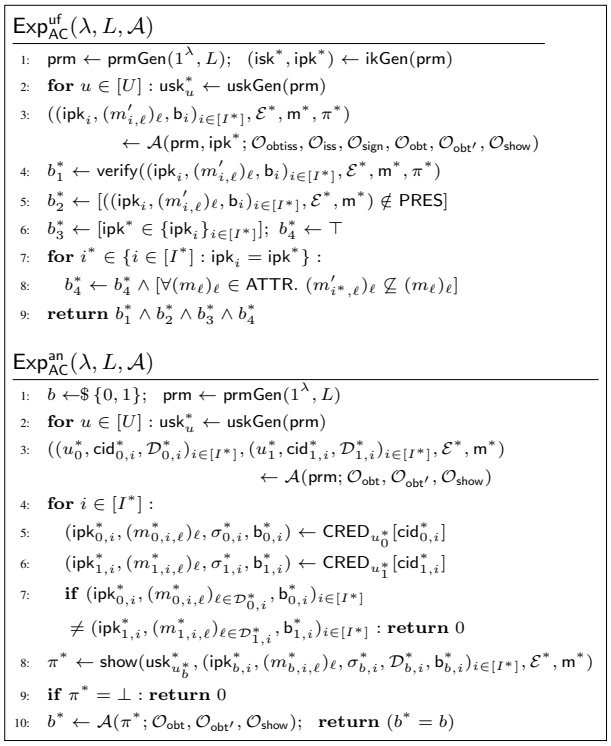

Figure 5. Unforgeability and anonymity of anonymous credentials

nontrivial, that is, the forgery must not have been created by a $\mathcal{O}_{\mathsf{show}}$ oracle (b^*_2), and must contain at least one credential issued by an honest issuer (b^*_3). Furthermore, forgeries based on those honest credentials must not be derivable from the credentials obtained by the adversary via the $\mathcal{O}_{\mathsf{iss}}$ or $\mathcal{O}_{\mathsf{sign}}$ oracles (b^*_4). We say that $(m'_\ell)_\ell \subseteq (m_\ell)_\ell$ if $m'_\ell\neq\bot\Rightarrow m'_\ell=m_\ell$ is valid for all ℓ.

We define the anonymity of anonymous credentials as follows:

Definition 4 (Anonymity)**.** We say that AC is *anonymous* if $\Pr[\mathsf{Exp}^{\mathsf{an}}_{\mathsf{AC}}(\lambda,L,\mathcal{A})=1]$ is negligible for any $\lambda\in\mathbb{N}$, $L\geq 1$ and PPT adversary $\mathcal{A}$, where $\mathsf{Exp}^{\mathsf{an}}_{\mathsf{AC}}$ is defined in Fig. 5.

The above definition means that any PPT adversary cannot extract any knowledge from the proof π^* that helps identify who presents π^* and which credentials are used to construct π^*, except to the extent that it is trivially identified from the revealed attributes and the issuers' public keys.

For example, we can construct an anonymous credential scheme with the above notions of unforgeability and anonymity, based on BBS + signatures [1], [9], [12] with the Fiat-Shamir heuristic [19], which is proven secure under the q-SDH assumption in the random oracle model.

3. Linked-Data based Verifiable Credentials

3.1. RDF Graph

In this paper, we follow [26] and refer to a set of documents, each containing a representation of an RDF graph [37] as Linked Data. Note that verifiable credentials based on JSON-LD can be uniquely deserialised to RDF graphs according to the standardised algorithm [27]. An RDF graph is a representation of a labelled directed graph as a set of triples. Specifically, for a message set $\mathcal{M}$, the set of three terms called subject, predicate, and object $(s,p,o)\in\mathcal{M}^3$ is called an RDF triple, and the set of RDF triples $G\subset\mathcal{M}^3$ is called an RDF graph. For simplicity, we assume that each term is an element of $\mathcal{M}$ and ignore the difference between IRIs and literals, and we do not deal with blank nodes.

The two credentials shown in Figure 1 are examples of RDF graphs. For example, credential 1 can be represented as the RDF graph G_1 shown below.

$$
\begin{aligned}
G_1 = \{(s_i,p_i,o_i) : 1\leq i\leq 4\},\ &\text{where}\\
&(s_1,p_1,o_1) = (\texttt{I/1},\ \texttt{issuer},\ \texttt{I})),\\
&(s_2,p_2,o_2) = (\texttt{I/1},\ \texttt{credentialSubject},\ \texttt{A}),\\
&(s_3,p_3,o_3) = (\texttt{A},\ \texttt{name},\ \texttt{John Smith}),\\
&(s_4,p_4,o_4) = (\texttt{A},\ \texttt{worksFor},\ \texttt{B})
\end{aligned}
$$

Hereinafter, we refer to RDF graphs and RDF triples simply as graphs and triples if it does not cause misunderstanding.

We use a canonicalisation algorithm canon, a deterministic algorithm that converts a graph $G=\{(s_j,p_j,o_j)\}_{j\in\mathcal{J}}$, an *unordered* set of triples, into the *ordered* set $\vec{G}=(s_j,p_j,o_j)_{j\in[|G|]}$. Since we assumed that the RDF graphs in this paper do not contain blank nodes, canonicalisation is trivially possible. For example, serialising each element $s,p,o\in\mathcal{M}$ by any appropriate serialisation rule and sorting them in lexicographic order is

sufficient.[3] For the above example of G_1, its canonicalised form is obtained as follows:

$$\begin{aligned}\mathsf{canon}(G_1) = ((&\texttt{A, name, John Smith}),\\ &(\texttt{A, worksFor, B}),\\ &(\texttt{I/1, credentialSubject, A}),\\ &(\texttt{I/1, issuer, I}))\end{aligned}$$

3.2. Reveal Function

Our verifiable credentials based on Linked Data allow users to selectively disclose their RDF graphs. Specifically, we introduce the following reveal function φ to express the operation of deriving a partially hidden graph G' from an RDF graph G as $G' = \varphi(G)$.

Let φ be a function that assigns the RDF triple $(s, p, o) \in \mathcal{M}^3$ to the empty set $\emptyset$ or (s', p', o') with zero or more terms replaced by *mask* $X \in \mathcal{X}$, where $s' \in \{s\} \cup \mathcal{X}$, $p' \in \{p\} \cup \mathcal{X}$, $o' \in \{o\} \cup \mathcal{X}$. Here, $\mathcal{X} = \{X_1, X_2, \ldots\}$ is a set of masks that is used to hide the labels of the vertices and arcs on the graph, which satisfies $\mathcal{X} \cap \mathcal{M} = \emptyset$. If the same mask appears in several places in the revealed graph G', this means that the attribute values are originally equal and their equivalence is later shown by the user in a zero-knowledge proof.

We say that $G' \sqsubseteq G$ if for any $(s', p', o') \in G'$ there exists some $(s, p, o) \in G$ such that $s' \in \{s\} \cup \mathcal{X}$, $p' \in \{p\} \cup \mathcal{X}$, and $o' \in \{o\} \cup \mathcal{X}$ hold.

A reveal function is called *incorrect* if it attempts to show the equivalence of non-equivalent values. More strictly, for a given $\varphi_i, \varphi_{i'} \in (\varphi_i)_i$, for any $t_j \in \mathcal{M}$ and $x_j \in \mathcal{M} \cup \mathcal{X}$ $(1 \le j \le 6)$ such that $\varphi_i((t_1, t_2, t_3)) = (x_1, x_2, x_3)$ and $\varphi_{i'}((t_4, t_5, t_6)) = (x_4, x_5, x_6)$, if both $t_j \neq t_{j'}$ and $x_j = x_{j'}$ hold for some $1 \le j, j' \le 6$, then we say that $(\varphi_i)_i$ is an incorrect set of reveal functions.

For example, for credential 1 in Figure 1, the following reveal function φ_1 specifies that the vertices `I/1` and `A` are to be masked by X_1 and X_2, respectively, and that the arc `name` with value `John Smith` will be deleted.

$$\begin{aligned}&\varphi_1((\texttt{I/1, issuer, I})) := (X_1, \texttt{issuer, I}),\\ &\varphi_1((\texttt{I/1, credentialSubject, A}))\\ &\qquad := (X_1, \texttt{credentialSubject}, X_2),\\ &\varphi_1((\texttt{A, name, John Smith})) := \emptyset,\\ &\varphi_1((\texttt{A, worksFor, B})) := (X_2, \texttt{worksFor, B})\end{aligned}$$

For an RDF graph G, we denote the set $\{\varphi((s,p,o))\}_{(s,p,o)\in G}$ by $\varphi(G)$. For a sequence $\vec{G} = (s_i, p_i, o_i)_i$ of RDF triples, we denote the sequence $(\varphi((s_i, p_i, o_i))_i = (\varphi((s_1, p_1, o_1)), \varphi((s_2, p_2, o_2)), \ldots)$ as $\varphi(\vec{G})$.

3.3. Syntax

Definition 5. A Linked-Data based verifiable credential system LDVC consists of the following PPT algorithms:

$\mathsf{prmGen}(1^\lambda, L) \to \mathsf{prm}$**:** Given a security parameter 1^λ and an upper bound L for the number of arcs in a graph, it outputs public parameters prm.

$\mathsf{ikGen}(\mathsf{prm}) \to (\mathsf{isk}, \mathsf{ipk})$**:** Given public parameters prm, it outputs an issuer's secret key isk and public key ipk.

$\mathsf{sign}(\mathsf{isk}, G) \to \sigma$**:** Given an issuer secret key isk and a graph G to be signed, it outputs a signature σ.
A tuple $(\mathsf{ipk}, G, \sigma, \mathsf{b} = 0)$ consisting of a public key ipk corresponding to isk, a graph G, a signature σ, and an unbound indicator $\mathsf{b} = 0$ is called an *unbound credential*.

$\mathsf{sigVf}(\mathsf{ipk}, G, \sigma) \to b$**:** Given an issuer's public key ipk, a graph G, and a signature σ, it outputs 1 (accept) or 0 (reject).

$\mathsf{uskGen}(\mathsf{prm}) \to \mathsf{usk}$**:** It takes prm as input and outputs a user's secret key usk.

$\langle \mathsf{obtain}(\mathsf{usk}, \mathsf{ipk}, G), \mathsf{issue}(\mathsf{isk}, G) \rangle \to \langle \sigma, \top \rangle$**:** It is an interactive protocol run by a user and an issuer, where the user runs obtain with its secret key usk, the issuer's public key ipk and a graph G to be signed, while the issuer runs issue with its secret key isk and G. If the protocol is successful, the user gets the signature σ and the issuer gets $\top$. Otherwise, both parties get $\bot$.
We name the issued $(\mathsf{ipk}, G, \sigma, \mathsf{b} = 1)$ as *bound credential*.

$\mathsf{show}(\mathsf{usk}, (\mathsf{ipk}_i, G_i, \sigma_i, \varphi_i, \mathsf{b}_i)_i, \mathsf{m}) \to \pi$**:** Given one or more pairs of credentials $(\mathsf{ipk}_i, G_i, \sigma_i, \mathsf{b}_i)$ and reveal functions φ_i, a user's secret key usk, and nonce $\mathsf{m} \in \{0,1\}^*$, it outputs a proof π for the derived graphs $(\varphi_i(G_i))_i$.
It outputs $\bot$ if $(\varphi_i)_i$ is an incorrect set of reveal functions.

$\mathsf{verify}((\mathsf{ipk}_i, G'_i, \mathsf{b}_i)_i, \mathsf{m}, \pi) \to b$**:** Given more than one triple of an issuer's public key ipk_i, a derived graph G'_i, and an indicator b_i of bound / unbound, as well as nonce $\mathsf{m} \in \{0,1\}^*$ and the proof π, it outputs 1 (accept) or 0 (reject).

Since the correctness of LDVC is almost the same as the correctness of an anonymous credential, the exact definition is omitted.

3.4. Security

We define the unforgeability and anonymity of LDVC based on the corresponding notions of anonymous credentials by replacing a sequence $(m_\ell)_\ell$ of attributes in an anonymous credential with an RDF graph G.

With the oracle definitions in Fig. 6, the unforgeability and anonymity of LDVC are defined as follows:

Definition 6 (Unforgeability)**.** We say that LDVC is *unforgeable* if $\Pr\left[\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{LDVC}}(\lambda, L, \mathcal{A}) = 1\right]$ is negligible for any $\lambda \in \mathbb{N}$, $L \ge 1$, and PPT adversary $\mathcal{A}$, where $\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{LDVC}}$ is defined in Fig. 7.

