A Relational Credential System from q-SDH-based Graph Signatures

Syh-Yuan Tan Ioannis Sfyrakis Thomas Groß
School of Computing
Newcastle University, UK

***Abstract*—An attribute-based credential system enables users to prove possession of a credential and statements over certified attributes to verifiers in zero-knowledge while maintaining anonymity and unlinkability. In a relational anonymous credential system, users can further prove their relationship to other entities in their social graph, such as position in an organizational hierarchy or friends-of-friends status in an online social network graph, while protecting their own privacy and that of other users involved in the social graph. While traditional anonymous credential schemes make no provisions for privacy-preserving relationship predicates, a relational credential system is more usable, because it (i) can facilitate relationship-based access control with a wide range of predicates and (ii) offers strong privacy guarantees for relationship proofs. We propose the first relational credential scheme, based on a new q-SDH graph signature scheme and an efficient zero-knowledge proof system for graph predicates. We rigorously prove the security for the proposed scheme and provide a benchmark using Facebook social graphs.**

I. Introduction

a) Anonymous Credential Systems.: An anonymous credential system is a privacy-enhancing technology that enables users to obtain credentials signed by trusted issuers. The user is then able to prove possession of credentials, while maintaining anonymity and unlinkability. Attribute-based systems allow users to also prove in zero-knowledge statements over private attributes.

As a concept, anonymous credential systems have been proposed by Chaum [1], and since then, been refined in decades of research [2], [3], [4], [5], [6], [7], [8]. They have been extended to make them more usable and practical in a number of ways, including 1) privacy-preserving revocation [3], [9], [10], 2) efficient finite-set attribute encoding [11], [12], and 3) privacy-preserving delegation [13], [14], [15]. They have been adopted in practice, for instance, in IBM's Identity Mixer system [16] or the Trusted Computing Group's Direct Anonymous Attestation (DAA) [6]. However, existing anonymous credential systems are not well prepared to express relationships between users and entities.

b) Privacy-preserving Relational Access.: To illustrate the need for privacy-preserving relational access, we introduce an application scenario in investigative journalism in face of surveillance in the following running example:

Example 1 (Journalism Vignette). Alice is an investigative journalist at The Guardian and active on a decentralized online social network. Bob is a whistleblower, seeking to highlight ethical misconduct in his organization. He is looking for trustworthy journalists to prepare a responsible disclosure. He has hosted articles and evidence on his findings on a service provider, configured only to be visible and accessible to friends-of-a-friend in his decentralized online social network. Alice and Bob have a mutual friend Carol. She is an attorney at a civil rights organisation, the Electronic Frontier Foundation (EFF).

We visualize this scenario in Figure 2. Of course the figure only shows the tiny relevant sub-graph of the social graphs of the parties involved. Figure 1a shows the friends-relationships of the people involved and the work-relationships to their organizations. Figure 1b depicts the corresponding abstraction in a social graph, where `self` and IDs are vertex identifiers, the data enclosed with vertices and edges are labels.

What kinds of requirements might Bob place on his service provider site to restrict access to trusted individuals? In the simplest case, Bob may require that a user accessing his data is a friend-of-a-friend (FOAF). This requirement translates to a predicate on the social graph arguing about the presence of a path:

```
prove you are connected to my node (≤ two hops).
```

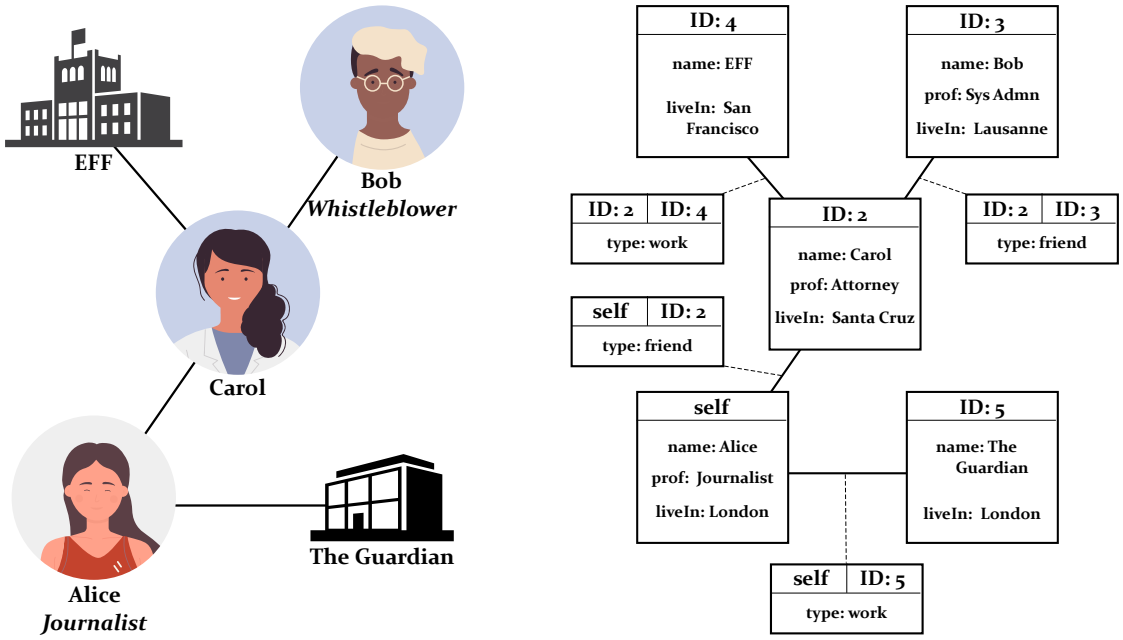
While this is a simple requirement, Bob could specify policies that are more elaborate and take structure and label distribution of sub-graphs into account.

```
prove you are connected to my node (≤ two hops) ∧
prove that you are labeled (prof: Journalist) ∧
prove that you are with The Guardian (type: work) ∧
prove that our mutual friend is with the EFF.
```

While an online social network (ONS) could facilitate such access to Bob's service provider, how can access be achieved without making the users traceable by the ONS provider? This does not only mean keeping the identities of users private, but also requires unlinkability of transactions. Of course, Bob needs to be sure that only true statements will be accepted and that the system, hence, guarantees impersonation resilience.

c) Relational Credential Systems.: We propose the new concept of a *relational credential system*. In an relational credential system (ReCS), a trusted issuer signs the user's social graph as part of the credential. The user can subsequently prove predicates over her various relationships to verifiers, while protecting anonymity. While the issuer may still maintain a comprehensive social graph to issue the credentials, the user's transactions with verifiers then become unlinkable.

Naive attempts to model similar functionality in a traditional



(a) Relationships (b) Social graph

Fig. 1: Relationships and social-graph abstraction of the running whistleblower example.

attribute-based credential (ABC) system face a combinatorial blowup of the credential size in the number of possible connection and label combinations. Complex queries that involve a node's properties essentially require computing the powerset of the attributes, as done in Okishima and Nakanishi's ABC system [17], leading to an exponential credential size.

ReCS can realize a wide range of graph predicates, including disjointness and isolation, which are particularly useful to establish separation-of-duty scenarios. In this work, we focus on connection predicates and relationship-based access control most used in the social network scenario introduced above.

A. Related Work

a) Anonymous Credential Systems.: Our proposal to create relational anonymous credentials is rooted in the Camenisch-Lysyanskaya (CL) line of anonymous credential schemes [4], [5], especially the SDH-based CL signature scheme [5], [18], [19] based on bilinear pairings. Specifically, we build our construction on the q-SDH MoniPoly attribute-based credential system proposed by Tan and Groß [8], which realizes a set commitment scheme and an attribute-based credential scheme expressed on monic polynomials. This approach is in the tradition of the work of Kate et al. [20], who offered a first conceptualization and constructions for constant-size commitments to polynomials, which was also employed in a recent attribute-based credential system [21].

b) Authenticated Data Structures.: Set commitment scheme is related to authenticated data structure [22], [23], [24] (ADS) which allows a data owner to outsource computations to a server, requiring the operations to be verifiable. Since ADS can be viewed as a database, it has been adopted to realize verifiable computation of database queries, particularly in the graph and relational databases [25], [26], [27]. While ADS also supports public verifiability, the scenario and security requirements for a graph signature scheme, however, are different: It requires information about the graph beyond the predicates proven to stay confidential. For instance, every data owner holds their own secret key for data structure authentication in an ADS scheme while it is the trusted third party who owns the secret key for graph encoding in a graph signature scheme.

c) Graph Signature Schemes.: Encoding social graphs in anonymous credentials is also related to general research on graph signatures [28], [29]. While graphs could be also be signed in transitive signatures [30], [31], these signature schemes are not well prepared for complex zero-knowledge proofs of graph predicates. Groß [32] proposed the general notion of a graph signature scheme with a corresponding proof system, which emphasised the independence of the graph encoding from the proof predicates used eventually and the expressivity in graph representations and proof systems. This construction was built on the SRSA Camenisch-Lysysanskaya signature scheme [4] and on the idea that graphs can be embedded in CL signatures using prime numbers as means of encoding. In that, this work drew inspiration from the Camenisch-Groß attribute-based credential scheme [11], [12], which offers efficient encoding of finite-set attributes based on pre-certified prime representatives.

While the SRSA graph signature scheme was originally proposed as a means to certify system topologies [33], it could also be used to certify social graphs. However, that scheme's limitation to operate over pre-certified prime representatives constitutes an obstacle to forming a relational anonymous credential system on arbitrary user identifiers in a dynamically chaning social graph. While this obstacle could be overcome in principle by hashing such identifiers to prime numbers, doing

this in zero-knowledge proofs of knowledge to safeguard privacy would entail the prohibitive costs of hash-to-prime zero-knowledge predicates [34], which would render the scheme's response time impractical for authentication and authorization scenarios.

There exist specialized realizations of graph signature schemes, such as the one by Nakanishi et al. [35], which focuses on optimized connectivity and isolation proofs by encoding strongly connected components into accumulators. The special encoding constitutes an obstacle for the proofs of general predicates over social graphs needed for the construction of relational anonymous credentials.

B. Contributions

We propose the concept and first efficient construction of a relational credential system (ReCS). It enables users not only to prove statements about private attributes, but also to prove relationship statements over their social graph while maintaining privacy. As our ReCS is constructed following the conventional commit-and-sign approach, we also propose a multi-set commitment scheme that can commit a collection of sets. The collection can be viewed as a graph if elements in the sets are interconnected. Subsequently, we devise a graph signature scheme, which can sign multi-labeled social graphs. To use the graph signature as a ReCS system, we offer an expressive and efficient graph zero-knowledge proof system, which allows users to prove predicates over their signed social graph that are not required to be anticipated at signing time.

Our new construction overcomes limitations of the prior SRSA graph signature scheme [32] as it (i) incorporate arbitrary group elements as vertex identifiers and labels out of the box, (ii) uses shorter proofs for the same range of expressive graph predicates. The first improvement is essential to enable the new relational anonymous credential system on social graphs, because it overcomes the previously mandated restriction to a pre-certified fixed graph alphabet. We prove the security of the ReCS system based on the q-SDH assumption in a tight reduction in the standard model. The computation and communication complexity of the new q-SDH graph signature scheme is consistently lower than the earlier SRSA graph signature scheme [32], unless graphs have many labels per element (> 50). Our performance evaluation on the implementation shows that it is indeed practical.

II. Relational Credential Systems

A relational credential system (ReCS) can encode entire labeled graphs $\mathcal{G}$ where the vertices and edges are bit-string attributes from an attribute space $\mathcal{M}$. To express the interface of a relational anonymous credential system, we need to define their constituent elements.

Definition 1 (Notation)**.** *We use the following symbols: 1) $\perp$: null output on error; 2) $\leftarrow$: an assignment to a variable; 3) $x \in_R S$: x is sampled uniformly at random from set S; 4) $\mathcal{P}(S)$: powerset of set S; 5) $\mathsf{A}() \leftrightarrow \mathsf{B}()$: interactive protocol between Interactive Turing Machines (ITMs) A and B; 6) $\mathcal{A}(\ldots : \mathcal{O})$: the adversary ITM $\mathcal{A}$ runs with access to oracles $\mathcal{O}$.*

Definition 2 (Graph)**.** *A* graph *over disjoint vertex and label universes, $\mathcal{V}$ and $\mathcal{L}$, is defined as $\mathcal{G} = (V, E, f_{\mathcal{V}}, f_{\mathcal{E}})$, where:*

- *V is a set of vertices from the vertex identifier universe $\mathcal{V}$,*
- *E is a set of edges drawn from $V \times V \subseteq \mathcal{E} = \mathcal{V} \times \mathcal{V}$,*
- *$f_{\mathcal{V}} : \mathcal{V} \rightarrow \mathcal{P}(\mathcal{L})$ is a relation associating labels to vertices,*
- *$f_{\mathcal{E}} : \mathcal{E} \rightarrow \mathcal{P}(\mathcal{L})$ is a relation associating labels to edges.*

We call the universe of all possible vertex and edge associations $F_{\mathcal{V}}$ and $F_{\mathcal{E}}$, respectively. For a graph $\mathcal{G}$ with vertex set V and edge set E, graph size *$|\mathcal{G}| := |V| + |E|$. We call a graph $\mathcal{G}$ a* social graph*, if $\mathcal{V} = \mathcal{V}' \cup \texttt{self}$ such that $\texttt{self} \notin \mathcal{V}'$ is a special vertex identifier designating the owner of the graph.*

Remark 1 (Self). We include a special vertex identifier `self` to introduce the concept of ownership. The identifier `self` enables owners of a relational anonymous credential to refer to their own place in their social graph.

Definition 3 (Graph in Set Representation)**.** *When a graph $\mathcal{G} = (V, E, f_{\mathcal{V}}, f_{\mathcal{E}})$ of size $|\mathcal{G}| = L$ is encoded into a multi-set $A = \{A_1, \ldots, A_L\} \in \mathfrak{M}^{L \times n}$ where $\mathfrak{M}$ is the set element space, we use the following notation:*

- *$V_i = \{i, f_{\mathcal{V}}(i)\}$ is the set of vertex with identifier i and associated labels $f_{\mathcal{V}}(i) = (m_1, \ldots, m_{n-1})$.*
- *$E_{(i,j)} = \{i, j, f_{\mathcal{E}}(i, j)\}$ is the set of undirected edge (i, j) and associated labels $f_{\mathcal{E}}(i, j) = (m_1, \ldots, m_{n-2})$.*
- *$A = \{\{V_i\}_{i \in V}, \{E_{(i,j)}\}_{(i,j) \in E}\}$*

Definition 4 (Predicates)**.** *We call a relation $\phi \subseteq (\mathcal{V}, \mathcal{E}, \mathcal{F}_{\mathcal{V}}, \mathcal{F}_{\mathcal{E}})$ a* graph predicate*. A graph $\mathcal{G} = (V, E, f_{\mathcal{V}}, f_{\mathcal{E}})$ satisfies a predicate ϕ if and only if $\phi(\mathcal{G}) = 1$. We call the set of predicates Φ.*

A. Interface

We define a relational credential system as a tuple of five (interactive) algorithms $\mathsf{ReCS} := \{\mathsf{Setup}, \mathsf{Obtain}, \mathsf{Sign}, \mathsf{Prove}, \mathsf{Verify}\}$:

1) $\texttt{Setup}(1^\kappa, L, n) \rightarrow (mpk, msk)$: On the input of the security parameter κ, maximum graph size L and maximum node size n, a signer ($\mathtt{S}$) generates the master key pair (mpk, msk).
2) $(\texttt{Obtain}(mpk, \mathcal{G}^{\mathtt{U}}) \leftrightarrow \texttt{Issue}(mpk, msk, \mathcal{G}^{\mathtt{I}})) \rightarrow (cred \text{ or } \perp)$: The `Obtain` algorithm is executed by a user $\mathtt{U}$ with a hidden graph $\mathcal{G}^{\mathtt{U}}$ while the `Issue` algorithm is executed by an issuer $\mathtt{I}$ who assigns a graph $\mathcal{G}^{\mathtt{I}}$ to the user. The issuing protocol outputs credential as $cred = (\sigma, \mathcal{G})$ if it completes successfully where σ is a valid graph signature for $\mathcal{G} = \mathcal{G}^{\mathtt{U}} \cup \mathcal{G}^{\mathtt{I}}$. Otherwise, it outputs an error $\perp$.
3) $(\texttt{Prove}(mpk, cred, \phi) \leftrightarrow \texttt{Verify}(mpk, \phi) \rightarrow (b)$: Prover's private input is a credential σ which consist of a graph signature and the corresponding graph. This interactive showing protocol establishes a show proof on $cred$ for a predicate $\phi \in \Phi$ queried by the verifier such that $\phi(\mathcal{G}) = 1$. At the end of the protocol, if $\phi(\mathcal{G}) = 1$ and prover's proof is valid, `Verify` outputs $b = 1$ and outputs $b = 0$ otherwise.

B. Security Requirements

The ReCS system is governed by the requirements *unlinkability* and *impersonation resilience* against active and concurrent attacks. Unlinkability guarantees that an adversary cannot learn anything from a user's transactions beyond the size of the graph. Impersonation resilience means that an adversary cannot make false statements or impersonate honest users. The requirements are defined in games against a probabilistic and polynomially-time Interactive Turing Machine (ITM), adversary $\mathcal{A}$.

1) Oracles: We first give an overview on the oracles required to define the security games. The details for these oracles are in Appendix A.

1) $\mathcal{O}_{\texttt{Issue}}(mpk, msk, \mathcal{G}^{\texttt{U}}, \mathcal{G}^{\texttt{I}}, sid)$: It runs an issuing protocol as an issuer $\texttt{I}$ for a session identified by sid on the graph $\mathcal{G} = (\mathcal{G}^{\texttt{U}} \cup \mathcal{G}^{\texttt{I}})$ and outputs the corresponding credential $cred$.
2) $\mathcal{O}_{\texttt{Issuing}}(mpk, \mathcal{G}^{\texttt{U}}, \mathcal{G}^{\texttt{I}}, sid)$: It returns an issuing transcript π corresponding to the graphs and session identity specified in the query.
3) $\mathcal{O}_{\texttt{Verify}}(mpk, \mathcal{G}^{\texttt{U}}, \mathcal{G}^{\texttt{I}}, \phi, sid)$: It runs a showing protocol as a verifier $\texttt{V}$ for a session identified by sid and outputs a showing transcript $\tilde{\pi}$ specified by the issuer public key mpk, graphs $(\mathcal{G}^{\texttt{U}}, \mathcal{G}^{\texttt{I}})$ and predicate ϕ in the query.
4) $\mathcal{O}_{\texttt{Showing}}(mpk, \mathcal{G}^{\texttt{U}}, \mathcal{G}^{\texttt{I}}, \phi, sid)$: It returns a showing transcript $\tilde{\pi}$ corresponds to the issuer public key mpk, graphs $\mathcal{G}^{\texttt{U}}, \mathcal{G}^{\texttt{I}}$, predicate ϕ and session identity sid specified in the query.
5) $\mathcal{O}_{\texttt{CorrU}}(\pi)$: It returns the credential $cred$ corresponds to the queried issuing transcript π.
6) $\mathcal{O}_{\texttt{CorrP}}(\tilde{\pi})$: It returns the credential $cred$ corresponds to the queried showing transcript $\tilde{\pi}$.

2) Unlinkability: We consider unlinkability as the privacy goal for ReCS, similar to that in some attribute-based credential systems [8], [21]. Unlinkability guarantees that an adversary cannot learn anything from a user beyond the size of the signed graph. In an ReCS system, we assume the issuer is honest but curious, and we concern about four types of unlinkabilities under active and concurrent attacks (ACA):

1) Issuing unlinkability (IUNL) means that an adversary should not be able to learn anything about user's hidden graph $\mathcal{G}^{\texttt{U}}$ from an issuing protocol.
2) Showing unlinkability (SUNL) means that an adversary should not be able to learn anything about user's credential $cred$ from a showing protocol.
3) Graph unlinkability (GUNL) means that an adversary should not be able to learn anything about user's signed graph $\mathcal{G} = \mathcal{G}^{\texttt{U}} \cup \mathcal{G}^{\texttt{I}}$ from a showing protocol.
4) Protocol unlinkability (PUNL) means that an adversary should not be able to learn anything about a user's hidden graph $\mathcal{G}^{\texttt{U}}$ from both issuing and showing protocols.

In the Appendix A, we show that while SUNL−ACA and IUNL−ACA are independent from each other, SUNL−ACA and GUNL−ACA are equivalent. Moreover, we prove that PUNL−ACA implies both IUNL−ACA and GUNL−ACA. Therefore, for our proposed ReCS, we prove only the security of protocol unlinkability under active and concurrent attack (PUNL − ACA) which is defined as a security game between an adaptive adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ in Table I.

Essentially, the protocol unlinkability game is a combination of the issuing unlinkability game (cf. §A) and showing unlinkability game (cf. §B). In an issuing unlinkability game, $\mathcal{A}$ declares two hidden graphs $(\mathcal{G}_0^{\texttt{U}}, \mathcal{G}_1^{\texttt{U}})$ as challenge and have to decide the sequence they are used during the issuing protocol. In a showing unlinkability game, $\mathcal{A}$ declares two credentials $(cred_0, cred_1)$ and a predicate ϕ^* as challenge and have to decide the sequence they are used during the showing protocol.

Here, in the protocol unlinkability game, $\mathcal{A}$ declares two hidden graphs $(\mathcal{G}_0^{\texttt{U}}, \mathcal{G}_1^{\texttt{U}})$ and a predicate ϕ^* as challenge and have to decide the sequences they are used during both the issuing and showing protocols. The random bit b_1 is to capture the issuing unlinkability while the random bit b_2 is to capture the showing unlinkability. If a protocol unlinkability adversary $\mathcal{A}$ can at least classify the issuing and showing protocols based on their hidden graphs, respectively, then it wins the game by outputting $b' = b_1 \oplus b_2$.

Definition 5. *A probabilistic and polynomial-time adversary $\mathcal{A}$ is said to $(t_{\mathsf{punl}}, \varepsilon_{\mathsf{punl}})$-break the security of the protocol unlinkability under active and concurrent attacks (PUNL−ACA) of a ReCS system if $\mathcal{A}$ runs in time at most t_{punl} and wins in $\mathbf{Game}^{\mathsf{punl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}}$ such that:*

$$\Pr\left[\mathbf{Game}^{\mathsf{punl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}\right] - \frac{1}{2} \geq \varepsilon_{\mathsf{punl}}.$$

We say that a ReCS is PUNL−ACA-secure, if no adversary $(t_{\mathsf{punl}}, \varepsilon_{\mathsf{punl}})$-wins $\mathbf{Game}^{\mathsf{punl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}}$.

3) Impersonation Resilience: This requirement ensures a malicious prover cannot convince a verifier that he is the owner of the credential or convince a verifier of statements that are false. We define our security model as the security against impersonation under active and concurrent attacks (IMP-ACA) [8] in the game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as in Table II.

Definition 6. *A probabilistic and polynomial-time adversary $\mathcal{A}$ is said to $(t_{\mathsf{imp}}, \varepsilon_{\mathsf{imp}})$-break the security against impersonation under active and concurrent attacks (IMP−ACA) of a ReCS system if $\mathcal{A}$ runs in time at most t_{imp} and wins in $\mathbf{Game}^{\mathsf{imp\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}}$ such that:*

$$\Pr[\mathbf{Game}^{\mathsf{imp\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}] \geq \varepsilon_{\mathsf{imp}}.$$

We say that a ReCS is imp-aca-secure if no adversary $(t_{\mathsf{imp}}, \varepsilon_{\mathsf{imp}})$-wins $\mathbf{Game}^{\mathsf{imp\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}}$.

III. Cryptographic Preliminaries

A. Zero-Knowledge Notation

We use the Camenisch-Stadler [36] notation for zero-knowledge proofs of knowledge of a statement validity in addition to the knowledge of discrete logarithms. As an example, let σ be a signature on the value C:

$$PK\{(\alpha, \beta, \sigma) : C = g^{\alpha} h^{\beta} \ \wedge \ \mathsf{Verify}(pk, \sigma, C) = 1\}$$

denotes a zero-knowledge proof of knowledge of discrete logarithms α and β such that $C = g^{\alpha} h^{\beta}$ holds and σ is a

TABLE I: The $\texttt{PUNL}-\texttt{ACA}$ security game.

$\mathbf{Game}^{\mathsf{punl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}}$

$IS \leftarrow \emptyset; SS \leftarrow \emptyset; CU \leftarrow \emptyset;\ CPV \leftarrow \emptyset;\ HU \leftarrow \emptyset;\ HPV \leftarrow \emptyset$
$\mathcal{O}(\cdot) \leftarrow \{\mathcal{O}_{\texttt{Issue}}, \mathcal{O}_{\texttt{Verify}}, \mathcal{O}_{\texttt{CorrU}}, \mathcal{O}_{\texttt{CorrP}}\}$
$b_1 \leftarrow \{0,1\};\ b_2 \leftarrow \{0,1\};\ (mpk, msk) \leftarrow \texttt{Setup}(1^\kappa, L, n)$
$(\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}, \phi^*) \leftarrow \mathcal{A}^{\mathcal{O}}(mpk, msk)$
If $|\mathcal{G}_0^{\mathtt{U}}| \neq |\mathcal{G}_1^{\mathtt{U}}|$, then return Lose
$(\pi_{b_1}, cred_{b_1}) \leftarrow (\texttt{Obtain}(mpk, \mathcal{G}_{b_1}^{\mathtt{U}}) \leftrightarrow \mathcal{A}^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\mathcal{G}_{b_1}^{\mathtt{I}}, \mathcal{G}_{1-b_1}^{\mathtt{I}}), \phi^*)$
$(\pi_{1-b_1}, cred_{1-b_1}) \leftarrow (\texttt{Obtain}(mpk, \mathcal{G}_{1-b_1}^{\mathtt{U}}) \leftrightarrow \mathcal{A}^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\mathcal{G}_{b_1}^{\mathtt{I}}, \mathcal{G}_{1-b_1}^{\mathtt{I}}), \phi^*)$
If $|\mathcal{G}_0^{\mathtt{I}}| \neq |\mathcal{G}_1^{\mathtt{I}}|$, then return Lose
If $\phi(\mathcal{G}_0^{\mathtt{U}} \cup \mathcal{G}_0^{\mathtt{I}}) = 0 \vee \phi(\mathcal{G}_1^{\mathtt{U}} \cup \mathcal{G}_1^{\mathtt{I}}) = 0$, then return Lose
$(\tilde{\pi}_{b_2}, 1) \leftarrow (\texttt{Prove}(mpk, cred_{b_2}, \phi^*) \leftrightarrow \mathcal{A}^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\mathcal{G}_{b_1}^{\mathtt{I}}, \mathcal{G}_{1-b_1}^{\mathtt{I}}), \phi^*)$
$(\tilde{\pi}_{1-b_2}, 1) \leftarrow (\texttt{Prove}(mpk, cred_{1-b_2}, \phi^*) \leftrightarrow \mathcal{A}^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\mathcal{G}_{b_1}^{\mathtt{I}}, \mathcal{G}_{1-b_1}^{\mathtt{I}}), \phi^*)$
$b' \leftarrow \mathcal{A}_2^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\mathcal{G}_{b_1}^{\mathtt{I}}, \mathcal{G}_{1-b_1}^{\mathtt{I}}), (\pi_{b_1}, \pi_{1-b_1}), \phi^*, (\tilde{\pi}_{b_2}, \tilde{\pi}_{1-b_2}))$
If $(\cdot,\cdot,\cdot,\pi_{b_1}) \in CU \vee (\cdot,\cdot,\cdot,\pi_{1-b_1}) \in CU$, then return Lose
If $(\cdot,\cdot,\cdot,\cdot,\tilde{\pi}_{b_2}) \in CPV \vee (\cdot,\cdot,\cdot,\cdot,\tilde{\pi}_{1-b_2}) \in CPV$, then return Lose
If $b_1 \oplus b_2 = b'$, then return Win, else return Lose

Note: $(cred_0, cred_1)$: credentials, ϕ^* : challenge predicate,
$(\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}})$: user hidden graphs, $(\mathcal{G}_0^{\mathtt{I}}, \mathcal{G}_1^{\mathtt{I}})$: issuer-assigned graphs,
(π_0, π_1) : issuing transcript, $(\tilde{\pi}_0, \tilde{\pi}_1)$: showing transcript,
IS: active issuing session, SS: active showing session
(CU, CPV): corrupted user list, (HU, HPV): honest user list

TABLE II: The imp-aca security game.

$\mathbf{Game}^{\mathsf{imp\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}}$

$IS \leftarrow \emptyset; SS \leftarrow \emptyset; CU \leftarrow \emptyset; CPV \leftarrow \emptyset; HU \leftarrow \emptyset; HPV \leftarrow \emptyset$
$(mpk, msk) \leftarrow \texttt{Setup}(1^\kappa, L, n)$
$\mathcal{O} \leftarrow \{\mathcal{O}_{\texttt{Issuing}}, \mathcal{O}_{\texttt{Verify}}, \mathcal{O}_{\texttt{CorrU}}, \mathcal{O}_{\texttt{CorrP}}\}$
$(\mathcal{G}^*, \phi^*) \leftarrow \mathcal{A}^{\mathcal{O}}(mpk)$
$(\tilde{\pi}, b) \leftarrow (\mathcal{A}^{\mathcal{O}}(mpk, \mathcal{G}^*, \phi^*) \leftrightarrow \mathsf{Verify}(mpk, \phi^*))$
If $\phi^*(\mathcal{G}^{\mathtt{U}} \cup \mathcal{G}^{\mathtt{I}}) = 1$ for $\exists(\cdot, \mathcal{G}^{\mathtt{U}}, \mathcal{G}^{\mathtt{I}}, \cdot, \cdot) \in CU$, return Lose
If $\phi^*(\mathcal{G}^{\mathtt{U}} \cup \mathcal{G}^{\mathtt{I}}) = 1$ for $\exists(\cdot, \mathcal{G}^{\mathtt{U}}, \mathcal{G}^{\mathtt{I}}, \cdot, \cdot, \cdot) \in CPV$, return Lose
If $b = 1$, return Win, else return Lose

Note: $\mathcal{G}^*$: *challenge graph,* ϕ^* : *challenge predicate,*
IS : *active issuing session,* SS : *active showing session,*
(CU, CPV) : *corrupted user list,* (HU, HPV) : *honest user list*

valid signature for C. Essentially, the secrets to be proven are listed in the bracket followed by the corresponding statements.

B. Mathematical Assumptions

Definition 7. *co-Discrete Logarithm Assumption (co-DLOG) [37]. An algorithm $\mathcal{C}$ is said to $(t_{\mathsf{codlog}}, \varepsilon_{\mathsf{codlog}})$-break the co-DLOG assumption if $\mathcal{C}$ runs in time at most t_{codlog} and furthermore:*

$$\Pr[x \in \mathbb{Z}_p : \mathcal{C}(g_1, g_1^x \in \mathbb{G}_1, g_2, g_2^x \in \mathbb{G}_2) = x] \geq \varepsilon_{\mathsf{dlog}}.$$

We say that the co-DLOG assumption is $(t_{\mathsf{codlog}}, \varepsilon_{\mathsf{codlog}})$-secure if no algorithm $(t_{\mathsf{codlog}}, \varepsilon_{\mathsf{codlog}})$-solves the co-DLOG problem.

Definition 8. *q-Strong Diffie-Hellman Assumption (SDH) [19]. An algorithm $\mathcal{C}$ is said to $(t_{\mathsf{sdh}}, \varepsilon_{\mathsf{sdh}})$-break the SDH assumption if $\mathcal{C}$ runs in time at most t_{sdh} and furthermore:*

$$\Pr[x \in \mathbb{Z}_p, c \in \mathbb{Z}_p \setminus \{-x\}] : \mathcal{C}(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x) = (g_1^{\frac{1}{x+c}}, c)] \geq \varepsilon_{\mathsf{sdh}}.$$

We say that the SDH assumption is $(t_{\mathsf{sdh}}, \varepsilon_{\mathsf{sdh}})$-secure if no algorithm $(t_{\mathsf{sdh}}, \varepsilon_{\mathsf{sdh}})$-solves the SDH problem.

Definition 9. *$q-$co-Strong Diffie-Hellman Assumption (co-SDH) [37]. An algorithm $\mathcal{C}$ is said to $(t_{\mathsf{cosdh}}, \varepsilon_{\mathsf{cosdh}})$-break the co-SDH assumption if $\mathcal{C}$ runs in time at most t_{cosdh} and furthermore:*

$$\Pr[x \in \mathbb{Z}_p, c \in \mathbb{Z}_p \setminus \{-x\}] : \mathcal{C}(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x, \ldots, g_2^{x^q}) = (g_1^{\frac{1}{x+c}}, c)] \geq \varepsilon_{\mathsf{cosdh}}.$$

We say that the co-SDH assumption is $(t_{\mathsf{cosdh}}, \varepsilon_{\mathsf{cosdh}})$-secure if no algorithm $(t_{\mathsf{cosdh}}, \varepsilon_{\mathsf{cosdh}})$-solves the co-SDH problem.

C. Multi-Set Commitment Schemes

Multi-set commitment schemes are designed to commit to L multiple sets of messages $\{A_1, \ldots, A_L\}$, where each constituent message set A_i is bounded by size n. We define a multi-set commitment scheme MSC as a tuple of seven algorithms:

$\texttt{MSC} := (\texttt{Setup}, \texttt{Commit}, \texttt{Open}, \texttt{OpenIntersection}, \texttt{VerifyIntersection}, \texttt{OpenDifference}, \texttt{VerifyDifference})$

1) $\texttt{Setup}\ (1^\kappa, L, n) \rightarrow (pk, sk)$. Generate a pair of public and secret keys (pk, sk) based on the security parameter 1^κ. The public key pk defines the message space $\mathfrak{M}$, the maximum number of sets in the multi-set based on L, and the maximal size of each constituent set based on n.
2) $\texttt{Commit}\ (pk, A) \rightarrow (C)$. On the input of pk, a message multi-set $A = \{A_1, \ldots, A_L\} \in \mathfrak{M}^{L \times n}$, select a set of random opening values $O = \{o_1, \ldots, o_L\}$ and output the commitment as C.
3) $\texttt{Open}\ (pk, C, A, O) \rightarrow b$. Return $b = 1$ if C is a valid commitment to multi-set A with the corresponding opening values O under pk. Return $b = 0$ otherwise.
4) $\texttt{OpenIntersection}\ (pk, C, A, O, (A', (\ell, \kappa))) \rightarrow (I, W)$ or $\perp$. Let $A' = \{A'_1, \ldots, A'_L\}, 1 \leq \ell \leq L, \kappa = \{k_1, \ldots, k_L\}$ and $I_{i,j} = A'_j \cap A_i$. If there are ℓ-many pair of indexes (i, j) such that $|I_{i,j}| = k_j \leq |A'_j|$ holds

where indexes i and j are unique, respectively, return these intersecting sets $I_{i,j}$ as the multi-set I and the corresponding witnesses W, and return an error $\perp$ otherwise.

5) $\texttt{VerifyIntersection}\ (pk, C, (I, W), (A', (\ell, \kappa))) \rightarrow b$. Return $b = 1$ if W is a witness for I being the intersecting multi-set of size ℓ for A' and the multi-set committed to in C. Return $b = 0$ otherwise.
6) $\texttt{OpenDifference}\ (pk, C, A, O, (A', (\bar{\ell}, \bar{\kappa}))) \rightarrow (D, W)$. Let $A' = \{A'_1, \ldots, A'_L\}, \bar{\kappa} = \{\bar{k}_1, \ldots, \bar{k}_L\}$, $1 \leq \bar{\ell} \leq L$ and $D_{i,j} = A'_j \setminus A_i$. If there are $\bar{\ell}$-many A'_j such that $|D_{i,j}| = \bar{k}_i \leq |A'_j|$ holds for all $A_j \in A$, return these differing sets $D_{i,j}$ as the multi-set D and the corresponding witness W, and return an error $\perp$ otherwise.
7) $\texttt{VerifyDifference}\ (pk, C, (D, W), (A', (\bar{\ell}, \bar{\kappa}))) \rightarrow b$. Return $b = 1$ if W is the witness for D being the differing multi-set of size $\bar{\ell}$ for A' and the multi-set committed to in C, and return $b = 0$ otherwise.

Based on $\texttt{MSC}$, we can define a graph commitment $\texttt{GC}$ as follows:

$$\texttt{GC} := (\texttt{Setup}, \texttt{Commit}, \texttt{Open}, \texttt{OpenIntersection}, \texttt{VerifyIntersection}, \texttt{OpenDifference}, \texttt{VerifyDifference})$$

where the multi-set $A = \texttt{G2MS}(\mathcal{G})$ is now an encoded graph which is an output of a graph encoding $\texttt{G2MS}$ (cf. §V). Therefore, whenever we invoke $\texttt{MSC}$ with a graph $\mathcal{G}$, it means the algorithms in $\texttt{MSC}$ takes in $\texttt{G2MS}(\mathcal{G})$.

Definition 10. *A multi-set commitment scheme is perfectly hiding, if every commitment $C = \mathsf{Commit}(pk, A)$ is uniformly distributed such that there exists a set of opening values $O' \neq O$ for all multi-sets $A' \neq A$ where $\mathsf{Open}(pk, C, A', O') = 1$.*

Definition 11. *An adversary $\mathcal{A}$ is said to $(t_{\mathsf{bind}}, \varepsilon_{\mathsf{bind}})$-break the binding security of a multi-set commitment scheme, if $\mathcal{A}$ runs in time at most t_{bind} and furthermore:*

$$\Pr[\mathsf{Open}(pk, C, A_1, O_1) = \mathsf{Open}(pk, C, A_2, O_2) = 1] \geq \varepsilon_{\mathsf{bind}}$$

for any two pairs $(A_1, O_1), (A_2, O_2)$ output by $\mathcal{A}$. We say that a multi-set commitment scheme is $(t_{\mathsf{bind}}, \varepsilon_{\mathsf{bind}})$-secure wrt. binding if no adversary $(t_{\mathsf{bind}}, \varepsilon_{\mathsf{bind}})$-breaks the binding security of multi-the set commitment scheme.

Definition 12. *An adversary $\mathcal{A}$ is said to $(t_{\mathsf{inbind}}, \varepsilon_{\mathsf{inbind}})$-break the intersecting binding security of a multi-set commitment scheme, if $\mathcal{A}$ runs in time at most t_{inbind} and furthermore:*

$$\Pr[\mathsf{VerifyIntersection}(pk, C, (I_1, W_1), (A', (\ell, \kappa))) = \mathsf{VerifyIntersection}(pk, C, (I_2, W_2), (A', (\ell, \kappa))) = 1] \geq \varepsilon_{\mathsf{inbind}}$$

for any two pairs $(I_1, W_1), (I_2, W_2)$ output by $\mathcal{A}$. We say that a multi-set commitment scheme is $(t_{\mathsf{inbind}}, \varepsilon_{\mathsf{inbind}})$-secure wrt. intersecting binding if no adversary $(t_{\mathsf{inbind}}, \varepsilon_{\mathsf{inbind}})$-breaks the intersecting binding security of multi-the set commitment scheme.