Definition 7 (Anonymity and weak anonymity)**.** We say that LDVC is *anonymous* and *weakly anonymous* if $\Pr[\mathsf{Exp}^{\mathsf{an}}_{\mathsf{LDVC}}(\lambda, L, \mathcal{A}) = 1]$ and $\Pr[\mathsf{Exp}^{\mathsf{wan}}_{\mathsf{LDVC}}(\lambda, L, \mathcal{A}) = 1]$ are negligible for any $\lambda \in \mathbb{N}$, $L \ge 1$ and PPT adversary $\mathcal{A}$, where the experiments are defined in Fig. 7, respectively.

Although the unforgeability and anonymity of LDVC are obtained as straightforward extensions of the anonymous credential, we additionally define the notion of *weak* anonymity for LDVC. Informally speaking, (full) anonymity ensures that no PPT adversary can identify the

3. For canonicalisation of RDF graphs *with* blank nodes, there exist several proposed methods including [29] and [22].

$\mathcal{O}_{\mathsf{obtiss}}(u, G, \mathsf{b})$

1: **if** b : $\langle \sigma, \cdot \rangle \leftarrow \langle \mathsf{obtain}(\mathsf{usk}^*_u, \mathsf{ipk}^*, G), \mathsf{issue}(\mathsf{isk}^*, G) \rangle$
2: **else** : $\sigma \leftarrow \mathsf{sign}(\mathsf{isk}^*, G)$
3: **if** $\sigma = \bot$: **return** $\bot$
4: $\mathsf{cid} \leftarrow\!\$\, \{0,1\}^*$; $\mathsf{CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}^*, G, \sigma, \mathsf{b})$
5: **return** cid

$\mathcal{O}_{\mathsf{iss}}(G)$

1: $\langle \cdot, b \rangle \leftarrow \langle \mathcal{A}, \mathsf{issue}(\mathsf{isk}^*, G) \rangle$
2: **if** $b \neq \bot$:
3: $\mathsf{ATTR} \leftarrow \mathsf{ATTR} \cup \{G\}$

$\mathcal{O}_{\mathsf{sign}}(G)$

1: $\sigma \leftarrow \mathsf{sign}(\mathsf{isk}^*, G)$
2: $\mathsf{ATTR} \leftarrow \mathsf{ATTR} \cup \{G\}$
3: **return** σ

$\mathcal{O}_{\mathsf{obt}}(u, \mathsf{cid}, \mathsf{ipk}, G)$

1: $\langle \sigma, \cdot \rangle \leftarrow$
$\langle \mathsf{obtain}(\mathsf{usk}^*_u, \mathsf{ipk}, G), \mathcal{A} \rangle$
2: **if** $\sigma = \bot$: **return** $\bot$
3: $\mathsf{CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}, G, \sigma, 1)$

$\mathcal{O}_{\mathsf{obt}'}(u, \mathsf{cid}, \mathsf{ipk}, G, \sigma)$

1: **if** $\mathsf{sigVf}(\mathsf{ipk}, G, \sigma) = 0$:
2: **return** $\bot$
3: $\mathsf{CRED}_u[\mathsf{cid}]$
$\leftarrow (\mathsf{ipk}, G, \sigma, 0)$

$\mathcal{O}_{\mathsf{show}}(u, (\mathsf{cid}_i, \varphi_i)_{i \in [I]}, \mathsf{m})$

1: **if** $(\varphi_i)_{i \in [I]}$ is incorrect : **return** $\bot$
2: **for** $i \in [I]$:
3: $(\mathsf{ipk}_i, G_i, \sigma_i, \mathsf{b}_i) \leftarrow \mathsf{CRED}_u[\mathsf{cid}_i]$; $G'_i \leftarrow \varphi_i(G_i)$
4: $\pi \leftarrow \mathsf{show}(\mathsf{usk}^*_u, (\mathsf{ipk}_i, G_i, \sigma_i, \varphi_i, \mathsf{b}_i)_{i \in [I]}, \mathsf{m})$
5: **if** $\pi = \bot$: **return** $\bot$
6: $\mathsf{PRES} \leftarrow \mathsf{PRES} \cup \{((\mathsf{ipk}_i, G'_i, \mathsf{b}_i)_{i \in [I]}, \mathsf{m})\}$
7: **return** π

Figure 6. Oracles to be used in the security definitions of LDVC

user presenting the proof π^* or learn anything about the user, except to the extent that it is trivially learnt from the revealed graphs and the issuers' public keys required to verify the proof. Regarding *weak* anonymity, we further extend this "trivially learnt" information; an adversary in the *weak* anonymity setting is able to additionally learn *the sizes and canonicalised forms of the graphs at the time of issuance* as well as the revealed graphs and the issuers' public keys.

4. Construction of Linked-Data based Verifiable Credential

4.1. Construction

Now we construct our LDVC from an anonymous credential system. Our construction can be seen as an extension of the LDP-BBS+ scheme [31]. The original LDP-BBS + scheme uses the Universal RDF Dataset Canonicalisation Algorithm 2015 (URDNA2015) [29] to convert an RDF graph into a canonicalised (ordered) list of RDF triples, and then generates and verifies a BBS+ signature taking each *RDF triple* as input attribute. As a result, zero-knowledge proofs in the scheme can only be performed for each triple, so that selective disclosure, range proofs, and any other ZKPs for individual values (i.e., ID, name, date of birth, etc.) cannot be realised.

However, in our construction, RDF triples are further decomposed into RDF terms, that is, subject, predicate, and object, which are regarded as attributes for generating and verifying signatures. This enables zero-knowledge proofs, such as selective disclosure, range proofs, and proof of equivalent attributes for individual *values* rather than triples. In particular, using the proof of equality, we

$\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{LDVC}}(\lambda, L, \mathcal{A})$

1: $\mathsf{prm} \leftarrow \mathsf{prmGen}(1^\lambda, L)$; $(\mathsf{isk}^*, \mathsf{ipk}^*) \leftarrow \mathsf{ikGen}(\mathsf{prm})$
2: **for** $u \in [U]$: $\mathsf{usk}^*_u \leftarrow \mathsf{uskGen}(\mathsf{prm})$
3: $((\mathsf{ipk}_i, G'_i, \mathsf{b}_i)_{i \in [I^*]}, \mathsf{m}^*, \pi^*)$
$\leftarrow \mathcal{A}(\mathsf{prm}, \mathsf{ipk}^*; \mathcal{O}_{\mathsf{obtiss}}, \mathcal{O}_{\mathsf{iss}}, \mathcal{O}_{\mathsf{sign}}, \mathcal{O}_{\mathsf{show}})$
4: $b^*_1 \leftarrow \mathsf{verify}((\mathsf{ipk}_i, G'_i, \mathsf{b}_i)_{i \in [I^*]}, \mathsf{m}^*, \pi^*)$
5: $b^*_2 \leftarrow [((\mathsf{ipk}_i, G'_i, \mathsf{b}_i)_{i \in [I^*]}, \mathsf{m}^*) \notin \mathsf{PRES}]$
6: $b^*_3 \leftarrow [\mathsf{ipk}^* \in \{\mathsf{ipk}_i\}_{i \in [I^*]}]$; $b^*_4 \leftarrow \top$
7: **for** $i^* \in \{i \in [I^*] : \mathsf{ipk}_i = \mathsf{ipk}^*\}$:
8: $b^*_4 \leftarrow b^*_4 \wedge [\forall G \in \mathsf{ATTR}.\ G'_{i^*} \not\sqsubseteq G]$
9: **return** $b^*_1 \wedge b^*_2 \wedge b^*_3 \wedge b^*_4$

$\mathsf{Exp}^{\mathsf{an}}_{\mathsf{LDVC}}(\lambda, L, \mathcal{A})$ $\mathsf{Exp}^{\mathsf{wan}}_{\mathsf{LDVC}}(\lambda, L, \mathcal{A})$

1: $b \leftarrow\!\$\, \{0,1\}$; $\mathsf{prm} \leftarrow \mathsf{prmGen}(1^\lambda, L)$
2: **for** $u \in [U]$: $\mathsf{usk}^*_u \leftarrow \mathsf{uskGen}(\mathsf{prm})$
3: $((u^*_0, \mathsf{cid}^*_{0,i}, \varphi^*_{0,i})_{i \in [I^*]}, (u^*_1, \mathsf{cid}^*_{1,i}, \varphi^*_{1,i})_{i \in [I^*]}, \mathsf{m}^*)$
$\leftarrow \mathcal{A}(\mathsf{prm}; \mathcal{O}_{\mathsf{obt}}, \mathcal{O}_{\mathsf{obt}'}, \mathcal{O}_{\mathsf{show}})$
4: **for** $i \in [I^*]$:
5: $(\mathsf{ipk}^*_{0,i}, G^*_{0,i}, \sigma^*_{0,i}, \mathsf{b}^*_{0,i}) \leftarrow \mathsf{CRED}_{u^*_0}[\mathsf{cid}^*_{0,i}]$
6: $(\mathsf{ipk}^*_{1,i}, G^*_{1,i}, \sigma^*_{1,i}, \mathsf{b}^*_{1,i}) \leftarrow \mathsf{CRED}_{u^*_1}[\mathsf{cid}^*_{1,i}]$
7: **if** $(\mathsf{ipk}^*_{0,i}, \varphi^*_{0,i}(G^*_{0,i}), \mathsf{b}^*_{0,i})_{i \in [I^*]}$
$\neq (\mathsf{ipk}^*_{1,i}, \varphi^*_{1,i}(G^*_{1,i}), \mathsf{b}^*_{1,i})_{i \in [I^*]}$: **return** 0
8: **if** $(\mathsf{ipk}^*_{0,i}, \varphi^*_{0,i}(\mathsf{canon}(G^*_{0,i})), |G^*_{0,i}|, \mathsf{b}^*_{0,i})_{i \in [I^*]}$
$\neq (\mathsf{ipk}^*_{1,i}, \varphi^*_{1,i}(\mathsf{canon}(G^*_{1,i})), |G^*_{1,i}|, \mathsf{b}^*_{1,i})_{i \in [I^*]}$: **return** 0
9: $\pi^* \leftarrow \mathsf{show}(\mathsf{usk}^*_{u^*_b}, (\mathsf{ipk}^*_{b,i}, G^*_{b,i}, \sigma^*_{b,i}, \varphi^*_{b,i}, \mathsf{b}^*_{b,i})_{i \in [I^*]}, \mathsf{m}^*)$
10: **if** $\pi^* = \bot$: **return** 0
11: $b^* \leftarrow \mathcal{A}(\pi^*; \mathcal{O}_{\mathsf{obt}}, \mathcal{O}_{\mathsf{obt}'}, \mathcal{O}_{\mathsf{show}})$; **return** $(b^* = b)$

Figure 7. Unforgeability and anonymity of LDVC. Highlighted parts are only evaluated in the weak anonymity game.

can combine multiple RDF graphs while hiding the URIs in the graphs, enabling the use cases shown in Figure 1.

Note that the selectively disclosed graph, passed from the user to the verifier, can be different from the original issued graph by the issuer, that is, some of the triples in the graph might be removed or some of the terms in the triples might be replaced by masks. These redactions can differ the result of canonicalisation, that is, the *order* of the canonicalised RDF triples by the verifier can be different from that of the issuer, leading to verification failure. To address this problem, in our construction, we let the user calculate a map ψ between $(\tilde{m}_\ell)_\ell$ at verification (verify) and $(m_\ell)_\ell$ at issuance (sign or issue), which is delivered to the verifier so that the verifier can reproduce the same attribute order as at the time of issuance.

Figure 8 shows our LDVC construction $\mathsf{LDVC}_{\mathsf{AC}}$ using AC as its building block.

flatten is a function that takes a sequence $\vec{G} = ((s_1, p_1, o_1), (s_2, p_2, o_2), \ldots)$ of triples as input to obtain a flat sequence $(m_\ell)_\ell = (s_1, p_1, o_1, s_2, p_2, o_2, \ldots)$.

mapGen is a function that takes sequences $(x_i)_{i=1}^I$ and $(y_j)_{j=1}^J$ as input to output a map $\psi : [I] \to [J]$ such that $x_i = y_{\psi(i)}$ is satisfied.

reorder uses ψ to reconstruct a sequence $(y_j)_{j=1}^J$ from $(x_i)_{i=1}^I$, where the missing elements are filled with the empty set $\emptyset$.

Masks $X_1, X_2, \ldots$ specified by φ are used in the computation of the equivalence proof indices $\mathcal{E}$. The attribute values assigned to the same mask are shown to be equivalent by a zero-knowledge proof of AC.