Definition 13. *An adversary $\mathcal{A}$ is said to $(t_{\mathsf{dibind}}, \varepsilon_{\mathsf{dibind}})$-break the differing binding security of a multi-set commitment scheme, if $\mathcal{A}$ runs in time at most t_{dibind} and furthermore:*

$$\Pr[\mathsf{VerifyDifference}(pk, C, (D_1, W_1), (A', (\bar{\ell}, \bar{\kappa}))) = \mathsf{VerifyDifference}(pk, C, (D_2, W_2), (A', (\bar{\ell}, \bar{\kappa}))) = 1] \geq \varepsilon_{\mathsf{dibind}}$$

for any two pairs $(D_1, W_1), (D_2, W_2)$ output by $\mathcal{A}$. We say that a multi-set commitment scheme is $(t_{\mathsf{dibind}}, \varepsilon_{\mathsf{dibind}})$-secure wrt. differing binding if no adversary $(t_{\mathsf{dibind}}, \varepsilon_{\mathsf{dibind}})$-breaks the differing binding security of multi-the set commitment scheme.

D. SDH Camenisch-Lysyanskaya Signatures

The SDH-based CL signature scheme [5], [18], [19] is closely related to the BBS signature scheme [38] and the foundation of the MoniPoly Attribute-based Credential system [8] we use in our construction. We define the signature scheme on messages m from a message space $\mathfrak{M} = \mathbb{Z}_p^*$. The SDH CL-signature scheme is a tuple of five algorithms $\texttt{CL} := \{\texttt{KeyGen}, \texttt{Sign}, \texttt{Verify}, \texttt{Randomize}, \texttt{VerifyRand}\}$:

$\texttt{KeyGen}\ (1^\kappa)$. Select random generators $a, b, c \in_R \mathbb{G}_1$, $g_2 \in_R \mathbb{G}_2$ and a secret value $x \in_R \mathbb{Z}_p^*$. Output the public key $pk = ((\mathsf{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p),\ a, b, c, g_2, X = g_2^x)$ and the secret key $sk = (x)$.

$\texttt{Sign}\ (pk, sk, m)$. Choose $s, t \in_R \mathbb{Z}_p^*$. For $m \in \mathfrak{M}$, compute $v = (a_0^m b^s c)^{\frac{1}{x+t}}$ where the signature is $\sigma = (t, s, v)$.

$\texttt{Verify}\ (pk, \sigma, m)$. Output 1 if $\mathsf{e}(v, X) = \mathsf{e}\,(a^m b^s c v^{-t}, g_2)$ holds and output 0 otherwise.

Theorem 1 ([19])**.** *The SDH-CL signature is strongly existential unforgeable against chosen message attack in the standard model if the SDH problem is intractable.*

There are two derivative algorithms from the MoniPoly credential randomization [8], [39]:

$\texttt{Randomize}\ (pk, \sigma)$. Choose $r, y \in_R \mathbb{Z}_p^*$ and output the randomized signature $sig' = (r, y, t' = ty, s' = sr, v' = v^{ry^{-1}})$.

$\texttt{VerifyRand}\ (pk, \sigma', m)$. Output 1 if $\mathsf{e}(v'^y, X) = \mathsf{e}\left(a^{mr} b^{s'} c^r v'^{-t'}, g_2\right)$ holds and output 0 otherwise.

Lemma 1. *The* $\texttt{Randomize}$ *algorithm is perfectly hiding.*

Proof. Firstly, as $t, s \in \mathbb{Z}_p^*$ are randomly selected, $t' = ty$ and $s' = sr$ are uniformly distributed over $\mathbb{Z}_p^*$. Secondly, since $z = ry^{-1}$ is randomly distributed over $\mathbb{Z}_p^*$, b^z and subsequently $v' = (b^z)^{\mathsf{dlog}_b(v)}$ have a uniform distribution over $\mathbb{G}_1$. Lastly, it is clear that there exists an unique $y \in \mathbb{Z}_p^*$ and therefore $z \in \mathbb{Z}_p^*$ for every $\mathsf{dlog}_b(v')$. □

E. Pedersen Commitment Scheme

Pedersen commitment scheme [40] is perfectly hiding and computationally binding under the discrete logarithm assumption. The public parameters $pk = (a, b \in \mathbb{G})$ are based on a

group $\mathbb{G}$ of order p in which the discrete logarithm assumption holds. In order to commit to a message m, one computes

$$C = \texttt{Commit}(pk, m) = a^m b^r$$

where $r \in_R \mathbb{Z}_p^*$ is the opening value.

F. MoniPoly Set Commitment

MoniPoly set commitment (SC) scheme [8], [39] is provably secure under the co-SDH assumption. The public parameters is $pk = ((\mathsf{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p), \{a_i, X_i\}_{0 \le i \le n} = \{g_1^{x'^i}, g_2^{x'^i}\}_{0 \le i \le n})$. The commitment on a message set $A = \{m_1, \ldots, m_{n-1}\} \in \mathbb{Z}_p^{n-1}$ is computed as:

$$C = \texttt{Commit}(pk, A) = a_0^{(x'+o)\prod_{m \in A}(x'+m)} = \prod_{i=0}^{n} a_i^{\mathsf{m}_i}$$

such that (A, o) are the roots for the monic polynomial:

$$(x'+o)(x'+m_1)\cdots(x'+m_{n-1}) = x'^n + \mathsf{m}_n x'^{n-1} + \cdots + \mathsf{m}_0$$

where $\texttt{MPEncode} : \mathbb{Z}_p^n \to \mathbb{Z}_p^{n+1}$ (cf. §O) is the encoding algorithm that transforms the multiplicative form at the left-hand side into the additive form at the right-hand side. In this work, we view a MoniPoly set commitment as:

$$C = \texttt{Commit}(pk, A) = a_0^{o\prod_{m \in A}(x'+m)} = \left(\prod_{i=0}^{n-1} a_i^{\mathsf{m}_i}\right)^o$$

that isolates the opening value $o \in_R \mathbb{Z}_p^*$ from the domain $\mathbb{Z}_p^* \times \mathbb{Z}_p^*$, a feature that is mandatory for our multi-set commitment (cf. §IV-A). The full construction and security analysis for this minor tweak is in Appendix P.

IV. A Multi-Set Commitment Scheme

We instantiate the interface of amulti-set commitment scheme MSC from Section III-C based on the MoniPoly SC scheme [8]. Our overall construction proceeds in three steps:

1) establish a MSC scheme suitable for graphs (§IV-A),
2) encode graphs $\mathcal{G}$ into the multi-set structure (§V), and
3) establish the relational credential system in a commit-and-sign approach (§VI).

The MoniPoly MSC scheme computes a commitment as $C = \prod_{i=1}^{L} C_i$ where every $C_i = a_{i_0}^{o_i \prod_{m \in A_i}(x'+m)}$ is a MoniPoly set commitment SC [8] on a constituent set A_i.

Remark 2. The trivial approach of aggregating the MoniPoly set commitment SC such that

$$C = \prod_{i=1}^{L} a_0^{o_i \prod_{m \in A_i}(x'+m)}$$

is not secure. To be precise, the commitment

$$C = a_0^{\sum_{i=1}^{L} o_i \prod_{m \in A_i}(x'+m)} = a_0^{o^* \prod_{m \in A^*}(x'+m)}$$

is also a commitment for some random (o^*, A^*). This would allow a prover to cheat during the showing protocol, for instance, to prove that she owns a set A^* but not the set A_1 queried by the verifier.

A. Construction

The MoniPoly MSC construction defines the seven algorithms[1] below:

$\texttt{Setup}\ (1^\kappa, L, n)$. Select random generators $g_1 \in_R \mathbb{G}_1, g_2 \in_R \mathbb{G}_2$, secret key $sk = (x', \alpha_1, \ldots, \alpha_L) \in_R \mathbb{Z}_p^*$ and set $\mathfrak{M} = \mathbb{Z}_p^{L \times n}$. Generate the public key as $pk = (\mathsf{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \{\{a_{i_k}, X_{i_k}\}_{k=0}^{n}\}_{i=0}^{L})$ where $\{a_{0_k}, X_{0_k}\}_{0 \le k \le n} = \{g_1^{x'^k}, g_2^{x'^k}\}_{0 \le k \le n}$ and $\{a_{i_k}, X_{i_k}\}_{1 \le i \le L} = \{a_{0_k}^{\alpha_i}, X_{0_k}^{\alpha_i}\}_{1 \le i \le L}$. If L and n are fixed, sk can be discarded after pk is generated.

$\texttt{Commit}\ (pk, A)$. Taking as input a multi-set $A = \{A_1, \ldots, A_L\} \in \mathbb{Z}_p^{L \times n}$, select opening values $O = \{o_1, \ldots, o_L\} \in_R \mathbb{Z}_p^L$ and output the commitment as $C = \prod_{A_i \in A} C_i = \prod_{A_i \in A} a_{i_0}^{o_i \prod_{m \in A_i}(x'+m)}$.

$\texttt{Open}\ (pk, C, A, O)$. Return 1 if

$$\mathsf{e}(C, X_{0_0}) = \prod_{A_i \in A} \mathsf{e}\left(a_{i_0}^{o_i \prod_{m \in A_i \setminus \{m_i\}}(x'+m)}, X_{0_1} X_{0_0}^{m_i}\right)$$

holds and return 0 otherwise for randomly selected $m_i \in A_i$.

$\texttt{OpenIntersection}\ (pk, C, A, O, (A', (\ell, \kappa)))$. Let $A' = \{A'_1, \ldots, A'_L\}$, $1 \le \ell \le L, \kappa = \{k_1, \ldots, k_L\}$ and $I_{i,j} = A'_j \cap A_i$. If there are ℓ-many pair of indexes (i, j) such that $|I_{i,j}| = k_j \le |A'_j|$ holds where indexes i and j are unique, respectively, return these intersecting sets $I_{i,j}$ as the multi-set I and the corresponding witnesses

$$W = \left(\left\{W_i = a_{i_0}^{o_i \prod_{m \in A_i \setminus \{m_i\}}(x'+m)}, m_i\right\}_{A_i \in A \setminus I}, \left\{W_i = a_{i_0}^{o_i \prod_{m \in A_i \setminus I_i}(x'+m)}\right\}_{A_i \in I}\right)$$

where every $m_i \in A_i$ is randomly selected. Otherwise, return an error $\perp$.

$\texttt{VerifyIntersection}\ (pk, C, (I, W), (A', (\ell, \kappa)))$.Return 1 if

$$\mathsf{e}\left(C \prod_{A'_i \in A'} a_{0_0}^{\prod_{m \in A'_i}(x'+m)}, X_0\right) = \prod_{A_i \in A \setminus I} \mathsf{e}\left(W_i, X_{0_1} X_{0_0}^{m_i}\right) \mathsf{e}\left(\prod_{A'_i \in A' \setminus I} a_{0_0}^{\prod_{m \in A'_i}(x'+m)}, X_{0_0}\right) \cdot \prod_{A_i \in I} \mathsf{e}\left(W_i a_{0_0}^{\prod_{m \in A'_i \setminus I_i}(x'+m)}, X_{0_0}^{\prod_{m \in I_i}(x'+m)}\right)$$

holds and return 0 otherwise.

B. Security Properties

Theorem 2 (Hiding)**.** *The MoniPoly multi-set commitment scheme is information-theoretically hiding.*

Proof. The hiding property follows directly from the opening values O being chosen uniformly at random from $\mathbb{Z}_p^*$, rendering the commitment C a random group element in $\mathbb{G}_1$. □

While the construction shares a number of similarity to the MoniPoly set commitment scheme [8], the security proof

[1]We place the OpenDifference and VerifyDifference algorithms in Appendix J as they are utilized by the disjointness predicate of our proposed ReCS which appear in the Appendix M only.

requires a considerate modification. Specifically, the computational binding property is now based on the hardness of co-DLOG problem instead of the q-co-SDH problem as shown in Appendix J.

Theorem 3 (Binding)**.** *The MoniPoly multi-set commitment scheme is computationally binding if the co-DLOG problem is hard.*

Theorem 4 (Intersecting Binding)**.** *The MoniPoly multi-set commitment scheme is intersecting binding if the co-DLOG problem is hard.*

Theorem 5 (Differing Binding)**.** *The MoniPoly multi-set commitment scheme is differing binding if the co-DLOG problem is hard.*

V. Graph to Multi-Set (G2MS) Encoding

The second step of our overall construction is to encode arbitrary graphs as specified in Def. 2 into the multi-set structure established in the preceding Section IV-A. For that, we recall the set representation for graphs from Def. 3, in which a set V_i includes the vertex identifier and all labels for Vertex i and a set $E_{(i,j)}$ includes the corresponding identifiers and labels. Formally, we require the encoding from graph to multi-set fulfils the definition as follows.

Definition 14 (G2MS Encoding)**.** *For a graph $\mathcal{G} = (V \subseteq \mathcal{V}, E \subseteq \mathcal{E}, f_\mathcal{V}, f_\mathcal{E})$ and a defined message space $\mathfrak{M}$, a graph to multi-set encoding $\texttt{G2MS} : \mathcal{G} \rightarrow \mathfrak{M}^{L\times n}$ is a function, which maps vertices $i \in V$ and their labels $f_\mathcal{V}(i)$ as well as edges $(i,j) \in E$ and their labels $f_\mathcal{E}(i,j)$ onto message space elements $m \in \mathfrak{M}$. A G2MS encoding is called* complete *if all graphs from $\mathcal{V}$ and $\mathcal{L}$ can be encoded in $\mathfrak{M}$. A graph encoding is called* unambiguous*, if the encoding is injective.*

There are ways to instantiate a G2MS encoding that meets the requirements defined above. For instance, the prime encoding [33], [35] that based on a list of certified prime numbers which fall in MSC message space $\mathfrak{M}^{L\times n}$ or using an appropriate hash function that can hash into $\mathfrak{M}^{L\times n}$.

For clarity of presentation, we take a simpler approach by assuming vertices and labels are always from the cyclic group $\mathbb{Z}_p^*$, that is, we only consider $\mathcal{G}$ such that its universes $\mathcal{V}$ and $\mathcal{L}$ is in $\mathfrak{M}$. If $\mathfrak{M} = \mathbb{Z}_p^*$ as in our MoniPoly MSC, we have $\mathcal{V} \subset \mathbb{Z}_p^* \setminus \mathcal{L}, \mathcal{L} \subset \mathbb{Z}_p^* \setminus \mathcal{V}$. A vertex with identifier i and associated labels $\{m_1, \ldots, m_n\} = f_\mathcal{V}(i)$ is then represented as a set $V_i = \{i, f_\mathcal{V}\}$ while an edge with identifier (i,j) and associated labels $\{m_1, \ldots, m_n\} = f_\mathcal{E}(i,j)$ is represented as a set $E_{(i,j)} = \{i, j, f_\mathcal{E}\}$. A graph $\mathcal{G} = (V \subseteq \mathcal{V}, E \subseteq \mathcal{E}, f_\mathcal{V}, f_\mathcal{E})$ is thus a multi-set $A = \{\{V_i\}_{i\in V}, \{E_{(i,j)}\}_{(i,j)\in E}\}$.

Theorem 6. *The* G2MS *encoding above is complete and unambiguous.*

Proof. As any vertex i and edge (i,j) can be encoded into a set $V_i \in \mathbb{Z}_p^n$ and a set $E_{(i,j)} \in \mathbb{Z}_p^n$, any graph whose $\mathcal{V} \subset \mathbb{Z}_p^*$ and $\mathcal{E} \subset \mathbb{Z}_p^*$ can be encoded into a multi-set $A \in \mathbb{Z}_p^{L\times n}$. So, the encoding above is complete.

It is clear that when the the universes $\mathcal{V}$ and $\mathcal{L}$ are subset of $\mathbb{Z}_p^*$, the encoding above is injective. □

We now define the related multi-set operations for $A = \texttt{G2MS}(\mathcal{G})$.

Definition 15 (Vertex Intersection)**.** *Given $A = \texttt{G2MS}(\mathcal{G})$ where $\mathcal{G} = (V, E, f_\mathcal{V}, f_\mathcal{E})$ is a graph. Let $V_i = \{i, f_\mathcal{V}(i)\} \in A$ represents the set of vertex identifier and associated labels for a vertex $i \in V$. Let $k_i \leq |V_i'| \leq |V_i|$ be a vertex intersection threshold for the intersecting vertex $\overset{\cap}{V}_i = (V_i' \cap V_i)$. If $|\overset{\cap}{V}_i| = k_i$, $\overset{\cap}{V}_i$ is the intersection of two vertices V_i' and V_i.*

Definition 16 (Vertex Difference)**.** *Given $A = \texttt{G2MS}(\mathcal{G})$ where $\mathcal{G} = (V, E, f_\mathcal{V}, f_\mathcal{E})$ is a graph. Let $V_i = \{i, f_\mathcal{V}(i)\} \in A$ represents the set of vertex identifier and associated labels for a vertex $i \in V$. Let $\bar{k}_i \leq |V_i'| \leq |V_i|$ be a vertex difference threshold for the differing vertex $\bar{V}_i = (V_i' - V_i)$. If $|\bar{V}_i| = \bar{k}_i$, $\bar{V}_i$ is the difference of two vertices V_i' and V_i.*

Definition 17 (Edge Intersection)**.** *Given $A = \texttt{G2MS}(\mathcal{G})$ where $\mathcal{G} = (V, E, f_\mathcal{V}, f_\mathcal{E})$ is a graph. Let $E_{(i,j)} = \{i, j, f_\mathcal{E}(i,j)\} \in A$ represents the set of edge identifier and associated labels for an edge $(i,j) \in E$. Let $k_{(i,j)}$ be an edge intersection threshold for the intersecting edge $\overset{\cap}{E}_{(i,j)} = (E'_{(i,j)} \cap E_{(i,j)})$. If $|\overset{\cap}{E}_{(i,j)}| = k_{(i,j)}$, $\overset{\cap}{E}_{(i,j)}$ is the intersection of two edges $E'_{(i,j)}$ and $E_{(i,j)}$.*

Definition 18 (Edge Difference)**.** *Given $A = \texttt{G2MS}(\mathcal{G})$ where $\mathcal{G} = (V, E, f_\mathcal{V}, f_\mathcal{E})$ is a graph. Let $E_{(i,j)} = \{i, j, f_\mathcal{E}(i,j)\} \in A$ represents the set of vertex identifier and associated labels for a vertex $(i,j) \in E$. Let $\bar{k}_{(i,j)} \leq |E'_{(i,j)}| \leq |E_{(i,j)}|$ be a vertex difference threshold for the differing edge $\bar{E}_{(i,j)} = (E'_{(i,j)} - E_{(i,j)})$. If $|\bar{E}_{(i,j)}| = \bar{k}_{(i,j)}$, $\bar{E}_{(i,j)}$ is the difference of two edges $E'_{(i,j)}$ and $E_{(i,j)}$.*

Definition 19 (Graph Intersection)**.** *Given $A' = \texttt{G2MS}(\mathcal{G}')$ and $A = \texttt{G2MS}(\mathcal{G})$ for two graphs $\mathcal{G}' = \{V', E', f_\mathcal{V}, f_\mathcal{E}\}$ and $\mathcal{G} = \{V, E, f_\mathcal{V}, f_\mathcal{E}\}$ such that $|V'| \leq |V|$ and $|E'| \leq |E|$. Let $(\ell \leq |\mathcal{G}'|, \kappa = \big\{\{k_i\}_{i\in V'}, \{k_{(i,j)}\}_{(i,j)\in E'}\big\})$ be the threshold for $I = \texttt{G2MS}(\overset{\cap}{\mathcal{G}})$ where the intersecting graph is $\overset{\cap}{\mathcal{G}} = \{\overset{\cap}{V}, \overset{\cap}{E}, f_\mathcal{V}, f_\mathcal{E}\} = \mathcal{G} \cap \mathcal{G}'$. If $|I| = \ell$ and furthermore every $|\overset{\cap}{V}_i| = k_i$ and $|\overset{\cap}{E}_{(i,j)}| = k_{(i,j)}$, I is the intersection of A' and A.*

Definition 20 (Graph Difference)**.** *Given $A' = \texttt{G2MS}(\mathcal{G}')$ and $A = \texttt{G2MS}(\mathcal{G})$ for two graphs $\mathcal{G}' = \{V', E', f_\mathcal{V}, f_\mathcal{E}\}$ and $\mathcal{G} = \{V, E, f_\mathcal{V}, f_\mathcal{E}\}$ such that $|V'| \leq |V|$ and $|E'| \leq |E|$. Let $(\bar{\ell} \leq |\mathcal{G}'|, \bar{\kappa} = \big\{\{\bar{k}_i\}_{i\in V'}, \{\bar{k}_{(i,j)}\}_{(i,j)\in E'}\big\})$ be the threshold for $D = \texttt{G2MS}(\bar{\mathcal{G}})$ where the differing graph is $\bar{\mathcal{G}} = \{\bar{V}, \bar{E}, f_\mathcal{V}, f_\mathcal{E}\} = \mathcal{G}' - \mathcal{G}$. If $|D| = \bar{\ell}$ and furthermore every $|\bar{V}_i| = \bar{k}_i$ and $|\bar{E}_{(i,j)}| = \bar{k}_{(i,j)}$, D is the difference of A' and A.*

VI. A Relational Credential System

This third step of our construction following a commit-and-sign approach, establishes the relational credential system (ReCS).

A. Construction

We instantiate the ReCS interface defined in Section II-A to consist of the five interactive algorithms: $\{\texttt{Setup}, \texttt{Obtain}, \texttt{Sign}, \texttt{Prove}, \texttt{Verify}\}$.

1) Setup: This can be viewed as the combination of parameter setup for MoniPoly MSC (cf. §IV-A) and CL signature scheme (cf. §III-D).

$\texttt{Setup}(1^\kappa, L, n)$**:** Select random generators $\{a_{i_0}\}_{i=0}^{L}, b, c \in_R \mathbb{G}_1$, $\{X_{i_0}\}_{i=0}^{L} \in_R \mathbb{G}_2$ and two secret values $x, x' \in_R \mathbb{Z}_p^*$. Output the master public key $mpk = ((\mathsf{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p), b, c, \{\{a_{i_k} = a_{i_0}^{x'^k}, X_{i_k} = X_{i_0}^{x'^k}\}_{i=0}^{L}\}_{k=0}^{n}, X = X_{0_0}^{x}, \mathcal{V}, \mathcal{L})$ and keep the master secret key $msk = (x', x)$ where $\mathcal{V}$ is the vertex identifier universe and $\mathcal{L}$ is the label universe.

2) Issuing Protocol: The signing protocol enables a user U who holds a hidden graph $\mathcal{G}^{\mathtt{U}}$ to obtain a credential sig from the issuer I for the graph $\mathcal{G} = (\mathcal{G}^{\mathtt{U}} \cup \mathcal{G}^{\mathtt{I}})$ where $\mathcal{G}^{\mathtt{I}}$ is a signer-defined graph and $\mathcal{G} = (V, E, f_{\mathcal{V}}, f_{\mathcal{E}})$ is a graph.

$(\texttt{Obtain}(mpk, \mathcal{G}^{\mathtt{U}}) \leftrightarrow \texttt{Issue}(mpk, msk, \mathcal{G}^{\mathtt{I}}))$:

1) Firstly, the user U chooses a random $s' \in_R \mathbb{Z}_p^*$ and proves his hidden graph $A = \texttt{G2MS}(\mathcal{G}^{\mathtt{U}})$ to the signer S by running the zero-knowledge protocol below:

$$PK\Bigg\{((\forall i \in V : \varepsilon_{i_0}, \ldots, \varepsilon_{i_{n_i}}), (\forall (i,j) \in E : \varepsilon_{(i,j)_0}, \ldots, \varepsilon_{(i,j)_{n_{(i,j)}}}), \rho) : M = \prod_{i \in V} \prod_{k=0}^{n_i} a_{i_k}^{\varepsilon_{i_k}} \prod_{(i,j) \in E} \prod_{k=0}^{n_{(i,j)}} a_{(i,j)_k}^{\varepsilon_{(i,j)_k}} b^{\rho} \Bigg\} \quad (1)$$

where clause (1) proves the knowledge of $b^\rho = b^{s'}$ and A such that

$$\prod_{i \in V} \prod_{k=0}^{n_i} a_{i_k}^{\varepsilon_{i_k}} \prod_{(i,j) \in E} \prod_{k=0}^{n_{(i,j)}} a_{(i,j)_k}^{\varepsilon_{(i,j)_k}} = \prod_{i \in V} a_{i_0}^{(x'+i)\prod_{m \in f_{\mathcal{V}}(i)}(x'+m)} \prod_{(i,j) \in E} a_{(i,j)_0}^{(x'+i)(x'+j)\prod_{m \in f_{\mathcal{E}}(i)}(x'+m)}$$

is a multi-set commitment $\texttt{MSC.Commit}(mpk, A)$ with $O = \{1\}^{|A|}$.

2) If the proof is verified, I computes a pre-signature $\sigma' = \texttt{CL.Sign}(mpk, msk, \texttt{G2MS}(\mathcal{G}))$ where $\mathcal{G} = \mathcal{G}^{\mathtt{U}} \cup \mathcal{G}^{\mathtt{I}}$. To be precise, I randomly selects $s'', t \in \mathbb{Z}_p^*$ to sign on M and $C = \texttt{MPC.Commit}(mpk, B)$ as $v = \left(MCb^{s''}c\right)^{(x+t)^{-1}}$ where $B = \texttt{G2MS}(\mathcal{G}^{\mathtt{I}})$ and $O = \{1\}^{|B|}$. I returns $(\sigma' = (t, s'', v), \mathcal{G}^{\mathtt{I}})$ to U.

3) If σ' is a valid CL signature on $\mathcal{G} = \mathcal{G}^{\mathtt{U}} \cup \mathcal{G}^{\mathtt{I}}$, U completes the pre-signature σ' with its own randomness to obtain the final credential $cred = (\sigma = (t, s = s' + s'', v), \mathcal{G})$.

3) Showing Protocol: Essentially, possession is a compound proof of the PoK protocols for CL.VerifyRand algorithm (cf. §III-D) and MSC.Open algorithm (cf. §IV). Let the randomized graph signature be $\sigma' = (r, y, t' = ty, s' = sr, v' = v^{ry^{-1}}) = \texttt{CL.Randomize}(mpk, \sigma)$, obtained through the CL-signature randomization algorithm where $r, y \in_R \mathbb{Z}_p^*$. The prover selects $\{r_i, r_{(i,j)}\} \in_R \mathbb{Z}_p^*$ and runs the showing protocol for possession as follows:

$(\mathsf{Prove}(mpk, \sigma, \texttt{possession}) \leftrightarrow \mathsf{Verify}(mpk, \texttt{possession}))$:

$$PK\Bigg\{(\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}), (\forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), \rho, \omega, \tau, \gamma) : \prod_{i \in V} \mathsf{e}\left(W_i, X_{0_1}^{\varepsilon_{i_1}} X_{0_0}^{\varepsilon_{i_0}}\right) \prod_{(i,j) \in E} \mathsf{e}\left(W_{(i,j)}, X_{0_1}^{\varepsilon_{(i,j)_1}} X_{0_0}^{\varepsilon_{(i,j)_0}}\right) \cdot \quad (2)$$

$$\mathsf{e}\left(b^\rho c^\omega v'^{-\tau}, X_{0_0}\right) = \mathsf{e}(v'^\gamma, X)\Bigg\} \quad (3)$$

where the witnesses

$$\left\{W_i = a_{i_0}^{r_i^{-1} r \prod_{m \in f_{\mathcal{V}}(i)}(x'+m)}, W_{(i,j)} = a_{(i,j)_0}^{r_{(i,j)}^{-1} r (x'+j) \prod_{m \in f_{\mathcal{E}}(i,j)}(x'+m)}\right\}$$

in line (2) correspond to the committed graph $\mathcal{G}$ in the credential where $X_{0_1}^{\varepsilon_{i_1}} X_{0_0}^{\varepsilon_{i_0}} = X_{0_0}^{r_i(x'+i)}$ and $X_{0_1}^{\varepsilon_{(i,j)_1}} X_{0_0}^{\varepsilon_{(i,j)_0}} = X_{0_0}^{r_{(i,j)}(x'+i)}$. Next, in line (3), $v' = v^{ry^{-1}}$ are the public input for the randomized signature σ' where $b^\rho c^\omega v'^{-\tau} = b^{rs} c^r v'^{-ty}$ and $v'^\gamma = v'^y$.

As possession predicate is a compound proofs with PoK for MSC.Open and the PoK for a CL.VerifyRand in a *plug-and-play* manner, additional proofs can be modularly added to the compound proofs to answer complex queries as shown in Section VII.

B. Security Properties

Theorem 7 (PUNL-ACA)**.** *The proposed relational credential system ReCS is protocol unlinkable under active and concurrent attacks.*

We prove Theorem 7 in Appendix M.

Theorem 8 (IMP-ACA)**.** *The proposed relational credential system ReCS is secure against impersonation under active and concurrent attacks, if the co-SDH problem is intractable.*

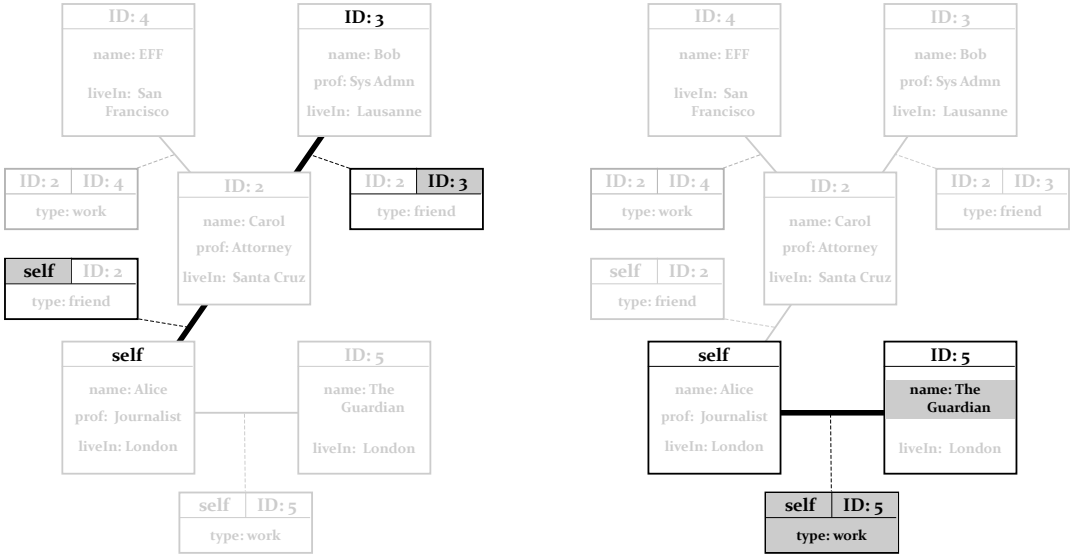
We restate this theorem precisely in Theorem 15 and prove its security in Appendix M.

VII. A Proof System on Social Graphs

In this section, we complement the relational credential system with a zero-knowledge proof system that operates on social graphs. We focus on the vital predicates for a social graph application introduced in Example 1 and Figure 2, connection through multiple hops (§VII-A) and graph coverage (§VII-B). We include more predicate types in Appendix M.

A. Graph Connection Proof

The purpose of the connection predicate is to show that there exists a path in the signed graph $\mathcal{G}$ with at most ℓ hops between two vertices. The predicate $\texttt{connected}_{(i^*, j^*, \ell)}$ verifies such connectivity by proving that there exists a secret sequence of sequential edges $E' = \{E'_1, \ldots, E'_\ell\} \subset \mathcal{G}$ that connects i^* and j^*. Let $I = \{(i^*, j_1), (j_1, j_2), \ldots, (j_{\ell-1}, j^*)\}$ be the edge



(a) Connection $\mathsf{connected}_{(i,j,\ell)}$ (b) Label coverage $\mathsf{cover}_{(\mathcal{G}',1)}$

Fig. 2: Graph proof predicates

identifiers for all the edges in E'. We place the detailed zero-knowledge proof for `connected` predicate in Appendix M and describe the brief idea here:

$$PK\Bigg\{(\sigma', \mathcal{G}, E', I) : \mathsf{CL.VerifyRand}(mpk, \sigma', \mathcal{G}) = 1 \wedge \tag{4}$$

$$\texttt{MSC.VerifyIntersection}(mpk, C, (E', W), (E', (\ell, \kappa))) = 1 \wedge \tag{5}$$

$$\texttt{MSC.VerifyIntersection}(mpk, C_{E'}, (I, W_I), (I, (|I|, 2))) = 1 \wedge \tag{6}$$

$$\texttt{MSC.VerifyIntersection}(mpk, C_{\mathcal{V}}, (I, W_{\mathcal{V}}), (I, |I|)) = 1\Bigg\} \tag{7}$$

where clause (4) verifies the validity of the randomized signature $\sigma' = \texttt{CL.Randomize}(mpk, \sigma)$ and clause (5) confirms the existence of a set of edges E' in $\mathcal{G}$ by making use of the `MSC` intersection algorithms such that $|E'_l| = k_l \in \kappa$ and $C = \texttt{MSC.Commit}(mpk, \mathcal{G})$. Similarly, clause (6) extracts all edge identifiers I from E' by setting the intersection thresholds as $k_1 = 2, \ldots, k_{|I|} = 2$ for every set in I where $C_{E'} = \texttt{MSC.Commit}(mpk, E')$. Clause (7) is the set membership proof for I in which $C_{\mathcal{V}} = a_{0_0}^{\prod_{i \in \mathcal{V}}(x'+i)} = \mathsf{MSC.Commit}(mpk, \mathcal{V})$ is the MoniPoly `MSC` commitment on all vertices in the vertex universe $\mathcal{V}$.

Example 2 (Friend-of-a-Friend). We can use the $\texttt{connected}_{(i^*, j^*, \ell=2)}$ predicate to prove a FOAF statement as introduced with the scenario in Example 1.

```
prove you are connected to my node (≤ two hops).
```

Figure 2a depicts how the consecutive edges between Alice and Bob are proven, without revealing intermediate vertices.

B. Graph Coverage Proof

The purpose of the graph coverage proof is to show that the prover's $\mathcal{G}$ holds vertex, edge and label constellations the verifier requires. In formal terms, the graph coverage predicate $\texttt{cover}_{(\mathcal{G}', \ell)}$ shows that a graph $\mathcal{G}' = (V', E', f_{\mathcal{V}}, f_{\mathcal{E}})$ specified in the verifier's predicate intersects with the prover's signed graph $\mathcal{G} = (V, E, f_{\mathcal{V}}, f_{\mathcal{E}})$ at ℓ secret vertices i and/or edges (i, j). Let $I = \mathcal{G}' \cap \mathcal{G}$, the detailed zero-knowledge proof can be found in Appendix M in which the idea behind the proof is as follows:

$$PK\Bigg\{(\sigma', \mathcal{G}, I, \mathcal{G}') : \texttt{CL.VerifyRand}(mpk, \sigma', \mathcal{G}) = 1 \wedge \tag{8}$$

$$\texttt{MSC.VerifyIntersection}(mpk, C, (I, W), (\mathcal{G}', (\ell, \kappa))) = 1 \wedge \tag{9}$$

$$\texttt{MSC.Open}(mpk, C', \mathcal{G}', O') = 1 \wedge \tag{10}$$

$$\texttt{MSC.VerifyIntersection}(mpk, C', (I, W_I), (I, (\ell, \kappa))) = 1\Bigg\} \tag{11}$$

where clause (8) works the same as clause (4) while clause (9) shows the existence of I in the signed graph $\mathcal{G}$ such that $|I_l| = k_l \in \kappa$. Lastly, clause (11) confirms the same I that appears in clause (8) exists in $C' = \texttt{MSC.Commit}(mpk, \mathcal{G}')$ whose correctness is verified by the clause (10). Referring to the running example and Figure 2, let us consider a query inquiring about labels:

Example 3 (Label Coverage). Bob could require that being a member of staff of any specified trusted newspaper:

```
prove that you are with The Guardian (type: work) ∨
prove that you are with The WP (type: work) ∨
prove that you are with Der Spiegel (type: work).
```

This example uses $\mathsf{cover}_{(\mathcal{G}',\ell=1)}(\mathcal{G})$ predicate with

$$\mathcal{G}' = \Big\{ V' = \big\{\{\mathtt{self}\}, \{\mathsf{ID}:5, \mathsf{name}:\mathsf{TheGuardian}\}, \{\mathsf{ID}:5, \mathsf{name}:\mathsf{TheWP}\}, \{\mathsf{ID}:5, \mathsf{name}:\mathsf{DerSpiegel}\}\big\}$$
$$E' = \big\{\{\mathtt{self}, \mathsf{ID}:5, \mathsf{type}:\mathsf{work}\}\big\} \Big\}.$$

To connect to Bob's service provider, Alice proves that she is a member of staff from one of the three newspapers (cf. Figure 2b).

VIII. Performance Evaluation

We compare the proof complexity and implementation performance in Figure 3.

A. Proof Complexity

In Figure 3a, we compare the computational protocol complexity of the new scheme to the prior SRSA graph signature scheme by Groß [32]. For that, we use the number of point multiplications M_1 in $\mathbb{G}_1$ (resp. M_x for $\mathbb{G}_x$) as a benchmarking unit. We set the conversion parameters at 128-bit security level [8] to $1M_2 = 2M_1, 1M_T = 6M_1$, a pairing $1P = 9M_1$ and a modular exponentiation in SRSA $1E = 5M_1$. We assume that every vertex $i \in V$ and edge $(i,j) \in E$ has the same number of labels $\ell = 10$. The number of edges is set to be twice the number of vertices, $|E| = 2|V|$. Figure 3a shows that the computational complexity of the new q-SDH graph signature scheme is consistently lower than that of the SRSA-based graph signature scheme [32]. This holds, unless the number of labels per vertex/edge is large (> 50). For the relational credential system, our scheme will always be faster, because the numbers of labels per graph element is generally small (≤ 20). The proof size is proportional to the proof complexity.

B. Implementation Benchmarks

We have implemented the ReCS and the underlying q-SDH graph signature scheme in Java, based on the elliptic curve and pairing cryptography implementation of the Apache Milagro Crypto Library (AMCL), v. 3.2. The implementation is based on the BLS461 curve that offers a 128-bit security for our ReCS.

We conducted performance experiments on actual social graphs extracted from Facebook, based on the Stanford Snap dataset [41]. This dataset was obtained by traversing the social graph of eight user accounts. From this database, we randomly selected users and computed their user-specific social graphs. The graph-size distribution is heavily positively skewed with a $M_{|\mathcal{G}|} = 451$, $Mdn_{|\mathcal{G}|} = 94.5$ and $SD_{|\mathcal{G}|} = 770$.

Subsequently, we computed 10 warm-up rounds and then 50 evaluation rounds of the issuing and proof system protocols per given graph size. The performance evaluation was computed on an Intel Core i7-2600S CPU with 2.80GHz, with 4 cores, while the experiment was restricted to a single core. The machine had 8 GB RAM and was operating under Ubuntu 16.04 with kernel version 4.15 and the Oracle JDK v. 1.8.0_201. We chose a random friend-of-a-friend in the user's social network and computed connectivity proofs to establish the FOAF relationship. While the performance is linear in the overall graph size as predicted, we display the results in Figure 3b in *square-root scale* to better visualize the majority of small social graphs. The mean execution time heavily positively skewed, with $M_t = 6$s, $Mdn_t = 1.358$s and $SD_t = 10.3$s.