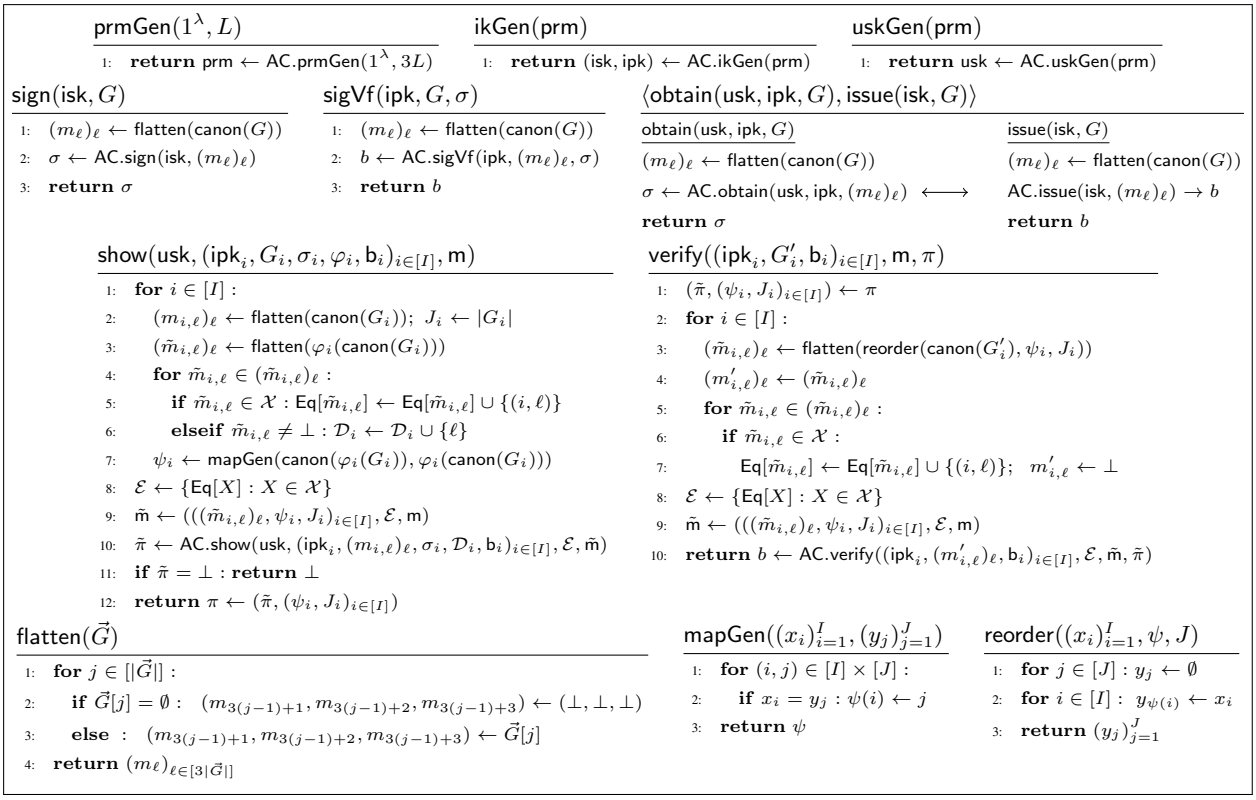

Figure 8. Generic construction $\mathsf{LDVC}_{\mathsf{AC}}$ of Linked-Data based Verifiable Credential

Eq is a dictionary-like data structure that maps a mask $X \in \mathcal{X}$ to a set of (i, ℓ)'s, each of which corresponds to an attribute $m_{i,\ell}$ and these $m_{i,\ell}$'s are masked by X in the derived graph G'. For example, if we have $\mathsf{Eq}[X_1] = \{(1,2),(2,3)\}$ then both $m_{1,2}$ in G'_1 and $m_{2,3}$ in G'_2 have the mask X_1 as their values. Note that $\{\mathsf{Eq}[X] : X \in \mathcal{X}\} = \{\mathsf{Eq}[X_1], \mathsf{Eq}[X_2], \ldots\}$, where each $\mathsf{Eq}[X_i]$ is a set of (i, ℓ)'s masked by X_i's.

4.2. Security

Theorem 1. $\mathsf{LDVC}_{\mathsf{AC}}$ *is unforgeable if* AC *is unforgeable.*

Proof. Assume that we have a PPT adversary $\mathcal{A}$ against the unforgeability of $\mathsf{LDVC}_{\mathsf{AC}}$. We will construct an adversary $\mathcal{B}$ that exploits $\mathcal{A}$ to break the unforgeability of AC. Since we assume AC to be unforgeable, showing the following relation is sufficient to conclude this proof.

$$\Pr\left[\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{LDVC}_{\mathsf{AC}}}(\lambda, L, \mathcal{A}) = 1\right] \leq \Pr\left[\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{AC}}(\lambda, L, \mathcal{B}) = 1\right] \tag{1}$$

To show how to construct $\mathcal{B}$ that meets Eq. (1), we define consecutive games described in Figure 11.

Game G_0**.** The initial game $\mathsf{G}_0(\lambda, L, \mathcal{A})$ is equivalent to $\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{LDVC}_{\mathsf{AC}}}(\lambda, L, \mathcal{A})$. Therefore, we have the following.

$$\Pr\left[\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{LDVC}_{\mathsf{AC}}}(\lambda, L, \mathcal{A}) = 1\right] = \Pr[\mathsf{G}_0(\lambda, L, \mathcal{A}) = 1]$$

Game G_1**.** We introduce three variables $\mathsf{AC.CRED}_u$, AC.ATTR, and AC.PRES to record some values during the processing of $\mathcal{O}_{\mathsf{obtiss}}$, $\mathcal{O}_{\mathsf{iss}}$, $\mathcal{O}_{\mathsf{sign}}$, and $\mathcal{O}_{\mathsf{show}}$.

With the processing added to line 13 in $\mathcal{O}_{\mathsf{show}}$, the tuple $(\mathsf{ipk}_i, (m_{i,\ell})_\ell, \sigma_i, \mathsf{b}_i)$ will be overwritten with the values recorded in $\mathsf{AC.CRED}_u[\mathsf{cid}_i]$. Note that ipk_i, σ_i, and b_i are the same in both $\mathsf{AC.CRED}_u[\mathsf{cid}]$ and $\mathsf{CRED}_u[\mathsf{cid}]$. Furthermore, we see that $(m_{i,\ell})_\ell$ is also generated as $(m_{i,\ell})_\ell = \mathsf{flatten}(\mathsf{canon}(G))$ in the lines 1 of $\mathcal{O}_{\mathsf{obtiss}}$, which is completely the same as the way in $\mathcal{O}_{\mathsf{show}}$. Therefore, these additional operations will not affect the game's output.

Although we introduce a variable $m'_{i,\ell}$ on the line 15 of $\mathcal{O}_{\mathsf{show}}$, this does not affect the game output.

In lines 16 and 17 of $\mathcal{O}_{\mathsf{show}}$, we let the oracle return $\perp$ when $\mathcal{E}$ satisfies certain conditions, but note that these conditions can never be true if $(\varphi_i)_{i\in[I]}$ are correct reveal functions. Since the validity of $(\varphi_i)_{i\in[I]}$ is guaranteed on the line 1 of $\mathcal{O}_{\mathsf{show}}$, these additional processes do not affect the game output.

From the above, we have $\Pr[\mathsf{G}_0(\lambda, L, \mathcal{A}) = 1] = \Pr[\mathsf{G}_1(\lambda, L, \mathcal{A}) = 1]$.

Game G_2**.** We replace the game output with four additional variables b^{**}_1, b^{**}_2, b^{**}_3, and b^{**}_4. We will show that $\bigwedge \mathsf{b}^{*}_i$ implies $\bigwedge \mathsf{b}^{**}_i$, from which we have $\Pr[\mathsf{G}_1(\lambda, L, \mathcal{A}) = 1] \leq \Pr[\mathsf{G}_2(\lambda, L, \mathcal{A}) = 1]$.

We can see that b^{*}_1 and b^{*}_3 are equivalent to b^{**}_1 and b^{**}_3, respectively.

Next, we show that $\mathsf{b}^{*}_2 = 1$ implies $\mathsf{b}^{**}_2 = 1$. Assume $\mathsf{b}^{**}_2 = 0$, namely, $((\mathsf{ipk}^*_i, (m'^*_{i,\ell})_\ell, \mathsf{b}^*_i)_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*) \in \mathsf{AC.PRES}$. Then $((\tilde{m}^*_{i,\ell})_\ell)_{i\in[I^*]}$ in $\tilde{\mathsf{m}}^*$ should have been computed as $(\tilde{m}^*_{i,\ell})_\ell \leftarrow \mathsf{flatten}(\varphi_i(\mathsf{canon}(G_i)))$ in line 5 of $\mathcal{O}_{\mathsf{show}}$. On the other hand, from the line 7 of G_2, we also have $(\tilde{m}^*_{i,\ell})_\ell = \mathsf{flatten}(\mathsf{reorder}(\mathsf{canon}(G'^*_i), \psi^*_i, J^*_i))$,

so that we can say $\mathsf{flatten}(\varphi_i(\mathsf{canon}(G_i))) = \mathsf{flatten}(\mathsf{reorder}(\mathsf{canon}(G'^*_i), \psi^*_i, J^*_i))$. Here, by the definition of flatten, we have $\varphi_i(\mathsf{canon}(G_i)) = \mathsf{reorder}(\mathsf{canon}(G'^*_i), \psi^*_i, J^*_i)$. Furthermore, from the definitions of φ_i, canon, and reorder, we have $\varphi_i(G_i) = G'^*_i$. Immediately after adding the tuple $((\mathsf{ipk}^*_i, (m'^*_{i,\ell})_\ell, \mathsf{b}^*_i)_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*)$ to AC.PRES on line 20 of $\mathcal{O}_{\mathsf{show}}$, PRES should get $((\mathsf{ipk}^*_i, \varphi_i(G_i), \mathsf{b}^*_i)_{i\in[I^*]}, \mathsf{m}^*)$ in line 23. Since $\varphi_i(G_i) = G'^*_i$ holds here, we can say that $\mathsf{b}^*_2 = 0$ and $\mathsf{b}^*_2 = 1 \Rightarrow \mathsf{b}^{**}_2 = 1$ as desired.

We can also show that $\mathsf{b}^*_4 = 1$ implies $\mathsf{b}^{**}_4 = 1$. Assume $\mathsf{b}^{**}_4 = 0$. Then there exist some $i^* \in [I^*]$ and some $(m_\ell)_\ell \in \mathsf{AC.ATTR}$ such that $(m'^*_{i^*,\ell})_\ell \subseteq (m_\ell)_\ell$. For each $(m'^*_{i^*,\ell})_\ell$ and $(m_\ell)_\ell$, there exist $G'^*_{i^*} = \{(\tilde{m}^*_{i^*,j}, \tilde{m}^*_{i^*,j+1}, \tilde{m}^*_{i^*,j+2})\}_j$ and $G = \{(m_j, m_{j+1}, m_{j+2})\}_j \in \mathsf{ATTR}$, respectively. We can check $G'^*_{i^*} \sqsubseteq G$, which implies $\mathsf{b}^*_4 = 0$ as desired.

Adversary $\mathcal{B}$. Figure 12 shows the description of the adversary $\mathcal{B}$ and its experiment $\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{AC}}(\lambda, L, \mathcal{B})$. We can see that the output of $\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{AC}}(\lambda, L, \mathcal{B})$ is equivalently generated to the output of G_2. Hence, we have $\Pr[\mathsf{G}_2(\lambda, L, \mathcal{A}) = 1] = \Pr\left[\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{AC}}(\lambda, L, \mathcal{B}) = 1\right]$

Putting all this together, we have $\Pr\left[\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{LDVC}_{\mathsf{AC}}}(\lambda, L, \mathcal{A}) = 1\right] \leq \Pr\left[\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{AC}}(\lambda, L, \mathcal{B}) = 1\right]$. Since the right-hand side is negligible by assumption, the left-hand side is also negligible. □

Theorem 2. $\mathsf{LDVC}_{\mathsf{AC}}$ *is weakly anonymous if* AC *is anonymous.*

Proof. Assume that we have a PPT adversary $\mathcal{A}$ against the weak anonymity of $\mathsf{LDVC}_{\mathsf{AC}}$. We will construct an adversary $\mathcal{B}$ that exploits $\mathcal{A}$ to break the anonymity of AC. Since we assume AC is anonymous, showing the following relation is sufficient to conclude this proof:

$$\Pr\left[\mathsf{Exp}^{\mathsf{wan}}_{\mathsf{LDVC}_{\mathsf{AC}}}(\lambda, L, \mathcal{A}) = 1\right] \leq \Pr[\mathsf{Exp}^{\mathsf{an}}_{\mathsf{AC}}(\lambda, L, \mathcal{B}) = 1] \tag{2}$$

To show how to construct $\mathcal{B}$ that meets Eq. (2), we define consecutive games described in Figure 13.