IX. Conclusion

We proposed the new concept of a relational credential system, a privacy-enhancing technology that enables users to gain access to resources not only based on the possession of credentials or statements on private attributes, but also based on predicates on a private social graph. We designed the first construction of such a system, based on a new construction for a q-SDH graph signature scheme, which enables the representation of the social graph in the credential and features an efficient and expressive zero-knowledge proof system thereon. This key ingredient of a new graph signature scheme can be of independent interest. We proved the unlinkability and impersonation resilience of this scheme in the standard model with tight reduction. We believe that our construction will open up new application cases for anonymous credential systems and, thereby, make this mainstay privacy-enhancing technology even more useful.

References

[1] D. Chaum, "Security without identification: Transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, Oct. 1985. [Online]. Available: http://doi.acm.org/10.1145/4372.4373

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

[3] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology — EUROCRYPT 2001*, B. Pfitzmann, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 93–118.

[4] ——, "A signature scheme with efficient protocols," in *Security in Communication Networks*, S. Cimato, G. Persiano, and C. Galdi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 268–289.

[5] ——, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology – CRYPTO 2004*, M. Franklin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 56–72.

[6] E. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, ser. CCS '04. New York, NY, USA: ACM, 2004, pp. 132–145. [Online]. Available: http://doi.acm.org/10.1145/1030083.1030103

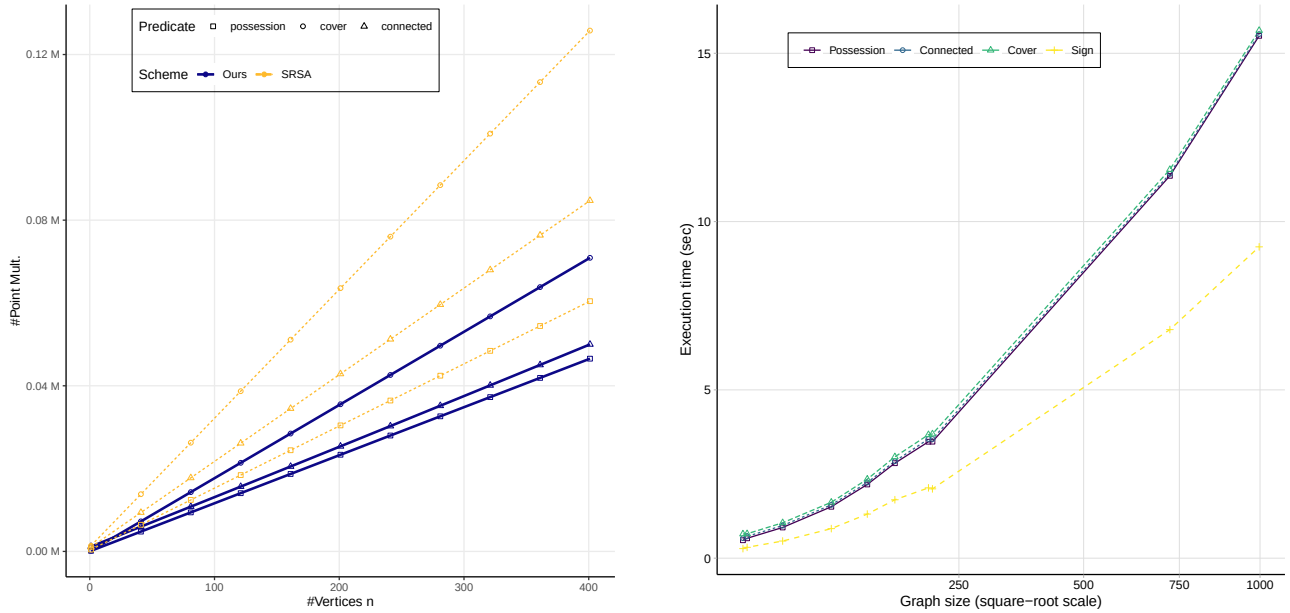
[7] G. Fuchsbauer, C. Hanser, and D. Slamanig, "Structure-preserving signatures on equivalence classes and constant-size anonymous credentials," *Journal of Cryptology*, vol. 32, no. 2, pp. 498–546, Apr 2019. [Online]. Available: https://doi.org/10.1007/s00145-018-9281-4

[8] S.-Y. Tan and T. Groß, "Monipoly—an expressive q-SDH-based anonymous attribute-based credential system," in *ASIACRYPT 2020*, May 2020, Cryptology ePrint Archive Report.

[9] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *Advances in Cryptology — CRYPTO 2002*, M. Yung, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 61–76.

[10] F. Baldimtsi, J. Camenisch, M. Dubovitskaya, A. Lysyanskaya, L. Reyzin, K. Samelin, and S. Yakoubov, "Accumulators with applications to anonymity-preserving revocation," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, April 2017, pp. 301–315.

[11] J. Camenisch and T. Groß, "Efficient attributes for anonymous credentials," in *Proceedings of the 15th ACM conference on Computer and communications security*. ACM, 2008, pp. 345–356.



(a) Complexity of SRSA vs. q-SDH

(b) Social graph performance (square-root scale)

Fig. 3: Proof complexity and performance on social graphs, BLS461/128-bit

[12] J. Camenisch and T. Groß, "Efficient attributes for anonymous credentials," *ACM Trans. Inf. Syst. Secur.*, vol. 15, no. 1, pp. 4:1–4:30, Mar. 2012. [Online]. Available: http://doi.acm.org/10.1145/2133375.2133379

[13] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham, "Randomizable proofs and delegatable anonymous credentials," in *Annual International Cryptology Conference*. Springer, 2009, pp. 108–125.

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

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

[16] IBM, "Specification of the identity mixer cryptographic library, v. 2.3.40." Specification, IBM Research, 2013.

[17] R. Okishima and T. Nakanishi, "An anonymous credential system with constant-size attribute proofs for cnf formulas with negations," in *Advances in Information and Computer Security*. Springer Verlag, 2019, pp. 89–106.

[18] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-taa," in *Security and Cryptography for Networks*, R. De Prisco and M. Yung, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 111–125.

[19] S. Schäge, "Tight proofs for signature schemes without random oracles," in *Advances in Cryptology – EUROCRYPT 2011*, K. G. Paterson, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 189–206.

[20] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Advances in Cryptology - ASIACRYPT 2010*, M. Abe, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 177–194.

[21] K. Y. Chan and T. H. Yuen, "Attribute-based anonymous credential: Optimization for single-use and multi-use," in *Cryptology and Network Security*, A. R. Beresford, A. Patra, and E. Bellini, Eds. Cham: Springer International Publishing, 2022, pp. 89–121.

[22] C. Papamanthou, R. Tamassia, and N. Triandopoulos, "Optimal verification of operations on dynamic sets," in *Advances in Cryptology – CRYPTO 2011*, P. Rogaway, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 91–110.

[23] R. Canetti, O. Paneth, D. Papadopoulos, and N. Triandopoulos, "Verifiable set operations over outsourced databases," in *Public-Key Cryptography – PKC 2014*, H. Krawczyk, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 113–130.

[24] Y. Zhang, J. Katz, and C. Papamanthou, "An expressive (zero-knowledge) set accumulator," in *2017 IEEE European Symposium on Security and Privacy (EuroS P)*, April 2017, pp. 158–173.

[25] Y. Zhang, C. Papamanthou, and J. Katz, "Alitheia: Towards practical verifiable graph processing," in *In CCS*, 2014.

[26] K. Mandal, B. Alomair, and R. Poovendran, "Outsourcing graph databases with label-constrain query verification," 2015, http://labs.ece.uw.edu/nsl/papers/GDB-full-15.pdf.

[27] Y. Zhang, C. Papamanthou, and J. Katz, "Verifiable graph processing," *ACM Trans. Priv. Secur.*, vol. 21, no. 4, pp. 20:1–20:23, Oct. 2018. [Online]. Available: http://doi.acm.org/10.1145/3233181

[28] C. Lin, D. He, X. Huang, M. K. Khan, and K.-K. R. Choo, "A new transitively closed undirected graph authentication scheme for blockchain-based identity management systems," *IEEE Access*, vol. 6, pp. 28 203–28 212, 2018.

[29] L. Wang, Y. Tian, and D. Zhang, "Toward cross-domain dynamic accumulator authentication based on blockchain in internet of things," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 4, pp. 2858–2867, 2021.

[30] S. Micali and R. L. Rivest, "Transitive signature schemes," in *Topics in Cryptology-CT–RSA 2002*. Springer, 2002, pp. 236–243.

[31] M. Bellare and G. Neven, "Transitive signatures based on factoring and rsa," in *Advances in Cryptology–ASIACRYPT 2002*. Springer, 2002, pp. 397–414.

[32] T. Groß, "Signatures and efficient proofs on committed graphs and NP-statements," in *Financial Cryptography and Data Security*, R. Böhme and T. Okamoto, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 293–314.

[33] ——, "Efficient certification and zero-knowledge proofs of knowledge on infrastructure topology graphs," in *Proceedings of the 6th Edition of the ACM Workshop on Cloud Computing Security*, ser. CCSW '14. New York, NY, USA: Association for Computing Machinery, 2014, pp. 69–80. [Online]. Available: https://doi.org/10.1145/2664168.2664175

[34] T. Groß, "Hashing to prime in zero-knowledge," in *Proceedings of the 18th International Conference on Security and Cryptography (SECRYPT 2021)*, 2021, pp. 62–74.

[35] 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.

[36] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Advances in Cryptology — CRYPTO '97*, B. S. Kaliski, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 410–424.

[37] S. Chatterjee and A. Menezes, "On cryptographic protocols employing asymmetric pairings - the role of ψ revisited," *Discrete Applied Mathematics*, vol. 159, no. 13, pp. 1311 – 1322, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166218X11001648
[38] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology – CRYPTO 2004*, M. Franklin, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 41–55.
[39] S.-Y. Tan and T. Groß, "Monipoly—an expressive q-SDH-based anonymous attribute-based credential system [extended version]," IACR, Cryptology ePrint Archive Report 2020/587 (ia.cr/2020/587), May 2020.
[40] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, ser. Lecture Notes in Computer Science, J. Feigenbaum, Ed., vol. 576. Springer, 1991, pp. 129–140. [Online]. Available: https://doi.org/10.1007/3-540-46766-1_9
[41] J. Leskovec and J. Mcauley, "Learning to Discover Social Circles in Ego Networks," *Advances in neural information processing systems*, vol. 25, 2012.
[42] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, "Relations among notions of security for public-key encryption schemes," in *Advances in Cryptology — CRYPTO '98*, H. Krawczyk, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 26–45.
[43] D. Boneh and X. Boyen, "Short signatures without random oracles and the sdh assumption in bilinear groups," *Journal of Cryptology*, vol. 21, no. 2, pp. 149–177, Apr 2008. [Online]. Available: https://doi.org/10.1007/s00145-007-9005-7
[44] E. Kiltz, D. Masny, and J. Pan, "Optimal security proofs for signatures from identification schemes," in *Advances in Cryptology – CRYPTO 2016*, M. Robshaw and J. Katz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 33–61.

APPENDIX

We include a brief definition of the oracles used in the security proofs in Table III.

Before presenting the analysis, we first present the security games that define IUNL−ACA, SUNL−ACA and GUNL−ACA.

A. IUNL−ACA

The privacy goal of a secure issuing protocol is to hide the user's secret graph $\mathcal{G}^{\mathtt{U}}$: an adversary should not be able to learn from the interaction with a user about $\mathcal{G}^{\mathtt{U}}$. We define this notion as issuing unlinkability (IUNL) under active and concurrent attack (ACA) through the security game defined in Table IV.

Definition 21 (IUNL−ACA)**.** *Let* $ReCS = (\mathtt{Setup}, (\mathtt{Obtain}, \mathtt{Issue}), (\mathtt{Prove}, \mathtt{Verify}))$ *be an relational credential system and let* $\mathcal{A}$ *be an adversary. Let*

$$\varepsilon_{\mathsf{iunl}} = \Pr\left[\mathbf{Game}^{\mathsf{iunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}\right] - \frac{1}{2},$$

we say that a ReCS *is* IUNL−ACA *secure if* $\varepsilon_{\mathsf{iunl}}$ *is negligible.*

B. SUNL−ACA

The privacy goal of a secure showing protocol is to hide the user credential $cred$: an adversary should not be able to learn from the interaction with a prover about $cred$. We define this notion as showing unlinkability (SUNL) under active attack (ACA) through the security game defined in Table V.

Definition 22 (SUNL−ACA)**.** *Let* $ReCS = (\mathtt{Setup}, (\mathtt{Obtain}, \mathtt{Issue}), (\mathtt{Prove}, \mathtt{Verify}))$ *be an attribute credential system and let* $\mathcal{A}$ *be an adversary. Let*

$$\varepsilon_{\mathsf{sunl}} = \Pr\left[\mathbf{Game}^{\mathsf{sunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}\right] - \frac{1}{2},$$

we say that ACS *is* SUNL−ACA *secure if* $\varepsilon_{\mathsf{sunl}}$ *is negligible.*

C. GUNL−ACA

The privacy goal of a secure showing protocol is to hide the graphs $(\mathcal{G}^{\mathtt{U}}, \mathcal{G}^{\mathtt{I}})$ in the credential: an adversary should not be able to learn from the interaction with a prover about $(\mathcal{G}^{\mathtt{U}}, \mathcal{G}^{\mathtt{I}})$. We define this notion as graph unlinkability (GUNL) under active and concurrent attack (ACA) through the security game in Table VI.

Definition 23 (GUNL−ACA)**.** *Let* $ReCS = (\mathtt{Setup}, (\mathtt{Obtain}, \mathtt{Issue}), (\mathtt{Prove}, \mathtt{Verify}))$ *be an relational credential system and let* $\mathcal{A}$ *be an adversary. Let*

$$\varepsilon_{\mathsf{gunl}} = \Pr\left[\mathbf{Game}^{\mathsf{gunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}\right] - \frac{1}{2},$$

we say that ReCS *is* GUNL−ACA *secure if* $\varepsilon_{\mathsf{gunl}}$ *is negligible.*

TABLE III: Oracle specifications.

$\mathcal{O}_{\texttt{Issue}}(mpk, msk, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, sid)$ If $(\mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, sid) \notin IS$, then $IS \leftarrow IS \cup (\mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, sid)$ $(\pi, \sigma) \leftarrow \mathsf{U}(sid, mpk, \mathcal{G}^{\mathsf{U}})$ $HU \leftarrow HU \cup \{sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \sigma, \pi\}$ Return $(cred, \pi)$	$\mathcal{O}_{\texttt{Issuing}}(mpk, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, sid)$ If $(sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \cdot, \pi) \in HU \cup CU$, then return $\{\pi\}$ $(\cdot, \pi) \leftarrow \mathcal{O}_{\texttt{Issue}}(mpk, msk, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, sid)$ Return (π)

$\mathcal{O}_{\texttt{Verify}}(mpk, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \phi, sid)$
If $(\mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \phi, sid) \notin SS$, then
$SS \leftarrow SS \cup (\mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \phi, sid)$
If $(sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \cdot, \cdot) \notin CU \cup HU$, then
If $(sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \cdot, \cdot, \cdot) \notin CPV \cup HPV$, then
$(cred, \cdot) \leftarrow \mathcal{O}_{\texttt{Issue}}(mpk, msk, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, sid)$
$(\tilde{\pi}, 1) \leftarrow \mathsf{P}(sid, mpk, cred, \phi)$
$HPV \leftarrow HPV \cup \{sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, cred, \phi, \tilde{\pi}\}$
Return $(\tilde{\pi})$
	$\mathcal{O}_{\texttt{Showing}}(mpk, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \phi, sid)$
	If $(\mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \phi, sid) \notin SS$, then
	$SS \leftarrow SS \cup (\mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \phi, sid)$
	If $(sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \cdot, \phi, \tilde{\pi}) \in CPV \cup HPV$, then
	return $\{\tilde{\pi}\}$
	$(\tilde{\pi}) \leftarrow \mathcal{O}_{\texttt{Verify}}(mpk, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, \phi, sid)$
	Return $(\tilde{\pi})$

$\mathcal{O}_{\texttt{CorrU}}(\pi)$ If $(\cdot, \cdot, \cdot, \cdot, \pi) \notin HU$, return $\perp$ $HU \leftarrow HU \setminus \{sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, cred, \pi\}$ $CU \leftarrow CU \cup \{sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, cred, \pi\}$ Return $(cred)$	$\mathcal{O}_{\texttt{CorrP}}(\tilde{\pi})$ If $(\cdot, \cdot, \cdot, \cdot, \cdot, \tilde{\pi}) \notin HPV$, return $\perp$ $HPV \leftarrow HPV \setminus \{sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, cred, \phi, \tilde{\pi}\}$ $CPV \leftarrow CPV \cup \{sid, \mathcal{G}^{\mathsf{U}}, \mathcal{G}^{\mathsf{I}}, cred, \phi, \tilde{\pi}\}$ Return $(cred)$

Note: IS : issuing session, SS : showing session, U: user, P: prover, (CU, CPV) : corrupted user list, (HU, HPV) : honest user list, π : signing transcript, $\tilde{\pi}$: showing transcript

TABLE IV: The $\texttt{IUNL} - \texttt{ACA}$ security game.

$\mathbf{Game}^{\text{iunl-aca}}_{\mathsf{ReCS},\mathcal{A}}$

$IS \leftarrow \emptyset; SS \leftarrow \emptyset; CU \leftarrow \emptyset;\ HU \leftarrow \emptyset$
$\mathcal{O}(\cdot) \leftarrow \{\mathcal{O}_{\texttt{Issue}}(\cdot), \mathcal{O}_{\texttt{CorrU}}(\cdot)\}$
$b \leftarrow \{0,1\};\ (mpk, msk) \leftarrow \texttt{Setup}(1^k);$
$(\mathcal{G}^{\mathsf{U}}_0, \mathcal{G}^{\mathsf{U}}_1) \leftarrow \mathcal{A}^{\mathcal{O}}(mpk, msk)$
If $|\mathcal{G}^{\mathsf{U}}_0| \neq |\mathcal{G}^{\mathsf{U}}_1|$, then return Lose
$(\pi_b, cred_b) \leftarrow (\texttt{Obtain}(mpk, \mathcal{G}^{\mathsf{U}}_b) \leftrightarrow \mathcal{A}^{\mathcal{O}}(\mathcal{G}^{\mathsf{U}}_0, \mathcal{G}^{\mathsf{U}}_1, \mathcal{G}^{\mathsf{I}}_b, \mathcal{G}^{\mathsf{I}}_{1-b}))$
$(\pi_{1-b}, cred_{1-b}) \leftarrow (\texttt{Obtain}(mpk, \mathcal{G}^{\mathsf{U}}_{1-b}) \leftrightarrow \mathcal{A}^{\mathcal{O}}(\mathcal{G}^{\mathsf{U}}_0, \mathcal{G}^{\mathsf{U}}_1, \mathcal{G}^{\mathsf{I}}_b, \mathcal{G}^{\mathsf{I}}_{1-b}))$
If $|\mathcal{G}^{\mathsf{I}}_b| \neq |\mathcal{G}^{\mathsf{I}}_{1-b}|$, then return Lose
$b' \leftarrow \mathcal{A}^{\mathcal{O}}((\mathcal{G}^{\mathsf{U}}_0, \mathcal{G}^{\mathsf{U}}_1), (\mathcal{G}^{\mathsf{I}}_b, \mathcal{G}^{\mathsf{I}}_{1-b}), (\pi_b, \pi_{1-b}))$
If $(\cdot, \cdot, \cdot, \cdot, \pi_b) \in CU \vee (\cdot, \cdot, \cdot, \cdot, \pi_{1-b}) \in CU$, then return Lose
If $b = b'$, then return Win, else return Lose

Note: $(\mathcal{G}^{\mathsf{U}}_0, \mathcal{G}^{\mathsf{U}}_1)$: challenge graphs, $(\mathcal{G}^{\mathsf{I}}_0, \mathcal{G}^{\mathsf{I}}_1)$: issuer-assigned graphs, (π_0, π_1) : issuing transcript, IS: issuing session, SS: showing session, CU: corrupted user list, HU: honest user list

D. Relating the Security Notions

Adopting Bellare et al.'s approach [42], for each pair of notions

$$\mathbf{A}, \mathbf{B} \in \{\texttt{IUNL} - \texttt{ACA}, \texttt{SUNL} - \texttt{ACA}, \texttt{GUNL} - \texttt{ACA}, \texttt{PUNL} - \texttt{ACA}\},$$

we show one of the following:

1) Implication ($\mathbf{A} \Rightarrow \mathbf{B}$) : A proof that if ReCS is any relational credential system meeting notion of security **A** then ReCS also meets notion of security **B**.
2) Separation ($\mathbf{A} \not\Rightarrow \mathbf{B}$) : A construction of a ReCS that provably meets notion of security **A** but provably does not meet notion of security **B**.