Game G_0. The initial game $\mathsf{G}_0(\lambda, L, \mathcal{A})$ is equivalent to $\mathsf{Exp}^{\mathsf{wan}}_{\mathsf{LDVC}_{\mathsf{AC}}}(\lambda, L, \mathcal{A})$. Therefore, we have the following.

$$\Pr\left[\mathsf{Exp}^{\mathsf{wan}}_{\mathsf{LDVC}_{\mathsf{AC}}}(\lambda, L, \mathcal{A}) = 1\right] = \Pr[\mathsf{G}_0(\lambda, L, \mathcal{A}) = 1]$$

Game G_1. We introduce a variable $\mathsf{AC.CRED}_u$ to record the credentials that the honest user obtains through $\mathcal{O}_{\mathsf{obt}}$ and $\mathcal{O}'_{\mathsf{obt}}$. Similarly to the proof of Theorem 1, we have $\Pr[\mathsf{G}_0(\lambda, L, \mathcal{A}) = 1] = \Pr[\mathsf{G}_1(\lambda, L, \mathcal{A}) = 1]$.

Game G_2. We define $\bar{b} = 1 \oplus b$, as the inverse of bit b, and additional variables $(m^*_{\bar{b},i,\ell})_\ell$, $J^*_{\bar{b},i}$, $(\tilde{m}^*_{\bar{b},i,\ell})_\ell$, $\mathcal{D}^*_{\bar{b},i}$, $\psi^*_{\bar{b},i}$, $\mathcal{E}^*_{\bar{b}}$ and $\tilde{\mathsf{m}}^*_{\bar{b}}$ in the similar fashion as the ones indexed by b. These additional variables do not affect the output of the game, so that we have $\Pr[\mathsf{G}_1(\lambda, L, \mathcal{A}) = 1] = \Pr[\mathsf{G}_2(\lambda, L, \mathcal{A}) = 1]$.

Game G_3. Here we can see that both $\psi^*_{0,i} = \psi^*_{1,i}$ and $J^*_{0,i} = J^*_{1,i}$ hold for any $i \in [I^*]$ because of the checking in line 7 of the game. Note that this guarantee holds in the *weak* anonymity game, which implies the reason why we cannot prove (full) anonymity using the same argument here. In addition, we can also check that $\mathcal{E}^*_{\bar{b}}$ and $\tilde{\mathsf{m}}^*_{\bar{b}}$ are the same as their counterparts related to $\bar{b}$. To sum up, we can omit b and $\bar{b}$ from these values without affecting the output of the game, which results in $\Pr[\mathsf{G}_2(\lambda, L, \mathcal{A}) = 1] = \Pr[\mathsf{G}_3(\lambda, L, \mathcal{A}) = 1]$.

Game G_4. With the loop starting from line 31, the tuples $(\mathsf{ipk}^*_{b',i}, (m^*_{b',i,\ell})_\ell, \sigma^*_{b',i}, \mathsf{b}^*_{b',i})$ will be overwritten with the values recorded in $\mathsf{AC.CRED}_{u^*_{b'}}[\mathsf{cid}^*_{b',i}]$ for $b' \in \{0, 1\}$. Similarly to the proof of Theorem 1, these additional operations do not affect the output of the game. Hence, $\Pr[\mathsf{G}_3(\lambda, L, \mathcal{A}) = 1] = \Pr[\mathsf{G}_4(\lambda, L, \mathcal{A}) = 1]$ holds.

Adversary $\mathcal{B}$. Figure 14 shows the description of the adversary $\mathcal{B}$ and its experiment $\mathsf{Exp}^{\mathsf{an}}_{\mathsf{AC}}(\lambda, L, \mathcal{B})$. We can see that the output of $\mathsf{Exp}^{\mathsf{an}}_{\mathsf{AC}}(\lambda, L, \mathcal{B})$ is equivalently generated to the output of G_4. Hence, we have $\Pr[\mathsf{G}_4(\lambda, L, \mathcal{A}) = 1] = \Pr[\mathsf{Exp}^{\mathsf{an}}_{\mathsf{AC}}(\lambda, L, \mathcal{B}) = 1]$

Putting all this together, we have $\Pr\left[\mathsf{Exp}^{\mathsf{wan}}_{\mathsf{LDVC}_{\mathsf{AC}}}(\lambda, L, \mathcal{A}) = 1\right] \leq \Pr[\mathsf{Exp}^{\mathsf{an}}_{\mathsf{AC}}(\lambda, L, \mathcal{B}) = 1]$. Since the right-hand side is negligible by assumption, the left-hand side is also negligible. □

4.3. Impact of Weak Anonymity in Practice

Our construction leaks the size and the canonicalised form of the original graph, due to J and ψ in the proof π, respectively. Weak anonymity guarantees that nothing further will be leaked by our construction.

The leaked size of the graph helps adversaries guess which credential is included in the presented proof. For example, assume that a user has two credentials that include G_1 and G_2, respectively, where $|G_1| \neq |G_2|$ and $G_1 \cap G_2 \neq \emptyset$. If the user shows one of these credentials that partially discloses an intersection $G_1 \cap G_2$, anyone can determine which credential is presented by the leaked size $|G_i|$, even though the disclosed attributes $G_1 \cap G_2$ themselves do not leak any information to identify the original credential. A simple workaround is to add dummy RDF triples to each graph when issuing credentials so that all graphs are the same size.

Similarly, the leaked canonicalised form of the original graph can also be exploited to identify the credentials included in the presentation. Furthermore, in some specific situations, an adversary can even guess unrevealed attribute values from revealed attributes with a leaked canonicalised form. For example, let us assume that a user has the issued credential, including an RDF graph G, which is canonicalised to get the following ordered set:

$$\begin{aligned}\mathsf{canon}(G_1) = ((&\texttt{John, children, Albert})),\\ &(\texttt{John, children, Alice}),\\ &(\texttt{John, children, Allie}))\end{aligned}$$

Even if the user selectively discloses only the first and third triples of $\mathsf{canon}(G_1)$, anyone can easily guess that the unrevealed second triple must be like

(`John`, `children`, `Al...`), because the canonicalisation algorithm used here is just a simple sorting based on lexicographic order.

We can prevent this type of attack by using random permutation to shuffle the order of a canonicalised ordered set. This random permutation is contained in the credential that is to be shared between the issuer and the user, while the user does not have to show the permutation to the verifier because it can be implicitly embedded in the map ψ. However, such a workaround cannot prevent an adversary from guessing which credential is used in the presentation. Therefore, we cannot yet have a fully anonymous LDVC scheme at the moment.

We leave it as an open problem to make a full anonymous construction of LDVC instead of the above ad hoc workarounds.

5. Performance Evaluation

We extended MATTR's open source implementation [32] of the LDP-BBS+ scheme [31] to provide a prototype[4] and a demo application[5] running in a Web browser, implemented in TypeScript, WebAssembly and Rust.

The anonymous credential scheme we used as a building block is the same as LDP-BBS +, that is, the BBS+ signature scheme [1], [9] used with the BLS 12-381 curve [6], where the size of the private key, public key, signature σ and the non-interactive zero-knowledge proof $\tilde{\pi}$ are 32 bytes, 96 bytes, 112 bytes and $368 + 32n$ bytes (where n is the number of undisclosed RDF terms), respectively. We ran the above prototype on Google Chrome with an Intel i7-10750H @ 2.60GHz (6 cores, 12 threads) and 32GB RAM and measured its performance.

Figure 9 shows the processing time of `sign` and `sigVf` for the number of RDF terms contained in the Linked Data to be signed, where "sign (AC)" and "sigVf (AC)" indicate only the time for the anonymous credential computation, that is, BBS + signature generation and verification. Similarly, the processing times of `show` and `verify` for the number of *unrevealed* RDF terms during the verifiable presentation are shown in Figure 10. Here, the number of *revealed* terms is fixed at 30 in all cases because the dominant factor is the number of unrevealed terms rather than revealed terms.

It can be seen that most of the processing time is spent on the underlying anonymous credential, that is, the BBS + signature scheme. The processing time increases in proportion to the number of RDF terms, since both our construction and the underlying BBS+ scheme do not use any accumulator in order to implement the proof of equality that is necessary to link multiple data in the zero-knowledge fashion in our proposed use case. When around 200 out of 230 RDF terms are unrevealed, the processing time required for signing, signature verification, showing, and its verification can all be performed within one second, which shows that a user using Google Chrome takes at most one second to selectively disclose 50 attributes from the Linked-Data based verifiable credentials with about 60 attributes in total.

4. https://github.com/zkp-ld/jsonld-signatures-bbs
5. https://playground.zkp-ld.org/

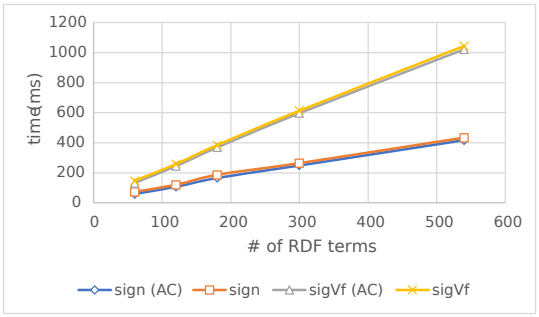

Figure 9. Processing time of sign / sigVf

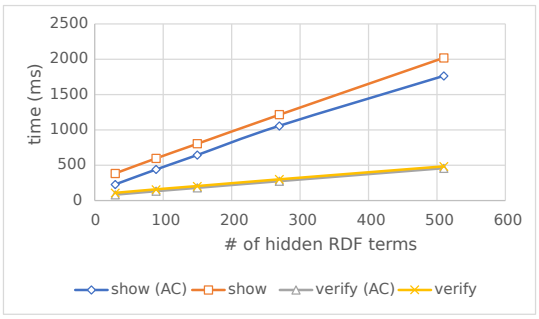

Figure 10. Processing time of show / verify

Note that the issuance of bound credentials, that is, `issue` and `obtain`, has not yet been implemented and should be treated as future work.

6. Conclusion and Future Work

In this paper, we present a construction of Linked-Data based verifiable credentials, introduce its security model handling RDF graphs, and prove the security and privacy of our construction using the model. Additionally, we implement a prototype and a demo application to evaluate the practicality of our construction. Our performance evaluation shows that a user using Google Chrome takes at most one second to selectively disclose 50 attributes from the Linked-Data based verifiable credentials with 60 attributes in total.

Our current construction satisfies a weaker notion of anonymity. Fully anonymous constructions are currently declared to be an open problem. Future work also includes adding features such as revocation, delegation, range proofs, pseudonyms, and anonymous issuer [4] to our model and configuration. As with existing graph signatures [33], [42] and redactable signatures [35], it is also desirable to achieve constant proof sizes and verification times independent of the number of attributes. Furthermore, in this paper, we have processed Linked Data in a *procedural* way using `canon`, `reorder` and other utility functions. However, it is worth considering a more *algebraic* and/or order-agnostic approach, such as the set commitment scheme in [20] or the Monipoly scheme [41], for the sake of constructing fully anonymous schemes.

References

[1] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-TAA," in *SCN 06*, ser. LNCS, R. D. Prisco and M. Yung, Eds., vol. 4116.

Springer, Heidelberg, Sep. 2006, pp. 111–125.

[2] F. Baldimtsi and A. Lysyanskaya, "Anonymous credentials light," in *ACM CCS 2013*, A.-R. Sadeghi, V. D. Gligor, and M. Yung, Eds. ACM Press, Nov. 2013, pp. 1087–1098.

[3] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data: The story so far," in *Semantic services, interoperability and web applications: emerging concepts*. IGI global, 2011, pp. 205–227.

[4] J. Bobolz, F. Eidens, S. Krenn, S. Ramacher, and K. Samelin, "Issuer-hiding attribute-based credentials," in *Cryptology and Network Security*, M. Conti, M. Stevens, and S. Krenn, Eds. Cham: Springer International Publishing, 2021, pp. 158–178.

[5] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *CRYPTO 2004*, ser. LNCS, M. Franklin, Ed., vol. 3152. Springer, Heidelberg, Aug. 2004, pp. 41–55.

[6] S. Bowe. (2017) BLS12-381: New zk-SNARK elliptic curve construction. [Online]. Available: https://electriccoin.co/blog/new-snark-curve

[7] S. Brands, *Rethinking public key infrastructures and digital certificates: building in privacy*. MIT Press, 2000.