Figure 4 is the summary of the relations among the security notions discussed in this work. We claim that for any pair of notions **A,B**, it is the case that **A** implies **B** if and only if there is a path from **A** to **B** in the graph. For instance, we claim that IUNL−ACA does not imply PUNL−ACA. If we have IUNL−ACA implies PUNL−ACA, then pair with GUNL−ACA implying SUNL−ACA would give IUNL−ACA implying SUNL−ACA, which has been proven to be false.

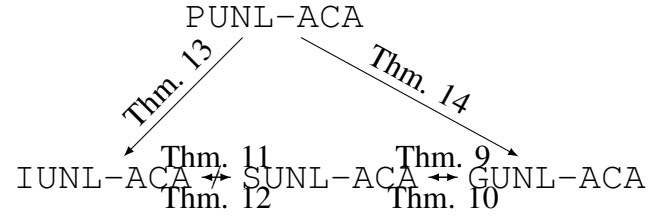


Fig. 4: Relationships among privacy security notions. An arrow represents an implication and a hatched arrow represents a separation. The number on an arrow or hatched arrow refers to the theorem in this paper which establishes this relationship.

E. SUNL−ACA ⇒ GUNL−ACA

Theorem 9 ((SUNL−ACA ⇒ GUNL−ACA))**.** *If a probabilistic polynomial time adversary $\mathcal{A}$ breaks the SUNL−ACA security of a* ReCS*, then it also breaks the GUNL−ACA security of a* ReCS *such that:*

$$\varepsilon_{\mathsf{sunl}} = \varepsilon_{\mathsf{gunl}}$$

Proof. To simplify the explanation, we view the graph unlinkability adversary as $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}$ with respect to the the training, challenge and guessing phase in $\mathbf{Game}^{\text{gunl-aca}}_{\mathsf{ReCS},\mathcal{B}}$.

TABLE V: The $\mathtt{SUNL}-\mathtt{ACA}$ security game.

$\mathbf{Game}^{\mathsf{sunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}}$

$IS \leftarrow \emptyset; SS \leftarrow \emptyset; CPV \leftarrow \emptyset;\ HPV \leftarrow \emptyset$
$\mathcal{O}(\cdot) \leftarrow \{\mathcal{O}_{\mathtt{Verify}}(\cdot), \mathcal{O}_{\mathtt{CorrP}}(\cdot)\}$
$b \leftarrow \{0,1\};\ (mpk, msk) \leftarrow \mathtt{Setup}(1^k);$
$(cred_0, cred_1, \phi^*) \leftarrow \mathcal{A}^{\mathcal{O}}(mpk, msk)$
If $|\mathcal{G}^{\mathtt{U}}_0| \neq |\mathcal{G}^{\mathtt{U}}_1| \vee |\mathcal{G}^{\mathtt{I}}_0| \neq |\mathcal{G}^{\mathtt{I}}_1|$, then return Lose
If $\phi^*(\mathcal{G}^{\mathtt{U}}_0 \cup \mathcal{G}^{\mathtt{I}}_0) = 0 \vee \phi^*(\mathcal{G}^{\mathtt{U}}_1 \cup \mathcal{G}^{\mathtt{I}}_1) = 0$, then return Lose
$(\tilde{\pi}_b, 1) \leftarrow (\mathtt{Prove}(mpk, cred_b, \phi^*) \leftrightarrow \mathcal{A}^{\mathcal{O}}(cred_0, cred_1, \phi^*)$
$(\tilde{\pi}_{1-b}, 1) \leftarrow (\mathtt{Prove}(mpk, cred_{1-b}, \phi^*) \leftrightarrow \mathcal{A}^{\mathcal{O}}(cred_0, cred_1, \phi^*)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}((cred_0, cred_1), \phi^*, (\tilde{\pi}_b, \tilde{\pi}_{1-b}))$
If $(\cdot,\cdot,\cdot,\cdot,\cdot,\tilde{\pi}_b) \in CPV \vee (\cdot,\cdot,\cdot,\cdot,\cdot,\tilde{\pi}_{1-b}) \in CPV$, then return Lose
If $b = b'$, then return Win, else return Lose

Note: $(cred_0, cred_1)$: challenge credentials, ϕ^* : challenge predicate,
$(\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_0, \mathcal{G}^{\mathtt{I}}_1)$: challenge graphs, (π_0, π_1) : issuing transcript,
IS: issuing session, SS: showing session,
CPV: corrupted user list, HPV: honest user list

TABLE VI: The $\mathtt{GUNL}-\mathtt{ACA}$ security game.

$\mathbf{Game}^{\mathsf{gunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}}$

$IS \leftarrow \emptyset; SS \leftarrow \emptyset; CPV \leftarrow \emptyset;\ HPV \leftarrow \emptyset$
$\mathcal{O}(\cdot) \leftarrow \{\mathcal{O}_{\mathtt{Verify}}(\cdot), \mathcal{O}_{\mathtt{CorrP}}(\cdot)\}$
$b \leftarrow \{0,1\};\ (mpk, msk) \leftarrow \mathtt{Setup}(1^k);$
$((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*) \leftarrow \mathcal{A}^{\mathcal{O}}(mpk, msk)$
If $|\mathcal{G}^{\mathtt{U}}_0| \neq |\mathcal{G}^{\mathtt{U}}_1| \vee |\mathcal{G}^{\mathtt{I}}_0| \neq |\mathcal{G}^{\mathtt{I}}_1|$, then return Lose
If $\phi(\mathcal{G}^{\mathtt{U}}_0 \cup \mathcal{G}^{\mathtt{I}}_0) = 0 \vee \phi(\mathcal{G}^{\mathtt{U}}_1 \cup \mathcal{G}^{\mathtt{I}}_1) = 0$, then return Lose
$(\pi_0, cred_0) \leftarrow (\mathtt{Obtain}(mpk, \mathcal{G}^{\mathtt{U}}_0) \leftrightarrow \mathtt{Issue}(mpk, msk, \mathcal{G}^{\mathtt{I}}_0)$
$(\pi_1, cred_1) \leftarrow (\mathtt{Obtain}(mpk, \mathcal{G}^{\mathtt{U}}_1) \leftrightarrow \mathtt{Issue}(mpk, msk, \mathcal{G}^{\mathtt{I}}_1)$
$(\tilde{\pi}_b, 1) \leftarrow (\mathtt{Prove}(mpk, cred_b, \phi^*) \leftrightarrow \mathcal{A}^{\mathcal{O}}((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*)$
$(\tilde{\pi}_{1-b}, 1) \leftarrow (\mathtt{Prove}(mpk, cred_{1-b}, \phi^*) \leftrightarrow \mathcal{A}^{\mathcal{O}}((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}}((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*, (\tilde{\pi}_b, \tilde{\pi}_{1-b}))$
If $(\cdot,\cdot,\cdot,\cdot,\cdot,\tilde{\pi}_b) \in CPV \vee (\cdot,\cdot,\cdot,\cdot,\cdot,\tilde{\pi}_{1-b}) \in CPV$, then return Lose
If $b = b'$, then return Win, else return Lose

Note: $(cred_0, cred_1)$: credentials, ϕ^* : challenge predicate,
$(\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_0, \mathcal{G}^{\mathtt{I}}_1)$: challenge graphs, (π_0, π_1) : issuing transcript,
IS: issuing session, SS: showing session,
CPV: corrupted user list, HPV: honest user list

Assume $\mathcal{B}$ exists, we can construct an adversary $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ to break the showing unlinkability of a ReCS with the help from $\mathcal{B}$. Specifically, $\mathcal{A}$ runs $\mathcal{B}$ as its sub-routine as follows:

Algorithm $\mathcal{A}^{\mathcal{O}}_1(mpk, msk)$
$((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*) \leftarrow \mathcal{B}^{\mathcal{O}}_1(mpk, msk)$
$(cred_0) \leftarrow \mathcal{O}_{\mathtt{Issue}}(mpk, msk, (\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), sid)$
$(cred_1) \leftarrow \mathcal{O}_{\mathtt{Issue}}(mpk, msk, (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), sid)$
return $(cred_0, cred_1, \phi^*)$

Algorithm $\mathcal{A}^{\mathcal{O}}_2(cred_0, cred_1, \phi^*)$
$(\tilde{\pi}_b, 1) \leftarrow \mathcal{B}^{\mathcal{O}}_2((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*)$
$(\tilde{\pi}_{1-b}, 1) \leftarrow \mathcal{B}^{\mathcal{O}}_2((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*)$
return $(\tilde{\pi}_b, \tilde{\pi}_{1-b})$

Algorithm $\mathcal{A}^{\mathcal{O}}_3((cred_0, cred_1), \phi^*, (\tilde{\pi}_b, \tilde{\pi}_{1-b}))$
$b' \leftarrow \mathcal{B}^{\mathcal{O}}_3((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*, (\tilde{\pi}_b, \tilde{\pi}_{1-b}))$
return b'

Since $\mathcal{A}$ simulates the environment perfectly, we have:

$$\begin{aligned}\varepsilon_{\mathsf{sunl}} &= \Pr[\mathbf{Game}^{\mathsf{sunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}] - \frac{1}{2} \\ &= \Pr[\mathbf{Game}^{\mathsf{gunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}] - \frac{1}{2} \\ &= \varepsilon_{\mathsf{gunl}} + \frac{1}{2} - \frac{1}{2} \\ &= \varepsilon_{\mathsf{gunl}}\end{aligned}$$

as required. □

F. GUNL-ACA ⇒ SUNL-ACA

Theorem 10 ($\mathtt{GUNL}-\mathtt{ACA} \Rightarrow \mathtt{SUNL}-\mathtt{ACA}$)**.** *If a probabilistic polynomial time adversary $\mathcal{A}$ breaks the GUNL-ACA security of a* ReCS*, then it also breaks the SUNL-ACA security of a* ReCS *such that:*

$$\varepsilon_{\mathsf{gunl}} = \varepsilon_{\mathsf{sunl}}$$

Proof. To ease the explanation, we view the showing unlinkability adversary as $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}$ with respect to the the training, challenge and guessing phase in $\mathbf{Game}^{\mathsf{sunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{B}}$. Assume $\mathcal{B}$ exists, we can construct an adversary $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ to break the graph unlinkability of a ReCS with the help from $\mathcal{B}$. Specifically, $\mathcal{A}$ runs $\mathcal{B}$ as its sub-routine as follows:

Algorithm $\mathcal{A}^{\mathcal{O}}_1(mpk, msk)$
$(cred_0, cred_1, \phi^*) \leftarrow \mathcal{B}^{\mathcal{O}}_1(mpk, msk)$
$(\sigma_0, \mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0) \leftarrow cred_0;\ (\sigma_1, \mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1) \leftarrow cred_1$
return $((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*)$

Algorithm $\mathcal{A}^{\mathcal{O}}_2((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*)$
$(\tilde{\pi}_b, 1) \leftarrow \mathcal{B}^{\mathcal{O}}_2(cred_0, cred_1, \phi^*)$
$(\tilde{\pi}_{1-b}, 1) \leftarrow \mathcal{B}^{\mathcal{O}}_2(cred_0, cred_1, \phi^*)$
return $(\tilde{\pi}_b, \tilde{\pi}_{1-b})$

Algorithm $\mathcal{A}^{\mathcal{O}}_3((\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{I}}_0), (\mathcal{G}^{\mathtt{U}}_1, \mathcal{G}^{\mathtt{I}}_1), \phi^*, (\tilde{\pi}_b, \tilde{\pi}_{1-b}))$
$b' \leftarrow \mathcal{B}^{\mathcal{O}}_3((cred_0, cred_1), \phi^*, (\tilde{\pi}_b, \tilde{\pi}_{1-b}))$
return b'

During the challenge phase, if the credential generation algorithm is not deterministic, with high probability, $\mathcal{A}_2$'s prover uses valid $(cred'_0 \neq cred_0, cred'_1 \neq cred_1)$ to interact with the verifier $\mathcal{A}_2$. We highlight that by the definition of SUNL−ACA, $\mathcal{B}_2$ cannot distinguish whether $(cred_0, cred_1)$ or $(cred'_0, cred'_1)$ are used in the showing protocol as long as $\phi(\mathcal{G}_0^{\mathtt{U}} \cup \mathcal{G}_0^{\mathtt{I}}) = \phi(\mathcal{G}_1^{\mathtt{U}} \cup \mathcal{G}_1^{\mathtt{I}}) = 1$ holds. If $\mathcal{B}$ can distinguish so, it can set $\mathcal{G}_0^{\mathtt{U}} = \mathcal{G}_1^{\mathtt{U}}$ and $\mathcal{G}_0^{\mathtt{I}} = \mathcal{G}_1^{\mathtt{I}}$ to run the challenge phase with $(cred_0, cred_1)$ in which it can always win the $\mathtt{SUNL-ACA}$ game.

Since $\mathcal{A}$ simulates the environment perfectly, we have:

$$\begin{aligned}\varepsilon_{\mathsf{gunl}} &= \Pr[\mathbf{Game}^{\mathsf{gunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}] - \frac{1}{2} \\ &= \Pr[\mathbf{Game}^{\mathsf{sunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}] - \frac{1}{2} \\ &= \varepsilon_{\mathsf{sunl}} + \frac{1}{2} - \frac{1}{2} \\ &= \varepsilon_{\mathsf{sunl}}\end{aligned}$$

as required. □

G. IUNL−ACA ⇏ SUNL−ACA

Theorem 11 (IUNL−ACA ⇏ SUNL−ACA)**.** *If there exists a* ReCS *which is IUNL−ACA secure, then there exists an* ReCS′ *which is IUNL−ACA secure but not SUNL2−ACA secure.*

Proof. Assume there exists an IUNL−ACA secure $\mathtt{ReCS} = (\mathtt{Setup}, (\mathtt{Obtain}, \mathtt{Issue}), (\mathtt{Prove}, \mathtt{Verify}))$, since otherwise, the theorem is vacuously true. We now modify the ReCS into a new $\mathtt{ReCS}' = (\mathtt{Setup}', (\mathtt{Obtain}', \mathtt{Issue}'), (\mathtt{Prove}', \mathtt{Verify}'))$ which is also IUNL−ACA secure but not SUNL−ACA secure. The new ReCS′ is defined as follows.

Algorithm $\mathtt{Setup}'(1^k)$
  $(mpk, msk) \leftarrow \mathtt{Setup}(1^k)$
  return (mpk, msk)

Algorithm $\mathtt{Obtain}'(mpk, \mathcal{G}^{\mathtt{U}})$
  $(\pi, cred) \leftarrow \mathtt{Obtain}(mpk, \mathcal{G}^{\mathtt{U}})$
  return $(\pi, cred)$

Algorithm $\mathtt{Issue}'(mpk, msk, \mathcal{G}^{\mathtt{I}})$
  $(\pi) \leftarrow \mathtt{Issue}(mpk, msk, \mathcal{G}^{\mathtt{I}})$
  return (π)

Algorithm $\mathtt{Prove}'(mpk, cred, \phi^*)$
  $(\tilde{\pi}) \leftarrow \mathtt{Prove}(mpk, cred, \phi^*)$
  $\tilde{m}_1 \leftarrow \tilde{\pi} \setminus \{\tilde{m}_2, \ldots\}$
  $\tilde{\pi} \leftarrow \tilde{\pi} \setminus \tilde{m}_1$
  $\tilde{m}_1 \leftarrow \tilde{m}_1 \cup cred$
  $\tilde{\pi} \leftarrow \tilde{\pi} \cup \tilde{m}_1$
  return $(\tilde{\pi})$

Algorithm $\mathtt{Verify}'(mpk, \phi^*)$
  $\tilde{m}_1 \leftarrow \tilde{\pi} \setminus \{\tilde{m}_2, \ldots\}$
  $\tilde{\pi} \leftarrow \tilde{\pi} \setminus \tilde{m}_1$
  $\tilde{m}_1 \leftarrow \tilde{m}_1 \setminus cred$
  $\tilde{\pi} \leftarrow \tilde{\pi} \cup \tilde{m}_1$
  $(\tilde{\pi}) \leftarrow \mathtt{Verify}(mpk, \phi^*)$
  return $(\tilde{\pi}, d)$

Essentially, the first outgoing message $\tilde{m}_1$ of $\mathtt{Prove}'$ is appended with $cred$ while $\mathtt{Verify}'$ ignores the appended $cred$. The incoming messages for Verify are always valid and the showing protocol executes successfully.

a) $\mathtt{ReCS}'$ *is not* SUNL−ACA *secure.:* As the outgoing message $\tilde{m}_1 \cup cred \rightarrow \tilde{m}_1 \in \tilde{\pi}$ of the prover in the showing protocol is the input message $\tilde{m}_1$ for the verifier, $\mathcal{A}$ from SUNL−ACA can view the showing transcript $\tilde{\pi}$ and extract $cred$ to win in the game.

b) $\mathtt{ReCS}'$ *is* IUNL−ACA *secure.:* By the definition of IUNL−ACA, the credential $cred$ is not accessible to its $\mathcal{A}$ through out the security game. Also, $\mathcal{A}$ does not have a way to force the prover in using the $cred$ (which might not exist) of its interest in a showing protocol. So, if ReCS is IUNL−ACA secure, then $\mathtt{ReCS}'$ is also IUNL−ACA secure. □

H. SUNL−ACA ⇏ IUNL−ACA

Theorem 12 (SUNL−ACA ⇏ IUNL−ACA)**.** *If there exists a* ReCS *which is SUNL−ACA secure, then there exists an* ReCS′ *which is SUNL−ACA secure but not IUNL−ACA secure.*

Proof. Assume there exists an SUNL−ACA secure $ReCS = (\mathtt{Setup}, (\mathtt{Obtain}, \mathtt{Issue}), (\mathtt{Prove}, \mathtt{Verify}))$, since otherwise, the theorem is vacuously true. We now modify the ReCS into a $\mathtt{ReCS}' = (\mathtt{Setup}', (\mathtt{Obtain}', \mathtt{Issue}'), (\mathtt{Prove}', \mathtt{Verify}'))$ which is also SUNL−ACA secure but not IUNL−ACA secure. The new $\mathtt{ReCS}'$ is defined as follows.

Algorithm $\mathtt{Setup}'(1^k)$
  $(mpk, msk) \leftarrow \mathtt{Setup}(1^k)$
  return (mpk, msk)

Algorithm $\mathtt{Obtain}'(mpk, \mathcal{G}^{\mathtt{U}})$
  $(\pi, cred) \leftarrow \mathtt{Obtain}(mpk, \mathcal{G}^{\mathtt{U}})$
  $m_1 \leftarrow \pi \setminus \{m_2, \ldots\}$
  $\pi \leftarrow \pi \setminus m_1$
  $m_1 \leftarrow m_1 \cup \mathcal{G}^{\mathtt{U}}$
  $\pi \leftarrow \pi \cup m_1$
  return $(\pi, cred)$

Algorithm $\mathtt{Issue}'(mpk, msk, \mathcal{G}^{\mathtt{I}})$
  $m_1 \leftarrow \pi \setminus \{m_2, \ldots\}$
  $\pi \leftarrow \pi \setminus m_1$
  $m_1 \leftarrow m_1 \setminus \mathcal{G}^{\mathtt{U}}$
  $\pi \leftarrow \pi \cup m_1$
  $(\pi) \leftarrow \mathtt{Issue}(mpk, msk, \mathcal{G}^{\mathtt{I}})$
  return (π)

Algorithm $\mathtt{Prove}'(mpk, cred, \phi^*)$
  $(\tilde{\pi}, d) \leftarrow \mathtt{Prove}(mpk, cred, \phi^*)$
  return $(\tilde{\pi})$

Algorithm $\mathtt{Verify}'(mpk, \phi^*)$
  $(\tilde{\pi}, d) \leftarrow \mathtt{Verify}(mpk, \phi^*)$
  return $(\tilde{\pi})$

Basically, the first outgoing message m_1 of $\mathtt{Obtain}'$ is appended with $\mathcal{G}^{\mathtt{U}}$ while $\mathtt{Issue}'$ ignores the appended $\mathcal{G}^{\mathtt{U}}$. The incoming messages for Issue are always valid and the issuing protocol executes successfully.

a) $\mathtt{ReCS}'$ *is not* IUNL−ACA *secure.:* As the outgoing message $m_1 \cup \mathcal{G}^{\mathtt{U}} \rightarrow m_1 \in \pi$ of the user in the issuing protocol is the input message m_1 for the issuer, $\mathcal{A}$ from IUNL−ACA can view the issuing transcript π and extract $\mathcal{G}^{\mathtt{U}}$ to win in the game.

b) $\mathtt{ReCS}'$ *is* SUNL−ACA *secure.:* By the definition of SUNL−ACA, the graph $\mathcal{G}^{\mathtt{U}}$ in the issuing protocol is not privy to its $\mathcal{A}$ which can choose $(\mathcal{G}^{\mathtt{U}}, \mathcal{G}^{\mathtt{I}})$ of its favour to generate $cred$. So, if the ReCS is SUNL−ACA secure, then $\mathtt{ReCS}'$ is also SUNL−ACA secure. □

I. PUNL−ACA ⇒ IUNL−ACA

Theorem 13 (PUNL−ACA ⇒ IUNL−ACA)**.** *If a probabilistic polynomial time adversary $\mathcal{A}$ breaks the* PUNL−ACA *security of a* ReCS*, then it also breaks the* IUNL−ACA *security of a* ReCS *such that:*

$$\varepsilon_{\mathsf{punl}} = \frac{1}{2}\varepsilon_{\mathsf{iunl}}$$

Proof. To explain in a concise manner, we view the issuing unlinkability adversary as $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}$ with respect to the the training, challenge and guessing phase in $\mathbf{Game}^{\mathsf{iunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{B}}$. Assume $\mathcal{B}$ exists, we can construct an adversary $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ to break the protocol unlinkability of a ReCS with the help from $\mathcal{B}$. Specifically, $\mathcal{A}$ runs $\mathcal{B}$ as its sub-routine as follows:

Algorithm $\mathcal{A}_1^{\mathcal{O}}(mpk, msk)$
  $(\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}) \leftarrow \mathcal{B}_1^{\mathcal{O}}(mpk, msk)$
  $\phi \leftarrow \Phi$ s.t. $\phi(\mathcal{G}_0^{\mathtt{U}}) = \phi(\mathcal{G}_1^{\mathtt{U}}) = 1$
  return $(\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}, \phi^*)$

Algorithm $\mathcal{A}_2^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\cdot,\cdot), \phi^*)$
  $(\pi_b, cred_b) \leftarrow \mathcal{B}_2^{\mathcal{O}}(\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}, \mathcal{G}_b^{\mathtt{I}}, \mathcal{G}_{1-b}^{\mathtt{I}})$
  $(\pi_{1-b}, cred_{1-b}) \leftarrow \mathcal{B}_2^{\mathcal{O}}(\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}, \mathcal{G}_b^{\mathtt{I}}, \mathcal{G}_{1-b}^{\mathtt{I}})$
  $\mathcal{G}_{b_1}^{\mathtt{I}} \leftarrow \mathcal{G}_b^{\mathtt{I}};\ \mathcal{G}_{1-b_1}^{\mathtt{I}} \leftarrow \mathcal{G}_{1-b}^{\mathtt{I}}$
  $\pi_{b_1} \leftarrow \pi_b; \pi_{1-b_1} \leftarrow \pi_{1-b}$
  return $((\mathcal{G}_{b_1}^{\mathtt{I}}, \mathcal{G}_{1-b_1}^{\mathtt{I}}), (\pi_{b_1}, \pi_{1-b_1}))$

Algorithm $\mathcal{A}_3^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\mathcal{G}_{b_1}^{\mathtt{I}}, \mathcal{G}_{1-b_1}^{\mathtt{I}}), (\pi_{b_1}, \pi_{1-b_1}), \phi^*, (\tilde{\pi}_{b_2}, \tilde{\pi}_{1-b_2}))$
  $b' \leftarrow \mathcal{B}_3^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\mathcal{G}_b^{\mathtt{I}}, \mathcal{G}_{1-b}^{\mathtt{I}}), (\pi_b, \pi_{1-b}))$
  return b'

where $\mathcal{A}$ simulates the environment perfectly for $\mathcal{B}$. During the challenge phase, as $\mathcal{A}_2$ acts as the man-in-the-middle between its user and $\mathcal{B}_2$, the graphs $(\mathcal{G}_{b_1}^{\mathtt{I}}, \mathcal{G}_{1-b_1}^{\mathtt{I}}) = (\mathcal{G}_b^{\mathtt{I}}, \mathcal{G}_{1-b}^{\mathtt{I}})$ are determined by $\mathcal{B}_2$ instead. Besides, as $\mathcal{B}_2$ does not run the showing protocols, $\mathcal{A}_2$ runs the showing protocols by itself to get $(\tilde{\pi}_{b_2}, \tilde{\pi}_{1-b_2})$. Finally, $\mathcal{A}$ sets the output b' from $\mathcal{B}$ as its guess. The probability of $\mathcal{A}$ winning the game is:

$$\begin{aligned}
&\Pr[\mathbf{Game}^{\mathsf{punl\text{-}aca}}_{\mathsf{ReCS},\mathcal{B}} = \mathsf{Win}] = \Pr[b_1 \oplus b_2 = b'] \\
&= \frac{1}{2}\left(\Pr[b' = 0 | b_1 \oplus b_2 = 0] + \Pr[b' = 1 | b_1 \oplus b_2 = 1]\right) \\
&= \frac{1}{2}\left(\Pr[b' = b_1 = b_2] + \Pr[b' = b_1 \neq b_2]\right) \\
&= \frac{1}{2}\left(\varepsilon_{\mathsf{iunl}} + \frac{1}{2}\right) + \frac{1}{2}\frac{1}{2} \\
&= \frac{1}{2}\varepsilon_{\mathsf{iunl}} + \frac{1}{2}.
\end{aligned}$$

Therefore, by definition, the advantage of $\mathcal{A}$ is:

$$\begin{aligned}
\varepsilon_{\mathsf{punl}} &= \Pr[\mathbf{Game}^{\mathsf{punl\text{-}aca}}_{\mathsf{ReCS},\mathcal{B}} = \mathsf{Win}] - \frac{1}{2} \\
&= \frac{1}{2}\varepsilon_{\mathsf{iunl}} + \frac{1}{2} - \frac{1}{2} = \frac{1}{2}\varepsilon_{\mathsf{iunl}}
\end{aligned}$$

as required. □

J. PUNL−ACA ⇒ GUNL−ACA

Theorem 14 (PUNL−ACA ⇒ GUNL−ACA)**.** *If a probabilistic polynomial time adversary $\mathcal{A}$ breaks the* PUNL−ACA *security of a* ReCS*, then it also breaks the* IUNL−ACA *security of a* ReCS *such that:*

$$\varepsilon_{\mathsf{punl}} = \frac{1}{2}\varepsilon_{\mathsf{gunl}}$$

Proof. To explain neatly, we view the graph unlinkability adversary as $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3\}$ with respect to the the training, challenge and guessing phase in $\mathbf{Game}^{\mathsf{gunl\text{-}aca}}_{\mathsf{ReCS},\mathcal{B}}$. Assume $\mathcal{B}$ exists, we can construct an adversary $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ to break the protocol unlinkability of a ReCS with the help from $\mathcal{B}$. Specifically, $\mathcal{A}$ runs $\mathcal{B}$ as its sub-routine as follows:

Algorithm $\mathcal{A}_1^{\mathcal{O}}(mpk, msk)$
  $((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_0^{\mathtt{I}}), (\mathcal{G}_1^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{I}}), \phi^*) \leftarrow \mathcal{B}_1^{\mathcal{O}}(mpk, msk)$
  return $(\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}, \phi^*)$

Algorithm $\mathcal{A}_2^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\mathcal{G}_0^{\mathtt{I}}, \mathcal{G}_1^{\mathtt{I}}), \phi^*)$
  $b_1' \leftarrow \{0,1\}$
  $(\pi_{b_1}, cred_{b_1}) \leftarrow \mathtt{Issue}(mpk, msk, \mathcal{G}_{b_1'}^{\mathtt{I}})$
  $(\pi_{1-b_1}, cred_{1-b_1}) \leftarrow \mathtt{Issue}(mpk, msk, , \mathcal{G}_{1-b_1'}^{\mathtt{I}})$
  $(\tilde{\pi}_b, 1) \leftarrow \mathcal{B}_2^{\mathcal{O}}(\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}, \mathcal{G}_0^{\mathtt{I}}, \mathcal{G}_1^{\mathtt{I}}, \phi^*)$
  $(\tilde{\pi}_{1-b}, 1) \leftarrow \mathcal{B}_2^{\mathcal{O}}(\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}, \mathcal{G}_0^{\mathtt{I}}, \mathcal{G}_1^{\mathtt{I}}, \phi^*)$
  $\tilde{\pi}_{b_2} \leftarrow \tilde{\pi}_b; \tilde{\pi}_{1-b_2} \leftarrow \tilde{\pi}_{1-b}$
  return $((\pi_{b_1}, \pi_{1-b_1}), (\tilde{\pi}_{b_1}, \tilde{\pi}_{1-b_1}))$

Algorithm $\mathcal{A}_3^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{U}}), (\mathcal{G}_{b_1}^{\mathtt{I}}, \mathcal{G}_{1-b_1}^{\mathtt{I}}), (\pi_{b_1}, \pi_{1-b_1}), \phi^*, (\tilde{\pi}_{b_2}, \tilde{\pi}_{1-b_2}))$
  $b' \leftarrow \mathcal{B}_3^{\mathcal{O}}((\mathcal{G}_0^{\mathtt{U}}, \mathcal{G}_0^{\mathtt{I}}), (\mathcal{G}_1^{\mathtt{U}}, \mathcal{G}_1^{\mathtt{I}}), \phi, (\pi_b, \pi_{1-b}))$
  return b'

where $\mathcal{A}$ simulates the environment perfectly for $\mathcal{B}$. During the challenge phase, as $\mathcal{B}_2$ does not run the issuing protocols, the signer $\mathcal{A}_2$ interacts with its user to generate $(\pi_{b_1}, cred_{b_1})$ and $(\pi_{1-b_1}, cred_{1-b_1})$. During the issuing, $\mathcal{A}_2$ guesses the sequence $b_1 \in \{0,1\}$ selected by its prover and chooses a graph $\mathcal{G}_0^{\mathtt{I}}$ or $\mathcal{G}_1^{\mathtt{I}}$, that is, $\mathcal{G}_{b_1'}^{\mathtt{I}}$ and $\mathcal{G}_{1-b_1'}^{\mathtt{I}}$ for a bit $b_1' \in \{0,1\}$. Even if $\mathcal{A}_2$ made a wrong guess such that $b_1 \neq b_1'$, $\mathcal{B}$ cannot discover this error because when $\phi(\mathcal{G}_0^{\mathtt{U}} \cup \mathcal{G}_0^{\mathtt{I}}) = \phi(\mathcal{G}_1^{\mathtt{U}} \cup \mathcal{G}_1^{\mathtt{I}}) = 1$, it must be $\phi(\mathcal{G}_0^{\mathtt{U}}) = \phi(\mathcal{G}_1^{\mathtt{U}})$ and $\phi(\mathcal{G}_0^{\mathtt{I}}) = \phi(\mathcal{G}_1^{\mathtt{I}})$ and therefore $\phi(\mathcal{G}_0^{\mathtt{U}} \cup \mathcal{G}_1^{\mathtt{I}}) = \phi(\mathcal{G}_1^{\mathtt{U}} \cup \mathcal{G}_0^{\mathtt{I}}) = 1$. Moreover, as explained in Section F, $\mathcal{B}$ is not able to distinguish valid $(cred_{b_2}, cred_{1-b_2})$ under ϕ^* during a showing protocol.

Subsequently, $\mathcal{A}_2$ acts as the man in the middle between its prover and $\mathcal{B}_2$ to get $(\tilde{\pi}_{b_2}, \tilde{\pi}_{1-b_2})$. Finally, $\mathcal{A}$ sets the output b' from $\mathcal{B}$ as its guess. The probability of $\mathcal{A}$ winning the game is:

$$\begin{aligned}
&\Pr[\mathbf{Game}^{\mathsf{punl\text{-}aca}}_{\mathsf{ReCS},\mathcal{B}} = \mathsf{Win}] = \Pr[b_1 \oplus b_2 = b'] \\
&= \frac{1}{2}\left(\Pr[b' = 0 | b_1 \oplus b_2 = 0] + \Pr[b' = 1 | b_1 \oplus b_2 = 1]\right) \\
&= \frac{1}{2}\left(\Pr[b' = b_2 = b_1] + \Pr[b' = b_2 \neq b_1]\right) \\
&= \frac{1}{2}(\varepsilon_{\mathsf{gunl}} + \frac{1}{2}) + \frac{1}{2}\frac{1}{2} \\
&= \frac{1}{2}\varepsilon_{\mathsf{gunl}} + \frac{1}{2}.
\end{aligned}$$

Hence, by definition, the advantage of $\mathcal{A}$ is:

$$\begin{aligned}
\varepsilon_{\mathsf{punl}} &= \Pr[\mathbf{Game}^{\mathsf{punl\text{-}aca}}_{\mathsf{ReCS},\mathcal{B}} = \mathsf{Win}] - \frac{1}{2} \\
&= \frac{1}{2}\varepsilon_{\mathsf{gunl}} + \frac{1}{2} - \frac{1}{2} = \frac{1}{2}\varepsilon_{\mathsf{gunl}}
\end{aligned}$$

as required. □

`OpenDifference` $(pk, C, A, O, (A', (\bar{\ell}, \bar{\kappa})))$. Let $A' = \{A'_1, \ldots, A'_L\}, \bar{\kappa} = \{\bar{k}_1, \ldots, \bar{k}_L\}$, $1 \leq \bar{\ell} \leq L$ and $D_{i,j} = A'_j \setminus A_i$. If there are $\bar{\ell}$-many A'_j such that $|D_{i,j}| = \bar{k}_j \leq |A'_j|$ holds for all $A_i \in A$, group these differing sets $D_{i,j}$ with respect to A'_j as D_j and subsequently as the multi-set D where each D_j contains $D_{i,j}$ for every $A_i \in A$. Return D and the corresponding witness

$$W = \Big(\Big\{ \forall A_i \in A : W_{i,j} = a_{i_0}^{q_{i,j}(x')}, r_{i,j}(x'), \Big\{ \forall m \in D_{i,j} : W_{i,j,m} = a_{i_0}^{q_{i,j,m}(x')}, r_{i,j,m}(x') \Big\} \Big\}_{A'_j \in D} \Big).$$

Specifically, let the polynomial divisor be $d_{i,j}(x') = \prod_{m \in D_{i,j}} (x' + m)$, the monic polynomial $f_i(x') = o_i \prod_{m \in A_i} (x' + m)$ in the commitment $C_i = a_{i_0}^{f_i(x')}$ can be rewritten[2] as $f_i(x') = d_{i,j}(x') q_{i,j}(x') + r_{i,j}(x')$ while the $(\bar{k}_i - 1)$-degree $r_{i,j}(x') = (x' + m) q_{i,j,m}(x') + r_{i,j,m}(x')$ for every $m \in D_{i,j}$ where $r_{i,j,m}(x')$ is a non-zero constant. If no such multi-set D exists, return an error $\perp$.

`VerifyDifference` $(pk, C, (D, W), (A', (\bar{\ell}, \bar{\kappa})))$. Let $D = \{D_j\}_{A'_j \overset{\bar{\ell}}{\in} A'}$ where each D_j contains $D_{i,j}$ for every $A_i \in A$, return 1 if the following conditions hold:

1) $\forall D_{i,j} \in A' : \mathsf{e}\left(\prod_{A_i \in A} \prod_{A'_j \in A'} a_{0_0}^{\prod_{m \in A'_j} (x'+m)}, X_{0_0} \right) = \mathsf{e}\left(\prod_{A_i \in A} \prod_{A'_j \in A' \setminus D} a_{0_0}^{\prod_{m \in A'_j} (x'+m)}, X_{0_0} \right) \cdot \prod_{A_i \in A} \prod_{A'_j \in D} \mathsf{e}\left(a_{0_0}^{\prod_{m \in A'_j \setminus D_{i,j}} (x'+m)}, X_{0_0}^{\prod_{m \in D_{i,j}} (x'+m)} \right)$
2) $\forall D_{i,j} \in D$ is disjoint to A :
   a) $\forall D_{i,j} \in D : R_{i,j} = a_{i_0}^{r_{i,j}(x')} \neq 1_{\mathbb{G}_1}$
   b) $\forall D_j \in D : \mathsf{e}(C, X_{0_0}) = \prod_{A_i \in A} \mathsf{e}\left(W_{i,j}, X_{0_0}^{\prod_{m \in D_{i,j}} (x'+m)} \right) \cdot \mathsf{e}\left(\prod_{A_i \in A} R_{i,j}, X_{0_0} \right)$
3) $\forall m \in D_{i,j}$ is co-prime to the respective remainders $R_{i,j}$ above.
   a) $\forall m \in D_{i,j} : R_{i,j,m} = a_{i_0}^{r_{i,j,m}(x')} \neq 1_{\mathbb{G}_1}$
   b) $\forall m \in D_{i,j} : \mathsf{e}\left(\prod_{A_i \in A} R_{i,j}, X_{0_0} \right) = \mathsf{e}\left(\prod_{A_i \in A} W_{i,j,m}, X_{0_0}^{(x'+m)} \right) \cdot \mathsf{e}\left(\prod_{A_i \in A} R_{i,j,m}, X_{0_0} \right)$

and return 0 otherwise.

Theorem 3 (Binding)**.** *The MoniPoly multi-set commitment scheme is computationally binding if the co-DLOG problem is hard.*

Proof. Given a co-DLOG instance $(g_1, h_1 = g_1^x \in \mathbb{G}_1, g_2, h_2 = g_2^x \in \mathbb{G}_2)$, we construct a challenger $\mathcal{C}$ that runs the adversary $\mathcal{A}$ of MoniPoly multi-set commitment scheme as a sub-routine to find the solution x.