[8] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 8259, Dec. 2017.

[9] J. Camenisch, M. Drijvers, and A. Lehmann, "Anonymous attestation using the strong diffie hellman assumption revisited," in *International Conference on Trust and Trustworthy Computing*. Springer, 2016, pp. 1–20.

[10] J. Camenisch, S. Krenn, A. Lehmann, G. L. Mikkelsen, G. Neven, and M. Ø. Pedersen, "Formal treatment of privacy-enhancing credential systems," in *SAC 2015*, ser. LNCS, O. Dunkelman and L. Keliher, Eds., vol. 9566. Springer, Heidelberg, Aug. 2016, pp. 3–24.

[11] J. Camenisch and A. Lysyanskaya, "A signature scheme with efficient protocols," in *SCN 02*, ser. LNCS, S. Cimato, C. Galdi, and G. Persiano, Eds., vol. 2576. Springer, Heidelberg, Sep. 2003, pp. 268–289.

[12] ——, "Signature schemes and anonymous credentials from bilinear maps," in *CRYPTO 2004*, ser. LNCS, M. Franklin, Ed., vol. 3152. Springer, Heidelberg, Aug. 2004, pp. 56–72.

[13] S. Celi, A. Davidson, A. Faz-Hernández, S. Valdez, and C. Wood. (2022, Apr.) Privacy pass issuance protocol (draft-ietf-privacypass-protocol-04). Work in progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-privacypass-protocol/

[14] D. Chaum, "Blind signatures for untraceable payments," in *CRYPTO'82*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Plenum Press, New York, USA, 1982, pp. 199–203.

[15] ——, "Security without identification: Transaction systems to make big brother obsolete," *Communications of the Association for Computing Machinery*, vol. 28, no. 10, pp. 1030–1044, 1985.

[16] S. Curren, T. Looker, and O. Terbu. (2022) DIDComm messaging. Work in progress. [Online]. Available: https://identity.foundation/didcomm-messaging/spec/

[17] A. Davidson, I. Goldberg, N. Sullivan, G. Tankersley, and F. Valsorda, "Privacy pass: Bypassing internet challenges anonymously," *PoPETs*, vol. 2018, no. 3, pp. 164–180, Jul. 2018.

[18] European Union. Eu digital covid certificate. [Online]. Available: https://ec.europa.eu/info/live-work-travel-eu/coronavirus-response/safe-covid-19-vaccines-europeans/eu-digital-covid-certificate_en

[19] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *CRYPTO'86*, ser. LNCS, A. M. Odlyzko, Ed., vol. 263. Springer, Heidelberg, Aug. 1987, pp. 186–194.

[20] G. Fuchsbauer, C. Hanser, and D. Slamanig, "Structure-preserving signatures on equivalence classes and constant-size anonymous credentials," *Journal of Cryptology*, vol. 32, no. 2, pp. 498–546, Apr. 2019.

[21] T. Groß, "Signatures and efficient proofs on committed graphs and NP-statements," in *FC 2015*, ser. LNCS, R. Böhme and T. Okamoto, Eds., vol. 8975. Springer, Heidelberg, Jan. 2015, pp. 293–314.

[22] A. Hogan, "Canonical forms for isomorphic and equivalent RDF graphs: algorithms for leaning and labelling blank nodes," *ACM Transactions on the Web (TWEB)*, vol. 11, no. 4, pp. 1–62, 2017.

[23] IATA(International Air Transport Association). IATA travel pass initiative. [Online]. Available: https://www.iata.org/en/programs/passenger/travel-pass/

[24] ISO Central Secretary, "Financial services — legal entity identifier (LEI) — part 1: Assignment," International Organization for Standardization, Geneva, CH, Standard ISO/IEC TR 17442-1:2020, 2020.

[25] Kaliya Young. Verifiable credentials flavors explained. [Online]. Available: https://www.lfph.io/wp-content/uploads/2021/02/Verifiable-Credentials-Flavors-Explained.pdf

[26] G. Kellogg, D. Longley, and P.-A. Champin. (2020, Jul.) JSON-LD 1.1. W3C. [Online]. Available: https://www.w3.org/TR/json-ld11/

[27] ——. (2020, Jul.) JSON-LD 1.1 processing algorithms and API. W3C. [Online]. Available: https://www.w3.org/TR/json-ld11-api/

[28] T. Lodderstedt and K. Yasuda. (2021, Dec.) OpenID Connect for verifiable credential issuance. Work in progress. [Online]. Available: https://openid.net/specs/openid-connect-4-verifiable-credential-issuance-1_0.html

[29] D. Longley. (2021) RDF dataset canonicalization. W3C Credentials Community Group. Work in progress. [Online]. Available: https://json-ld.github.io/rdf-dataset-canonicalization/spec/

[30] D. Longley and M. Sporny. (2022, Mar.) Data integrity 1.0. W3C Credentials Community Group. Work in progress. [Online]. Available: https://w3c-ccg.github.io/data-integrity-spec/

[31] T. Looker and O. Steele. (2021) BBS+ signatures 2020. W3C Credentials Community Group. [Online]. Available: https://w3c-ccg.github.io/ldp-bbs2020/

[32] MATTR Limited. jsonld-signatures-bbs. [Online]. Available: https://github.com/mattrglobal/jsonld-signatures-bbs

[33] T. Nakanishi, H. Yoshino, T. Murakami, and G.-V. Policharla, "Efficient zero-knowledge proofs of graph signature for connectivity and isolation using bilinear-map accumulator," in *Proceedings of the 7th ACM Workshop on ASIA Public-Key Cryptography*, 2020, pp. 9–18.

[34] D. Pointcheval and O. Sanders, "Reassessing security of randomizable signatures," in *CT-RSA 2018*, ser. LNCS, N. P. Smart, Ed., vol. 10808. Springer, Heidelberg, Apr. 2018, pp. 319–338.

[35] O. Sanders, "Efficient redactable signature and application to anonymous credentials," in *PKC 2020, Part II*, ser. LNCS, A. Kiayias, M. Kohlweiss, P. Wallden, and V. Zikas, Eds., vol. 12111. Springer, Heidelberg, May 2020, pp. 628–656.

[36] Schema.org. [Online]. Available: https://schema.org

[37] G. Schreiber and Y. Raimond. (2014, Jun.) RDF 1.1 primer. W3C. [Online]. Available: https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/

[38] Y. Sheffer, D. Hardt, and M. Jones, "JSON Web Token Best Current Practices," RFC 8725, Feb. 2020.

[39] M. Sporny, D. Longley, M. Sabadello, D. Reed, O. Steele, and C. Allen. (2021, Aug.) Decentralized identifiers (dids) v1.0. W3C. [Online]. Available: https://www.w3.org/TR/did-core/

[40] M. Sporny, G. Noble, D. Longley, D. Burnett, B. Zundel, and K. D. Hartog. (2022, Mar.) Verifiable credentials data model 1.1. W3C. [Online]. Available: https://www.w3.org/TR/2022/REC-vc-data-model-20220303/

[41] S.-Y. Tan and T. Groß, "MoniPoly - an expressive q-SDH-based anonymous attribute-based credential system," in *ASIACRYPT 2020, Part III*, ser. LNCS, S. Moriai and H. Wang, Eds., vol. 12493. Springer, Heidelberg, Dec. 2020, pp. 498–526.

[42] S.-Y. Tan, I. Sfyrakis, and T. Gross, "A q-SDH-based graph signature scheme on full-domain messages with efficient protocols," Cryptology ePrint Archive, Report 2020/1403, 2020, https://eprint.iacr.org/2020/1403.

[43] O. Terbu, T. Lodderstedt, K. Yasuda, A. Lemmon, and T. Looker. (2022, Jan.) OpenID Connect for verifiable presentations. Work in progress. [Online]. Available: https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0-ID1.html

[44] K. Yasuda and M. B. Jones. (2022, Jan.) Self-issued OpenID provider v2. Work in progress. [Online]. Available: https://openid.net/specs/openid-connect-self-issued-v2-1_0-ID1.html

G_0, G_1, G_2

$\mathsf{prm} \leftarrow \mathsf{AC.prmGen}(1^\lambda, 3L)$
$(\mathsf{isk}^*, \mathsf{ipk}^*) \leftarrow \mathsf{AC.ikGen}(\mathsf{prm})$
for $u \in [U]$: $\mathsf{usk}_u^* \leftarrow \mathsf{AC.uskGen}(\mathsf{prm})$
$((\mathsf{ipk}_i^*, G_i'^*, \mathsf{b}_i^*)_{i\in[I^*]}, \mathsf{m}^*, \pi^*)$
$\quad \leftarrow \mathcal{A}(\mathsf{prm}, \mathsf{ipk}^*; \mathcal{O}_{\mathsf{obtiss}}, \mathcal{O}_{\mathsf{iss}}, \mathcal{O}_{\mathsf{sign}}, \mathcal{O}_{\mathsf{show}})$
$(\tilde{\pi}^*, (\psi_i^*, J_i^*)_{i\in[I^*]}) \leftarrow \pi^*$
for $i \in [I^*]$:
  $(\tilde{m}_{i,\ell}^*)_\ell \leftarrow \mathsf{flatten}(\mathsf{reorder}(\mathsf{canon}(G_i'^*), \psi_i^*, J_i^*))$
  $(m_{i,\ell}'^*)_\ell \leftarrow (\tilde{m}_{i,\ell}^*)_\ell$
  for $\tilde{m}_{i,\ell}^* \in (\tilde{m}_{i,\ell}^*)_\ell$:
    if $\tilde{m}_{i,\ell}^* \in \mathcal{X}$:
      $\mathsf{Eq}[\tilde{m}_{i,\ell}^*] \leftarrow \mathsf{Eq}[\tilde{m}_{i,\ell}^*] \cup \{(i,\ell)\}$; $m_{i,\ell}'^* \leftarrow \bot$
$\mathcal{E}^* \leftarrow \{\mathsf{Eq}[X] : X \in \mathcal{X}\}$
$\tilde{\mathsf{m}}^* \leftarrow (((\tilde{m}_{i,\ell}^*)_\ell, \psi_i^*, J_i^*)_{i\in[I^*]}, \mathcal{E}^*, \mathsf{m}^*)$
$b_1^* \leftarrow \mathsf{AC.verify}((\mathsf{ipk}_i^*, (m_{i,\ell}'^*)_\ell, \mathsf{b}_i^*)_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*, \tilde{\pi}^*)$
$b_1^{**} \leftarrow \mathsf{AC.verify}((\mathsf{ipk}_i^*, (m_{i,\ell}'^*)_\ell, \mathsf{b}_i^*)_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*, \tilde{\pi}^*)^{(2)}$
$b_2^* \leftarrow [((\mathsf{ipk}_i^*, G_i'^*, \mathsf{b}_i^*)_{i\in[I^*]}, \mathsf{m}^*) \notin \mathsf{PRES}]$
$b_2^{**} \leftarrow [((\mathsf{ipk}_i^*, (m_{i,\ell}'^*)_\ell, \mathsf{b}_i^*)_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*) \notin \mathsf{AC.PRES}]^{(2)}$
$b_3^* \leftarrow [\mathsf{ipk}^* \in \{\mathsf{ipk}_i^*\}_{i\in[I^*]}]$; $b_4^* \leftarrow \top$
$b_3^{**} \leftarrow [\mathsf{ipk}^* \in \{\mathsf{ipk}_i^*\}_{i\in[I^*]}]$; $b_4^{**} \leftarrow \top^{(2)}$
for $i^* \in \{i \in [I^*] : \mathsf{ipk}_i^* = \mathsf{ipk}^*\}$:
  $b_4^* \leftarrow b_4^* \wedge [\forall G \in \mathsf{ATTR}.\ G_{i^*}'^* \not\sqsubseteq G]$
  $b_4^{**} \leftarrow b_4^{**} \wedge [\forall (m_\ell)_\ell \in \mathsf{AC.ATTR}.\ (m_{i^*,\ell}'^*)_\ell \not\sqsubseteq (m_\ell)_\ell]^{(2)}$
return $b_1^* \wedge b_2^* \wedge b_3^* \wedge b_4^{*(0,1)}$
return $b_1^{**} \wedge b_2^{**} \wedge b_3^{**} \wedge b_4^{**(2)}$

$\mathcal{O}_{\mathsf{show}}(u, (\mathsf{cid}_i, \varphi_i)_{i\in[I]}, \mathsf{m})$

if $(\varphi_i)_{i\in[I]}$ is incorrect : **return** $\bot$
for $i \in [I]$:
  $(\mathsf{ipk}_i, G_i, \sigma_i, \mathsf{b}_i) \leftarrow \mathsf{CRED}_u[\mathsf{cid}_i]$; $G_i' \leftarrow \varphi_i(G_i)$
  $(m_{i,\ell})_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G_i))$; $J_i \leftarrow |G_i|$
  $(\tilde{m}_{i,\ell})_\ell \leftarrow \mathsf{flatten}(\varphi_i(\mathsf{canon}(G_i)))$
  for $\tilde{m}_{i,\ell} \in (\tilde{m}_{i,\ell})_\ell$:
    if $\tilde{m}_{i,\ell} \in \mathcal{X}$: $\mathsf{Eq}[\tilde{m}_{i,\ell}] \leftarrow \mathsf{Eq}[\tilde{m}_{i,\ell}] \cup \{(i,\ell)\}$
    elseif $\tilde{m}_{i,\ell} \neq \bot$: $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{\ell\}$
  $\psi_i \leftarrow \mathsf{mapGen}(\mathsf{canon}(\varphi_i(G_i)), \varphi_i(\mathsf{canon}(G_i)))$
$\mathcal{E} \leftarrow \{\mathsf{Eq}[X] : X \in \mathcal{X}\}$
$\tilde{\mathsf{m}} \leftarrow (((\tilde{m}_{i,\ell})_\ell, \psi_i, J_i)_{i\in[I]}, \mathcal{E}, \mathsf{m})$
for $i \in [I]$:$^{(1,2)}$
  $(\mathsf{ipk}_i, (m_{i,\ell})_\ell, \sigma_i, \mathsf{b}_i) \leftarrow \mathsf{AC.CRED}_u[\mathsf{cid}_i]^{(1,2)}$
  for $m_{i,\ell} \in (m_{i,\ell})_\ell$:$^{(1,2)}$
    if $\ell \in \mathcal{D}_i$: $m'_{i,\ell} \leftarrow m_{i,\ell}$ **else** : $m'_{i,\ell} \leftarrow \bot^{(1,2)}$
if $\exists (i,\ell), (i',\ell') \in E \in \mathcal{E}.\ [m_{(i,\ell)} \neq m_{(i',\ell')}]$: **return** $\bot^{(1,2)}$
if $\exists (i,\ell) \in E \in \mathcal{E}.\ [\ell \in \mathcal{D}_i]$: **return** $\bot^{(1,2)}$
$\tilde{\pi} \leftarrow \mathsf{AC.show}(\mathsf{usk}_u^*, (\mathsf{ipk}_i, (m_{i,\ell})_\ell, \sigma_i, \mathcal{D}_i, \mathsf{b}_i)_{i\in[I]}, \mathcal{E}, \tilde{\mathsf{m}})$
if $\tilde{\pi} = \bot$: **return** $\bot$
$\mathsf{AC.PRES} \leftarrow \mathsf{AC.PRES} \cup \{((\mathsf{ipk}_i, (m'_{i,\ell})_\ell, \mathsf{b}_i)_{i\in[I]}, \mathcal{E}, \tilde{\mathsf{m}})\}^{(1,2)}$
$\pi \leftarrow (\tilde{\pi}, (\psi_i, J_i)_{i\in[I]})$
if $\pi = \bot$: **return** $\bot$
$\mathsf{PRES} \leftarrow \mathsf{PRES} \cup \{((\mathsf{ipk}_i, G_i', \mathsf{b}_i)_{i\in[I]}, \mathsf{m})\}$
return π

$\mathcal{O}_{\mathsf{obtiss}}(u, G, \mathsf{b})$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
if b : $\langle \sigma, \cdot \rangle \leftarrow \langle \mathsf{AC.obtain}(\mathsf{usk}_u^*, \mathsf{ipk}^*, (m_\ell)_\ell), \mathsf{AC.issue}(\mathsf{isk}^*, (m_\ell)_\ell) \rangle$
else : $\sigma \leftarrow \mathsf{AC.sign}(\mathsf{isk}^*, (m_\ell)_\ell)$
if $\sigma = \bot$: **return** $\bot$
$\mathsf{cid} \leftarrow\$ \{0,1\}^*$; $\mathsf{CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}^*, G, \sigma, \mathsf{b})$
$\mathsf{AC.CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}^*, (m_\ell)_\ell, \sigma, \mathsf{b})^{(1,2)}$
return cid

$\mathcal{O}_{\mathsf{iss}}(G)$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
$\langle \cdot, b \rangle \leftarrow \langle \mathcal{A}, \mathsf{AC.issue}(\mathsf{isk}^*, (m_\ell)_\ell) \rangle$
if $b \neq \bot$:
  $\mathsf{ATTR} \leftarrow \mathsf{ATTR} \cup \{G\}$
  $\mathsf{AC.ATTR} \leftarrow \mathsf{AC.ATTR} \cup \{(m_\ell)_\ell\}^{(1,2)}$

$\mathcal{O}_{\mathsf{sign}}(G)$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
$\sigma \leftarrow \mathsf{AC.sign}(\mathsf{isk}^*, (m_\ell)_\ell)$
$\mathsf{ATTR} \leftarrow \mathsf{ATTR} \cup \{G\}$
$\mathsf{AC.ATTR} \leftarrow \mathsf{AC.ATTR} \cup \{(m_\ell)_\ell\}^{(1,2)}$
return σ

Figure 11. Unforgeability games of $\mathsf{LDVC}_{\mathsf{AC}}$. Highlighted codes with superscript $i \in \{0,1,2\}$ are evaluated only in game G_i.

$\mathsf{Exp}^{\mathsf{uf}}_{\mathsf{AC}}(\lambda, 3L, \mathcal{B})$

$\mathsf{prm} \leftarrow \mathsf{AC.prmGen}(1^\lambda, 3L)$
$(\mathsf{isk}^*, \mathsf{ipk}^*) \leftarrow \mathsf{AC.ikGen}(\mathsf{prm})$
for $u \in [U]$: $\mathsf{usk}^*_u \leftarrow \mathsf{AC.uskGen}(\mathsf{prm})$
$((\mathsf{ipk}^*_i, (m'^*_{i,\ell})_\ell, \mathsf{b}^*_i)_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*, \tilde{\pi}^*)$
$\leftarrow \mathcal{B}(\mathsf{prm}, \mathsf{ipk}^*; \mathcal{O}_{\mathsf{AC.obtiss}}, \mathcal{O}_{\mathsf{AC.iss}}, \mathcal{O}_{\mathsf{AC.sign}}, \mathcal{O}_{\mathsf{AC.show}})$
  $((\mathsf{ipk}^*_i, G'^*_i, \mathsf{b}^*_i)_{i\in[I^*]}, \mathsf{m}^*, \pi^*) \leftarrow \mathcal{A}(\mathsf{prm}, \mathsf{ipk}^*; \mathcal{O}_{\mathsf{obtiss}}, \mathcal{O}_{\mathsf{iss}}, \mathcal{O}_{\mathsf{sign}}, \mathcal{O}_{\mathsf{show}})$
  $(\tilde{\pi}^*, (\psi^*_i, J^*_i)_{i\in[I^*]}) \leftarrow \pi^*$
  for $i \in [I^*]$:
    $(\tilde{m}^*_{i,\ell})_\ell \leftarrow \mathsf{flatten}(\mathsf{reorder}(\mathsf{canon}(G'^*_i), \psi^*_i, J^*_i))$
    $(m'^*_{i,\ell})_\ell \leftarrow (\tilde{m}^*_{i,\ell})_\ell$
    for $\tilde{m}^*_{i,\ell} \in (\tilde{m}^*_{i,\ell})_\ell$:
      if $\tilde{m}^*_{i,\ell} \in \mathcal{X}$:
        $\mathsf{Eq}[\tilde{m}^*_{i,\ell}] \leftarrow \mathsf{Eq}[\tilde{m}^*_{i,\ell}] \cup \{(i,\ell)\}$; $m'^*_{i,\ell} \leftarrow \perp$
  $\mathcal{E}^* \leftarrow \{\mathsf{Eq}[X] : X \in \mathcal{X}\}$
  $\tilde{\mathsf{m}}^* \leftarrow (((\tilde{m}^*_{i,\ell})_\ell, \psi^*_i, J^*_i)_{i\in[I^*]}, \mathcal{E}^*, \mathsf{m}^*)$
  return $((\mathsf{ipk}^*_i, (m'^*_{i,\ell})_\ell, \mathsf{b}^*_i)_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*, \tilde{\pi}^*)$
$b^{**}_1 \leftarrow \mathsf{AC.verify}((\mathsf{ipk}^*_i, (m'^*_{i,\ell})_\ell, \mathsf{b}^*_i)_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*, \tilde{\pi}^*)$
$b^{**}_2 \leftarrow [((\mathsf{ipk}^*_i, (m'^*_{i,\ell})_\ell, \mathsf{b}^*_i)_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*) \notin \mathsf{AC.PRES}]$
$b^{**}_3 \leftarrow [\mathsf{ipk}^* \in \{\mathsf{ipk}^*_i\}_{i\in[I^*]}]$; $b^{**}_4 \leftarrow \top$
for $i^* \in \{i \in [I^*] : \mathsf{ipk}^*_i = \mathsf{ipk}^*\}$:
  $b^{**}_4 \leftarrow b^{**}_4 \wedge [\forall (m_\ell)_\ell \in \mathsf{AC.ATTR}.\ (m'^*_{i^*,\ell})_\ell \not\sqsubseteq (m_\ell)_\ell]$
return $b^{**}_1 \wedge b^{**}_2 \wedge b^{**}_3 \wedge b^{**}_4$

$\mathcal{O}_{\mathsf{show}}(u, (\mathsf{cid}_i, \varphi_i)_{i\in[I]}, \mathsf{m})$ // given to $\mathcal{A}$ by $\mathcal{B}$

if $(\varphi_i)_{i\in[I]}$ is incorrect : **return** $\perp$
for $i \in [I]$:
  $(\mathsf{ipk}_i, G_i, \sigma_i, \mathsf{b}_i) \leftarrow \mathsf{CRED}_u[\mathsf{cid}_i]$; $G'_i \leftarrow \varphi_i(G_i)$
  $(m_{i,\ell})_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G_i))$; $J_i \leftarrow |G_i|$
  $(\tilde{m}_{i,\ell})_\ell \leftarrow \mathsf{flatten}(\varphi_i(\mathsf{canon}(G_i)))$
  for $\tilde{m}_{i,\ell} \in (\tilde{m}_{i,\ell})_\ell$:
    if $\tilde{m}_{i,\ell} \in \mathcal{X}$: $\mathsf{Eq}[\tilde{m}_{i,\ell}] \leftarrow \mathsf{Eq}[\tilde{m}_{i,\ell}] \cup \{(i,\ell)\}$
    elseif $\tilde{m}_{i,\ell} \neq \perp$: $\mathcal{D}_i \leftarrow \mathcal{D}_i \cup \{\ell\}$
  $\psi_i \leftarrow \mathsf{mapGen}(\mathsf{canon}(\varphi_i(G_i)), \varphi_i(\mathsf{canon}(G_i)))$
$\mathcal{E} \leftarrow \{\mathsf{Eq}[X] : X \in \mathcal{X}\}$
$\tilde{\mathsf{m}} \leftarrow (((\tilde{m}_{i,\ell})_\ell, \psi_i, J_i)_{i\in[I]}, \mathcal{E}, \mathsf{m})$
$\tilde{\pi} \leftarrow \mathcal{O}_{\mathsf{AC.show}}(u, (\mathsf{cid}_i, \mathcal{D}_i)_{i\in[I]}, \mathcal{E}, \mathsf{m})$ // given to $\mathcal{B}$
  for $i \in [I]$:
    $(\mathsf{ipk}_i, (m_{i,\ell})_\ell, \sigma_i, \mathsf{b}_i) \leftarrow \mathsf{AC.CRED}_u[\mathsf{cid}_i]$
    for $m_{i,\ell} \in (m_{i,\ell})_\ell$:
      if $\ell \in \mathcal{D}_i$: $m'_{i,\ell} \leftarrow m_{i,\ell}$ **else** : $m'_{i,\ell} \leftarrow \perp$
  if $\exists (i,\ell), (i',\ell') \in E \in \mathcal{E}.\ [m_{(i,\ell)} \neq m_{(i',\ell')}]$: **return** $\perp$
  if $\exists (i,\ell) \in E \in \mathcal{E}.\ [\ell \in \mathcal{D}_i]$: **return** $\perp$
  $\tilde{\pi} \leftarrow \mathsf{AC.show}(\mathsf{usk}^*_u, (\mathsf{ipk}_i, (m_{i,\ell})_\ell, \sigma_i, \mathcal{D}_i, \mathsf{b}_i)_{i\in[I]}, \mathcal{E}, \tilde{\mathsf{m}})$
  if $\tilde{\pi} = \perp$: **return** $\perp$
  $\mathsf{AC.PRES} \leftarrow \mathsf{AC.PRES} \cup \{((\mathsf{ipk}_i, (m'_{i,\ell})_\ell, \mathsf{b}_i)_{i\in[I]}, \mathcal{E}, \tilde{\mathsf{m}})\}$
  return $\tilde{\pi}$
$\pi \leftarrow (\tilde{\pi}, (\psi_i, J_i)_{i\in[I]})$
if $\pi = \perp$: **return** $\perp$
$\mathsf{PRES} \leftarrow \mathsf{PRES} \cup \{((\mathsf{ipk}_i, G'_i, \mathsf{b}_i)_{i\in[I]}, \mathsf{m})\}$
return π

$\mathcal{O}_{\mathsf{obtiss}}(u, G, \mathsf{b})$ // given to $\mathcal{A}$ by $\mathcal{B}$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
$\mathsf{cid} \leftarrow \mathcal{O}_{\mathsf{AC.obtiss}}(u, (m_\ell)_\ell, \mathsf{b})$ // given to $\mathcal{B}$
  if $\mathsf{b} : \langle \sigma, \cdot \rangle \leftarrow \langle \mathsf{AC.obtain}(\mathsf{usk}^*_u, \mathsf{ipk}^*, (m_\ell)_\ell), \mathsf{AC.issue}(\mathsf{isk}^*, (m_\ell)_\ell) \rangle$
  else : $\sigma \leftarrow \mathsf{AC.sign}(\mathsf{isk}^*, (m_\ell)_\ell)$
  if $\sigma = \perp$: **return** $\perp$
  $\mathsf{cid} \leftarrow\!\!\$\ \{0,1\}^*$
  $\mathsf{AC.CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}^*, (m_\ell)_\ell, \sigma, \mathsf{b})$
  return cid
$\mathsf{CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}^*, G, \sigma, \mathsf{b})$; **return** cid

$\mathcal{O}_{\mathsf{iss}}(G)$ // given to $\mathcal{A}$ by $\mathcal{B}$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
$\mathcal{O}_{\mathsf{AC.iss}}((m_\ell)_\ell)$ // given to $\mathcal{B}$
  $\langle \cdot, b \rangle \leftarrow \langle \mathcal{A}, \mathsf{AC.issue}(\mathsf{isk}^*, (m_\ell)_\ell) \rangle$
  if $b \neq \perp$:
    $\mathsf{AC.ATTR} \leftarrow \mathsf{AC.ATTR} \cup \{(m_\ell)_\ell\}$
if $b \neq \perp$:
  $\mathsf{ATTR} \leftarrow \mathsf{ATTR} \cup \{G\}$

$\mathcal{O}_{\mathsf{sign}}(G)$ // given to $\mathcal{A}$ by $\mathcal{B}$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
$\sigma \leftarrow \mathcal{O}_{\mathsf{AC.sign}}((m_\ell)_\ell)$ // given to $\mathcal{B}$
  $\sigma \leftarrow \mathsf{AC.sign}(\mathsf{isk}^*, (m_\ell)_\ell)$
  $\mathsf{AC.ATTR} \leftarrow \mathsf{AC.ATTR} \cup \{(m_\ell)_\ell\}$
  return σ
$\mathsf{ATTR} \leftarrow \mathsf{ATTR} \cup \{G\}$
return σ

Figure 12. Adversary $\mathcal{B}$ against the unforgeability of $\mathsf{LDVC}_{\mathsf{AC}}$.

$\mathsf{G}_0, \mathsf{G}_1, \mathsf{G}_2, \mathsf{G}_3$

$b \leftarrow\$ \{0,1\}$; $\mathsf{prm} \leftarrow \mathsf{AC.prmGen}(1^\lambda, 3L)$
for $u \in [U]$: $\mathsf{usk}^*_u \leftarrow \mathsf{AC.uskGen}(\mathsf{prm})$
$((u^*_0, \mathsf{cid}^*_{0,i}, \varphi^*_{0,i})_{i\in[I^*]}, (u^*_1, \mathsf{cid}^*_{1,i}, \varphi^*_{1,i})_{i\in[I^*]}, \mathsf{m}^*)$
$\quad \leftarrow \mathcal{A}(\mathsf{prm}; \mathcal{O}_{\mathsf{obt}}, \mathcal{O}_{\mathsf{obt}'}, \mathcal{O}_{\mathsf{show}})$
for $i \in [I^*]$:
$(\mathsf{ipk}^*_{0,i}, G^*_{0,i}, \sigma^*_{0,i}, \mathsf{b}^*_{0,i}) \leftarrow \mathsf{CRED}_{u^*_0}[\mathsf{cid}^*_{0,i}]$
$(\mathsf{ipk}^*_{1,i}, G^*_{1,i}, \sigma^*_{1,i}, \mathsf{b}^*_{1,i}) \leftarrow \mathsf{CRED}_{u^*_1}[\mathsf{cid}^*_{1,i}]$
if $(\mathsf{ipk}^*_{0,i}, \varphi^*_{0,i}(\mathsf{canon}(G^*_{0,i})), |G^*_{0,i}|, \mathsf{b}^*_{0,i})_{i\in[I^*]}$
$\quad \neq (\mathsf{ipk}^*_{1,i}, \varphi^*_{1,i}(\mathsf{canon}(G^*_{1,i})), |G^*_{1,i}|, \mathsf{b}^*_{1,i})_{i\in[I^*]}$: **return** 0
for $i \in [I^*]$:
$(m^*_{b,i,\ell})_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G^*_{b,i}))$; $J^*_{b,i} \leftarrow |G^*_{b,i}|$
$(\tilde{m}^*_{b,i,\ell})_\ell \leftarrow \mathsf{flatten}(\varphi^*_{b,i}(\mathsf{canon}(G^*_{b,i})))$
for $\tilde{m}^*_{b,i,\ell} \in (\tilde{m}^*_{b,i,\ell})_\ell$:
if $\tilde{m}^*_{b,i,\ell} \in \mathcal{X} : \mathsf{Eq}[\tilde{m}^*_{b,i,\ell}] \leftarrow \mathsf{Eq}[\tilde{m}^*_{b,i,\ell}] \cup \{(i,\ell)\}$
elseif $\tilde{m}^*_{b,i,\ell} \neq \bot : \mathcal{D}^*_{b,i} \leftarrow \mathcal{D}^*_{b,i} \cup \{\ell\}$
$\psi^*_{b,i} \leftarrow \mathsf{mapGen}(\mathsf{canon}(\varphi^*_{b,i}(G^*_{b,i})), \varphi^*_{b,i}(\mathsf{canon}(G^*_{b,i})))$
$\bar{b} \leftarrow b \oplus 1^{(2,3,4)}$
for $i \in [I^*]$:$^{(2,3,4)}$
$(m^*_{b,i,\ell})_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G^*_{b,i}))$; $J^*_{b,i} \leftarrow |G^*_{b,i}|^{(2,3,4)}$
$(\tilde{m}^*_{b,i,\ell})_\ell \leftarrow \mathsf{flatten}(\varphi^*_{b,i}(\mathsf{canon}(G^*_{b,i})))^{(2,3,4)}$
for $\tilde{m}^*_{b,i,\ell} \in (\tilde{m}^*_{b,i,\ell})_\ell$:$^{(2,3,4)}$
if $\tilde{m}^*_{b,i,\ell} \in \mathcal{X} : \mathsf{Eq}[\tilde{m}^*_{b,i,\ell}] \leftarrow \mathsf{Eq}[\tilde{m}^*_{b,i,\ell}] \cup \{(i,\ell)\}^{(2,3,4)}$
elseif $\tilde{m}^*_{b,i,\ell} \neq \bot : \mathcal{D}^*_{b,i} \leftarrow \mathcal{D}^*_{b,i} \cup \{\ell\}^{(2,3,4)}$
$\psi^*_{b,i} \leftarrow \mathsf{mapGen}(\mathsf{canon}(\varphi^*_{b,i}(G^*_{b,i})), \varphi^*_{b,i}(\mathsf{canon}(G^*_{b,i})))^{(2,3,4)}$
if $(\psi^*_{0,i}, J^*_{0,i})_i \neq (\psi^*_{1,i}, J^*_{1,i})_i$: **return** $0^{(3,4)}$
else : $(\psi^*_i, J^*_i) \leftarrow (\psi^*_{0,i}, J^*_{0,i})^{(3,4)}$
$\mathcal{E}^*_b \leftarrow \{\mathsf{Eq}[X] : X \in \mathcal{X}\}$
$\mathcal{E}^*_b \leftarrow \{\mathsf{Eq}[X] : X \in \mathcal{X}\}^{(2,3,4)}$
$\tilde{\mathsf{m}}^*_b \leftarrow (((\tilde{m}^*_{b,i,\ell})_\ell, \psi^*_{b,i}, J^*_{b,i})_{i\in[I^*]}, \mathcal{E}^*_b, \mathsf{m}^*)$
$\tilde{\mathsf{m}}^*_b \leftarrow (((\tilde{m}^*_{b,i,\ell})_\ell, \psi^*_{b,i}, J^*_{b,i})_{i\in[I^*]}, \mathcal{E}^*_b, \mathsf{m}^*)^{(2,3,4)}$
if $(\mathcal{E}^*_0, \tilde{\mathsf{m}}^*_0) \neq (\mathcal{E}^*_1, \tilde{\mathsf{m}}^*_1)$: **return** $0^{(3,4)}$
else : $(\mathcal{E}^*, \tilde{\mathsf{m}}^*) \leftarrow (\mathcal{E}^*_0, \tilde{\mathsf{m}}^*_0)^{(3,4)}$
for $i \in [I^*]$:$^{(4)}$
$(\mathsf{ipk}^*_{0,i}, (m^*_{0,i,\ell})_\ell, \sigma^*_{0,i}, \mathsf{b}^*_{0,i}) \leftarrow \mathsf{AC.CRED}_{u^*_0}[\mathsf{cid}^*_{0,i}]^{(4)}$
$(\mathsf{ipk}^*_{1,i}, (m^*_{1,i,\ell})_\ell, \sigma^*_{1,i}, \mathsf{b}^*_{1,i}) \leftarrow \mathsf{AC.CRED}_{u^*_1}[\mathsf{cid}^*_{1,i}]^{(4)}$
if $(\mathsf{ipk}^*_{0,i}, (m^*_{0,i,\ell})_{\ell\in\mathcal{D}^*_{0,i}}, \mathsf{b}^*_{0,i})_{i\in[I^*]}{}^{(4)}$
$\quad \neq (\mathsf{ipk}^*_{1,i}, (m^*_{1,i,\ell})_{\ell\in\mathcal{D}^*_{1,i}}, \mathsf{b}^*_{1,i})_{i\in[I^*]}$: **return** $0^{(4)}$
$\tilde{\pi}^* \leftarrow \mathsf{AC.show}(\mathsf{usk}^*_{u^*_b}, (\mathsf{ipk}^*_{b,i}, (m^*_{b,i,\ell})_\ell, \sigma^*_{b,i}, \mathcal{D}^*_{b,i}, \mathsf{b}^*_{b,i})_{i\in[I^*]},$
$\quad \mathcal{E}^*_b, \tilde{\mathsf{m}}^{*(0,1,2)}_b \mathcal{E}^*, \tilde{\mathsf{m}}^{*(3,4)})$
if $\tilde{\pi}^* = \bot$: **return** 0
$\pi^* \leftarrow (\tilde{\pi}^*, (\psi^*_{b,i}, J^*_{b,i})_{i\in[I^*]})^{(0,1,2)}$
$\pi^* \leftarrow (\tilde{\pi}^*, (\psi^*_i, J^*_i)_{i\in[I^*]})^{(3,4)}$
$b^* \leftarrow \mathcal{A}(\pi^*; \mathcal{O}_{\mathsf{obt}}, \mathcal{O}_{\mathsf{obt}'}, \mathcal{O}_{\mathsf{show}})$; **return** $(b^* = b)$

$\mathcal{O}_{\mathsf{obt}}(u, \mathsf{cid}, \mathsf{ipk}, G)$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
$\langle \sigma, \cdot \rangle \leftarrow \langle \mathsf{AC.obtain}(\mathsf{usk}^*_u, \mathsf{ipk}, (m_\ell)_\ell), \mathcal{A} \rangle$
if $\sigma = \bot$: **return** $\bot$
$\mathsf{CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}, G, \sigma, 1)$
$\mathsf{AC.CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}, (m_\ell)_\ell, \sigma, 1)^{(1,2,3,4)}$

$\mathcal{O}_{\mathsf{obt}'}(u, \mathsf{cid}, \mathsf{ipk}, G, \sigma)$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
if $\mathsf{AC.sigVf}(\mathsf{ipk}, (m_\ell)_\ell, \sigma) = 0$: **return** $\bot$
$\mathsf{CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}, G, \sigma, 0)$
$\mathsf{AC.CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}, (m_\ell)_\ell, \sigma, 0)^{(1,2,3,4)}$

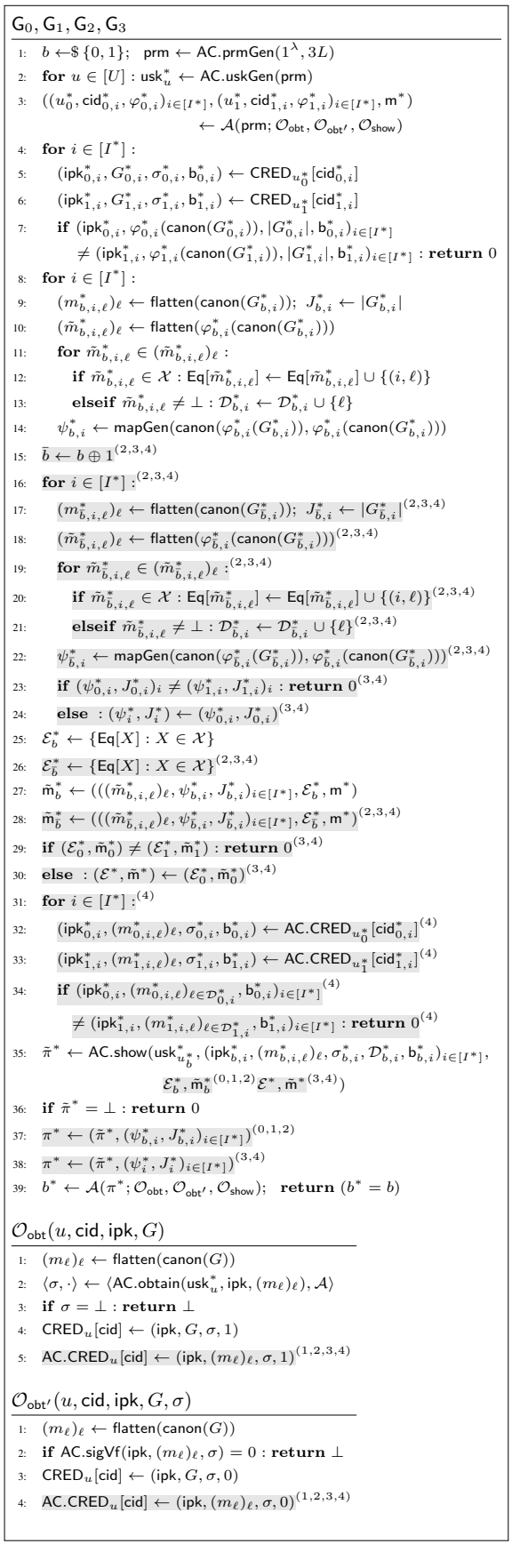

Figure 13. Weak anonymity games of LDVC. Highlighted codes with superscript i are evaluated only in game G_i. The description of $\mathcal{O}_{\mathsf{show}}$ is same as Fig. 11 and omitted here.

$\mathsf{Exp}^{\mathsf{an}}_{\mathsf{AC}}(\lambda, 3L, \mathcal{B})$

$b \leftarrow\$ \{0,1\}$; $\mathsf{prm} \leftarrow \mathsf{AC.prmGen}(1^\lambda, 3L)$
for $u \in [U]$: $\mathsf{usk}^*_u \leftarrow \mathsf{AC.uskGen}(\mathsf{prm})$
$((u^*_0, \mathsf{cid}^*_{0,i}, \mathcal{D}^*_{0,i})_{i\in[I^*]}, (u^*_1, \mathsf{cid}^*_{1,i}, \mathcal{D}^*_{1,i})_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*)$
$\leftarrow \mathcal{B}(\mathsf{prm}; \mathcal{O}_{\mathsf{AC.obt}}, \mathcal{O}_{\mathsf{AC.obt}'}, \mathcal{O}_{\mathsf{AC.show}})$
$((u^*_0, \mathsf{cid}^*_{0,i}, \varphi^*_{0,i})_{i\in[I^*]}, (u^*_1, \mathsf{cid}^*_{1,i}, \varphi^*_{1,i})_{i\in[I^*]}, \mathsf{m}^*)$
$\quad \leftarrow \mathcal{A}(\mathsf{prm}; \mathcal{O}_{\mathsf{obt}}, \mathcal{O}_{\mathsf{obt}'}, \mathcal{O}_{\mathsf{show}})$
for $i \in [I^*]$:
$(\mathsf{ipk}^*_{0,i}, G^*_{0,i}, \sigma^*_{0,i}, \mathsf{b}^*_{0,i}) \leftarrow \mathsf{CRED}_{u^*_0}[\mathsf{cid}^*_{0,i}]$
$(\mathsf{ipk}^*_{1,i}, G^*_{1,i}, \sigma^*_{1,i}, \mathsf{b}^*_{1,i}) \leftarrow \mathsf{CRED}_{u^*_1}[\mathsf{cid}^*_{1,i}]$
if $(\mathsf{ipk}^*_{0,i}, \varphi^*_{0,i}(\mathsf{canon}(G^*_{0,i})), |G^*_{0,i}|, \mathsf{b}^*_{0,i})_{i\in[I^*]}$
$\quad \neq (\mathsf{ipk}^*_{1,i}, \varphi^*_{1,i}(\mathsf{canon}(G^*_{1,i})), |G^*_{1,i}|, \mathsf{b}^*_{1,i})_{i\in[I^*]}$:
$\quad$ **return** 0
for $b' \in \{0,1\}$:
for $i \in [I^*]$:
$(m^*_{b',i,\ell})_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G^*_{b',i}))$; $J^*_{b',i} \leftarrow |G^*_{b',i}|$
$(\tilde{m}^*_{b',i,\ell})_\ell \leftarrow \mathsf{flatten}(\varphi^*_{b',i}(\mathsf{canon}(G^*_{b',i})))$
for $\tilde{m}^*_{b',i,\ell} \in (\tilde{m}^*_{b',i,\ell})_\ell$:
if $\tilde{m}^*_{b',i,\ell} \in \mathcal{X} : \mathsf{Eq}[\tilde{m}^*_{b',i,\ell}] \leftarrow \mathsf{Eq}[\tilde{m}^*_{b',i,\ell}] \cup \{(i,\ell)\}$
elseif $\tilde{m}^*_{b',i,\ell} \neq \bot : \mathcal{D}^*_{b',i} \leftarrow \mathcal{D}^*_{b',i} \cup \{\ell\}$
$\psi^*_{b',i} \leftarrow \mathsf{mapGen}(\mathsf{canon}(\varphi^*_{b',i}(G^*_{b',i})), \varphi^*_{b',i}(\mathsf{canon}(G^*_{b',i})))$
if $(\psi^*_{0,i}, J^*_{0,i})_i \neq (\psi^*_{1,i}, J^*_{1,i})_i$: **return** 0
else : $(\psi^*_i, J^*_i) \leftarrow (\psi^*_{0,i}, J^*_{0,i})$
$\mathcal{E}^*_{b'} \leftarrow \{\mathsf{Eq}[X] : X \in \mathcal{X}\}$
$\tilde{\mathsf{m}}^*_{b'} \leftarrow (((\tilde{m}^*_{b',i,\ell})_\ell, \psi^*_{b',i}, J^*_{b',i})_{i\in[I^*]}, \mathcal{E}^*_{b'}, \mathsf{m}^*)$
if $(\mathcal{E}^*_0, \tilde{\mathsf{m}}^*_0) \neq (\mathcal{E}^*_1, \tilde{\mathsf{m}}^*_1)$: **return** 0
else : $(\mathcal{E}^*, \tilde{\mathsf{m}}^*) \leftarrow (\mathcal{E}^*_0, \tilde{\mathsf{m}}^*_0)$
return $((u^*_0, \mathsf{cid}^*_{0,i}, \mathcal{D}^*_{0,i})_{i\in[I^*]},$
$\quad (u^*_1, \mathsf{cid}^*_{1,i}, \mathcal{D}^*_{1,i})_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*)$
for $i \in [I^*]$:
$(\mathsf{ipk}^*_{0,i}, (m^*_{0,i,\ell})_\ell, \sigma^*_{0,i}, \mathsf{b}^*_{0,i}) \leftarrow \mathsf{AC.CRED}_{u^*_0}[\mathsf{cid}^*_{0,i}]$
$(\mathsf{ipk}^*_{1,i}, (m^*_{1,i,\ell})_\ell, \sigma^*_{1,i}, \mathsf{b}^*_{1,i}) \leftarrow \mathsf{AC.CRED}_{u^*_1}[\mathsf{cid}^*_{1,i}]$
if $(\mathsf{ipk}^*_{0,i}, (m^*_{0,i,\ell})_{\ell\in\mathcal{D}^*_{0,i}}, \mathsf{b}^*_{0,i})_{i\in[I^*]}$
$\quad \neq (\mathsf{ipk}^*_{1,i}, (m^*_{1,i,\ell})_{\ell\in\mathcal{D}^*_{1,i}}, \mathsf{b}^*_{1,i})_{i\in[I^*]}$: **return** 0
$\tilde{\pi}^* \leftarrow \mathsf{AC.show}(\mathsf{usk}^*_{u^*_b},$
$\quad (\mathsf{ipk}^*_{b,i}, (m^*_{b,i,\ell})_\ell, \sigma^*_{b,i}, \mathcal{D}^*_{b,i}, \mathsf{b}^*_{b,i})_{i\in[I^*]}, \mathcal{E}^*, \tilde{\mathsf{m}}^*)$
$b^* \leftarrow \mathcal{B}(\tilde{\pi}^*; \mathcal{O}_{\mathsf{AC.obt}}, \mathcal{O}_{\mathsf{AC.obt}'}, \mathcal{O}_{\mathsf{AC.show}})$
$\pi^* \leftarrow (\tilde{\pi}^*, (\psi^*_i, J^*_i)_{i\in[I^*]})$
$b^* \leftarrow \mathcal{A}(\pi^*; \mathcal{O}_{\mathsf{obt}}, \mathcal{O}_{\mathsf{obt}'}, \mathcal{O}_{\mathsf{show}})$
return b^*
return $(b^* = b)$

$\mathcal{O}_{\mathsf{obt}}(u, \mathsf{cid}, \mathsf{ipk}, G)$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
$\mathcal{O}_{\mathsf{AC.obt}}(u, \mathsf{cid}, \mathsf{ipk}, (m_\ell)_\ell)$ // given to $\mathcal{B}$
$\quad \langle \sigma, \cdot \rangle \leftarrow \langle \mathsf{AC.obtain}(\mathsf{usk}^*_u, \mathsf{ipk}, (m_\ell)_\ell), \mathcal{A} \rangle$
$\quad$ **if** $\sigma = \bot$: **return** $\bot$
$\quad \mathsf{AC.CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}, (m_\ell)_\ell, \sigma, 1)$
$\mathsf{CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}, G, \sigma, 1)$

$\mathcal{O}_{\mathsf{obt}'}(u, \mathsf{cid}, \mathsf{ipk}, G, \sigma)$

$(m_\ell)_\ell \leftarrow \mathsf{flatten}(\mathsf{canon}(G))$
$\mathcal{O}_{\mathsf{AC.obt}'}(u, \mathsf{cid}, \mathsf{ipk}, (m_\ell)_\ell, \sigma)$ // given to $\mathcal{B}$
$\quad$ **if** $\mathsf{AC.sigVf}(\mathsf{ipk}, (m_\ell)_\ell, \sigma) = 0$: **return** $\bot$
$\quad \mathsf{AC.CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}, (m_\ell)_\ell, \sigma, 0)$
$\mathsf{CRED}_u[\mathsf{cid}] \leftarrow (\mathsf{ipk}, G, \sigma, 0)$

Figure 14. Adversary $\mathcal{B}$ against the anonymity of $\mathsf{LDVC}_{\mathsf{AC}}$. The description of $\mathcal{O}_{\mathsf{show}}$ is same as Fig. 12 and omitted here.