[2]Note that $r_{i,j}(x') \neq 0$ and therefore $a_{i_0}^{r_{i,j}(x')} \neq 1_{\mathbb{G}_1}$ whenever $d_{i,j}(x')$ cannot divide $f_i(x')$, i.e., the sets A_i and D_j are disjoint.

$\mathcal{C}$ sets $\{a_{0_k} = g_1^{x'^k}, X_{0_k} = g_2^{x'^k}\}_{k=0}^n$ and $\{a_{1_k} = h_1^{x'^k}, X_{1,k} = h_2^{x'^k}\}_{k=0}^n$ for randomly chosen $x' \in \mathbb{Z}_p^*$. Subsequently, $\mathcal{C}$ chooses random $\{b_i\}_{i=2}^L \in \mathbb{Z}_p^*$ and tosses a fair coin $c \in \{0, 1\}$ for $L-1$ times. If $c = 0$, $\{a_{i_k} = a_{0_k}^{b_i x'^k}, X_{i_k} = X_{0_k}^{b_i x'^k}\}$. Else if $c = 1$, $\{a_{i_k} = a_{1_k}^{b_i x'^k}, X_{i_k} = X_{1_k}^{b_i x'^k}\}$. $\mathcal{C}$ publishes $\{\{a_{i_k}, X_{i_k}\}_{k=0}^n\}_{i=0}^L$ as the public parameters.

If an adversary can output a commitment C for two different multi-sets $(A, O) \neq (A^*, O^*)$ such that:

$$\prod_{i=1}^{L} a_{i_0}^{o_i \prod_{m \in A_i} (x'+m)} = C = \prod_{i=1}^{L} a_{i_0}^{o_i^* \prod_{m^* \in A_i^*} (x'+m^*)},$$

the co-DLOG solution can be extracted. We first analyse the case of $A \neq A^*$ and $O \neq O^*$ but we do not consider the difference in a single set such that $A_i \neq A_i^*$ and $o_i \neq o_i^*$ as this will not happen by Theorem 16. Assume it is only $A_i \neq A_i^*$ and $o_j \neq o_j^*$ for two indices $i \neq j$. Assume c is distributed evenly, with probability of $\frac{L}{2(L-1)}$, the i-th and j-th sets fall under the bases g_1 and h_1, respectively:

$$\begin{aligned} & g_1^{\sum_{\forall is.t.c=0} b_i o_i \prod_{m \in A_k} (x'+m)} h_1^{\sum_{\forall js.t.c=1} b_j o_j \prod_{m \in A_k} (x'+m)} \\ &= g_1^u h_1^v \\ &= g_1^{\sum_{\forall is.t.c=0} b_i o_i^* \prod_{m \in A_k^*} (x'+m^*)} h_1^{\sum_{\forall js.t.c=1} b_j o_j^* \prod_{m^* \in A_k^*} (x'+m^*)} \\ &= g_1^{u^*} h_1^{v^*}. \end{aligned}$$

Therefore, $\mathcal{C}$ can compute $x = \frac{u - u^*}{v^* - v} \mod p$ to solve the co-DLOG problem. Now we consider the case of $A \neq A^*$ only, that is, $|A \cap A^*| \geq 2$ while $O = O^*$. It is clear that the equation above is still applicable and the same goes to the case of $O \neq O^*$ only, that is, $A = A^*$ while $|O \cap O^*| \geq 2$. □

Theorem 4 (Intersecting Binding)**.** *The MoniPoly multi-set commitment scheme is intersecting binding if the co-DLOG problem is hard.*

Proof. The intersecting witness W is itself a MoniPoly multi-set commitment on the intersecting set $I = \{A'_1 \cap A_1, \ldots, A'_\ell \cap A_\ell\}$. Therefore, if the commitment C is binding, so does W. Without loss of generality, let all the threshold values $k_i \in \kappa$ be $k_i = |A'_i|$ in OpenIntersection. When we have $A' \cap A = I = A' \cap A^*$ for a query set A', it is a special case of the pair $(A, O) \neq (A^*, O^*)$ that can fulfil the OpenIntersection algorithm. When $O \neq O^*$ (resp. $O = O^*$), it must be $|\overline{A \cap A^*}| \geq 1$ (resp. $|\overline{A \cap A^*}| \geq 2$) where we have $\ell \leq \min(|A|, |A^*|)$ (resp. $\ell \leq \min(|A|, |A^*|) - 1$) with $|A| \neq |A^*|$; or $\ell \leq |A| - 1$ (resp. $\ell \leq |A| - 2$) with $|A| = |A^*|$. □

Theorem 5 (Differing Binding)**.** *The MoniPoly multi-set commitment scheme is differing binding if the co-DLOG problem is hard.*

Proof. The intersecting witness W is itself a MoniPoly multi-set commitments on quotients $q_{i,j}(x')$ and $q_{i,j,m}(x')$. Also, by fundamental theorem of algebra, the non-zero remainders $r_{i,j}(x')$ and $r_{i,j,m}(x')$ are unique with respect to the unknown secret x'. Therefore, if the multi-set commitment C is binding, so does W. To elaborate further, let all the threshold values $\bar{k}_i \in \bar{\kappa}$ be $\bar{k}_i = |A'_i|$ in OpenDifference. When we have

$|A' - \bar{A}| = \bar{\ell} = |A' - \bar{A^*}|$ for a query A', it is a special case of the pair $(A, O) \neq (A^*, O^*)$ that can fulfil the OpenDifference algorithm. When $O \neq O^*$ (resp. $O = O^*$), it must be $|\overline{A \cap A^*}| \geq 1$ (resp. $|\overline{A \cap A^*}| \geq 2$), we have $\bar{\ell} \leq \min(|A|, |A^*|)$ when $|A| \neq |A^*|$; or $\bar{\ell} \leq |A|$ when $|A| = |A^*|$. □

K. Pedersen Vector Commitment Scheme

It is known that Pedersen commitment scheme can be extended to a vector commitment scheme by expanding pk to include more random generators such that $pk = (a_1, \ldots, a_n, b \in \mathbb{G})$. The vector commitment of n messages $\{m_1, \ldots, m_n\}$ is computed as:

$$C := a_1^{m_1} \cdots a_n^{m_n} b^r$$

where $r \in_R \mathbb{Z}_p^*$. For the completeness of the security analysis, we give the security proof of the perfectly hiding and binding properties for Pedersen vector commitment in the context of our ReCS issuing and showing protocols.

Lemma 2. *The Pedersen vector commitment in the issuing and showing protocols are perfectly hiding.*

Proof. In clause (3) of showing protocol, the equation $C = b^\rho c^\omega v'^\tau$ can be viewed as a Pedersen vector commitment such that for each vector (ρ, ω), there exists an unique opening value τ such that:

$$\begin{aligned} \mathsf{dlog}_b(C) &= \rho + \mathsf{dlog}_b(c)\omega + \mathsf{dlog}_b(v')\tau \pmod p \\ \tau &= \frac{\mathsf{dlog}_b(C) - \rho - \mathsf{dlog}_b(c)\omega}{\mathsf{dlog}_b(v')} \pmod p \end{aligned}$$

As the discrete logarithms with respect to the base b are not known and $\tau = -ty \in \mathbb{Z}_p^*$ is uniformly distributed, the vector $(\rho = sr, \omega = r)$ is perfectly hidden.

The proof for the clause (1) in issuing protocol follows the same approach. □

Lemma 3. *The Pedersen vector commitment in the issuing and showing protocol is binding if the DLOG problem is hard.*

Proof. The proof is similar to that in Theorem 3. Let the equation $C = b^\rho c^\omega v'^\tau$ in clause (3) of showing protocol be a Pedersen vector commitment. Given a DLOG instance $(g_1, g_2 = g_1^x)$, we construct a challenger $\mathcal{C}$ that finds the solution x with the help of an adversary $\mathcal{A}$ of the Pedersen vector commitment scheme. $\mathcal{C}$ tosses a fair coin to decide either g_1 or g_2 is used as the base for (b, c, v'), respectively. Assuming $b = g_1^{r_1}$ while $c = g_2^{r_2}, v' = g_2^{r_3}$ for randomly selected $r_1, r_2, r_3 \in \mathbb{Z}_p^*$, $\mathcal{C}$ publishes (b, c, v') as the public parameters for the Pedersen vector commitment scheme.

If an adversary can output a Pedersen vector commitment C for two different sets $A = (\alpha, (\beta, \gamma)), A^* = (\alpha^*, (\beta^*, \gamma^*))$ such that $|A \cap A^*| \geq 2$ and:

$$\begin{aligned} C = b^\alpha c^\beta v'^\gamma &= b^{\alpha^*} c^{\beta^*} v'^{\gamma^*} \\ \Leftrightarrow g_1^{r_1\alpha} g_2^{r_2\beta + r_3\gamma} &= g_1^{r_1\alpha^*} g_2^{r_2\beta^* + r_3\gamma^*} \end{aligned}$$

we have a probability of at least $1/2$ that the distinct set elements fall under g_1 and g_2, respectively. Without loss of generality, assuming $\alpha \neq \alpha^*$ and $\beta \neq \beta^*$, $\mathcal{C}$ can compute:

$$x = \frac{r_1(\alpha^* - \alpha)}{r_3(\beta - \beta^*)} \pmod p$$

to solve the DLOG problem.

The proof for the clause (1) in issuing protocol follows the same approach. □

L. Random Self-Reducibility

Lemma 4 (Issuing RSR)**.** *The initialization of the issuing protocols in the graph signature scheme has random self-reducibility.*

This follows directly from the random self-reducibility (RSR) of the q-SDH problem [43] and Schnorr-like identification scheme [44].

Lemma 5. *[Showing RSR] The showing protocols in the graph signature scheme have random self-reducibility.*

Proof. This follows directly from Lemma 4. Furthermore, the public input $v' = v^{ry^{-1}}$ from the randomized signature $\sigma' = (r, y, t \cdot r, s \cdot r, v^{ry^{-1}})$ also possesses self-reducibility from the Decisional DH assumption. Specifically, we can view v as:

$$v^x = \prod_{i \in V} C_i \prod_{(i,j) \in E} C_{(i,j)} b^s c v^{-t}$$

for the signature elements at the right hand side. In `CL.Randomize`, the prover first constructs a DH tuple (v, v^x, v^r, v^{rx}) for a randomly chosen $r \in \mathbb{Z}_p^*$. However, this DH tuple is linkable because the signature element v is revealed. The prover further randomizes the DH tuple by raising it to the power of y^{-1} to get $(v^{y^{-1}}, v^{xy^{-1}}, v^{ry^{-1}}, v^{rxy^{-1}})$ and constructs the Decisional DH tuple $(v^r, v^{rx}, v^{ry^{-1}}, v^{rxy^{-1}})$ for proving to the verifier. This is exactly exploiting the random self-reducibility of a Decisional DH tuple. □

M. Representation Hiding

Lemma 6. *The randomization in the ReCS showing protocol is perfectly hiding.*

Proof. The blinding factors $r, \{r_i, r_{(i,j)}\}$ act as the opening values and turns the $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ elements at the left-hand side into MoniPoly commitments which are perfectly hiding. Specifically, in `possession`, the random values $r, y, r_i, r_{(i,j)}$ turns the $\mathbb{G}_1, \mathbb{G}_2$ elements:

$$W_i = \left(a_{i_0}^{\prod_{w \in f_{\mathcal{V}^{(i)}}}(x'+w)}\right)^{r r_i^{-1}}, W_{(i,j)} = \left(a_{(i,j)_0}^{(x'+j)\prod_{w \in f_{\mathcal{E}^{(i,j)}}}(x'+w)}\right)^{r r_{(i,j)}^{-1}}$$

$$\left(X_{0_0}^{x'+i}\right)^{r_i}, \left(X_{0_0}^{x'+i}\right)^{r_{(i,j)}}$$

into MoniPoly set commitments and the $\mathbb{G}_1$ elements $b^{sr} c^r v'^{-ty}$ into a Pedersen vector commitment (Lemma 2). Therefore, these values inherit the perfectly hiding property from the two commitment schemes.

When the elements above are evaluated under pairing operations, as r is uniformly chosen, the $\mathbb{G}_T$ elements:

$$\mathsf{e}\left(a_{i_0}^{r_i^{-1}\prod_{w\in f_{\mathcal{V}^{(i)}}(x'+w)}}, X_{0_0}^{r_i(x'+i)}\right)^r,$$
$$\mathsf{e}\left(a_{(i,j)_0}^{r_{(i,j)}^{-1}(x'+j)\prod_{w\in f_{\mathcal{E}^{(i,j)}}(x'+w)}}, X_{0_0}^{r_{(i,j)}(x'+i)}\right)^r$$

are uniformly distributed under $\mathbb{G}_T$ regardless the combinations of vertex and edge identifiers $i, j, (i, j)$ and random exponents $r_i, r_{(i,j)}$ in a pairing function.

The proof for other predicates follow the same approach. □

To show the PUNL-ACA security, it is sufficient to show that the witnesses, the committed dataset in the signing protocols and showing protocols are perfectly hiding. Then, we demonstrate that every instance of the protocols is uniformly distributed due to the random self-reducibility property. This implies that even when $\mathcal{A}$ is given the master secret key and access to the issue oracle, it does not gain an advantage in linking a user's transaction. The security proof for PUNL-ACA dependent on (i) the random self-reducibility of the issuing and showing protocols (Lemmas 4 and 5). (ii) the perfect hiding property of the committed graph and randomized graph signatures (Lemma 6) and

Recall Theorem 7 from Section VI-B:

Theorem 7 (PUNL-ACA)**.** *The proposed relational credential system ReCS is protocol unlinkable under active and concurrent attacks.*

Proof. We show that an adversary $\mathcal{A}$ can win in the PUNL-ACA-security game with a negligible advantage $\varepsilon_{\mathsf{punl}}$ only, with $\mathcal{C}$ as the ReCS system simulator.

Game$_0$**.** This is an attack on the original ReCS scheme. Let S_i denotes a successful distinguishing attempt in **Game**$_i$, by definition we have:

$$\Pr\left[\mathbf{Game}^{\mathsf{punl-aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}\right] = \Pr[S_0] \leq \varepsilon_{\mathsf{punl}} + \frac{1}{2}. \quad (12)$$

Game$_1$**.** $\mathcal{C}$ selects two random bits $b_1, b_2 \in \{0, 1\}$ and generates (mpk, msk) as in the original algorithm and forwards to $\mathcal{A}$ so that the latter can play the role of users and signers. In addition, $\mathcal{C}$ maintains the corrupted user lists CU, CPV and honest user lists HU, HPV required by the oracles. Since $\mathcal{C}$ does not alter the key generation algorithm, this gives:

$$\Pr[S_1] = \Pr[S_0]. \quad (13)$$

Game$_2$**.** In the u-th session, $\mathcal{A}$ can retrieve an issuing protocol transcript $\pi_u \leftarrow \mathcal{O}_{\mathtt{Issuing}}(mpk, \mathcal{G}^{\mathtt{U}}_u, \mathcal{G}^{\mathtt{I}}_u, sid_u)$ and subsequently obtain $cred_u \leftarrow \mathcal{O}_{\mathtt{CorrU}}(\pi_u)$. Without lost of generality, we assume $\mathcal{A}$ always uses different graphs $(\mathcal{G}^{\mathtt{U}}_u, \mathcal{G}^{\mathtt{I}}_u)$. $\mathcal{A}$ can also acts as the issuer through $\mathcal{O}_{Issue}(mpk, msk, \mathcal{G}^{\mathtt{U}}, \mathcal{G}^{\mathtt{I}}, sid)$ oracle to interact with users which are simulated by $\mathcal{C}$. As every signing protocol session is uniformly distributed by Lemma 4 and the user hidden graph $\mathcal{G}^{\mathtt{U}}$ is perfectly hidden by Lemma 2, $\mathcal{A}$ does not gain any advantage in the two operations above:

$$\Pr[S_2] = \Pr[S_1]. \quad (14)$$

Game$_3$**.** Now $\mathcal{A}$ also queries for $\tilde{\pi}_u \leftarrow \mathcal{O}_{\mathtt{Verify}}(mpk, \mathcal{G}^{\mathtt{U}}_u, \mathcal{G}^{\mathtt{I}}_u, \phi_u, sid_u)$ concurrently to interact with $\mathcal{C}$ which acts as the provers. $\mathcal{A}$ can also query for $cred_u \leftarrow \mathcal{O}_{\mathtt{CorrP}}(\tilde{\pi}_u)$. As $\mathcal{C}$ knows $cred_u$ and every protocol session is uniformly distributed by Lemma 5, the interaction is the same as in the original show proof from the view of $\mathcal{A}$. Following Lemma 1 and 6, this is true even if $\mathcal{A}$ knows some credentials $cred_u$ from the issuing protocols, which now have been perfectly hidden by the randomly selected $r, y \in \mathbb{Z}^*_p$. This gives:

$$\Pr[S_3] = \Pr[S_2]. \quad (15)$$

Game$_4$**.** At some point, $\mathcal{A}$ decides the challenge $(\mathcal{G}^{\mathtt{U}}_0, \mathcal{G}^{\mathtt{U}}_1, \phi^*)$ such that every $|\mathcal{G}^{\mathtt{U}}_0| = |\mathcal{G}^{\mathtt{U}}_1|$. If the condition is not met, $\mathcal{C}$ aborts as the challenge graphs can be trivially identified. $\mathcal{C}$ obtains the credential in the sequence of $cred_{b_1}$ and $cred_{1-b_1}$ with $\mathcal{A}$ as the issuer. $\mathcal{A}$ is allowed to add new graphs $\mathcal{G}^{\mathtt{I}}_{b_1}, \mathcal{G}^{\mathtt{I}}_{1-b_1}$ of arbitrary sizes to the the signatures as long as $|\mathcal{G}^{\mathtt{I}}_0| = |\mathcal{G}^{\mathtt{I}}_1|$ and $\phi^*(\mathcal{G}^{\mathtt{U}^*}_0 \cup \mathcal{G}^{\mathtt{I}}_0) = \phi^*(\mathcal{G}^{\mathtt{U}^*}_1 \cup \mathcal{G}^{\mathtt{I}}_1) = 1$. $\mathcal{C}$ aborts otherwise. Next, $\mathcal{C}$ completes the challenge showing protocol with $\mathcal{A}$ as the verifier in the sequence of $cred^*_{b_2}$ and $cred^*_{1-b_2}$. From time to time, $\mathcal{A}$ still can query the oracles as before with the restriction of querying the challenge transcripts to the corruption oracles. In a matter of fact, even if $\mathcal{A}$ issue such queries, it gets only error reply $\perp$ from the corruption oracles. This is because the challenge issuing and showing transcripts are not recorded in the oracle list as they are not generated through querying oracles. Finally, if $\mathcal{A}$ makes a correct guess $b' = b_1 \oplus b_2$, it wins the game with the probability:

$$\Pr[S_4] = \Pr[S_3] = \Pr[b' = b_1 \oplus b_2] = \frac{1}{2} + \varepsilon_{\mathsf{punl}}. \quad (16)$$

Combining the probability from equation (12) to (16), we have a negligible $\varepsilon_{\mathsf{punl}}$ as required and $\mathcal{A}$ runs in time t_{punl}. □

We design the impersonation resilience security of the ReCS system based on a reduction to the (co-)SDH problem. We utilise the Multi-Instance Reset Lemma [44] as the knowledge extractor for achieving tight security reduction. Under the setting of this lemma, the adversary $\mathcal{A}$ can run N parallel instances of impersonation under active and concurrent attacks. The challenger $\mathcal{C}$ can simulate such environment by exploiting the random self-reducibility of the given SDH instance [43].

Recall Theorem 8 from Section VI-B

Theorem 8 (IMP-ACA)**.** *The proposed relational credential system ReCS is secure against impersonation under active and concurrent attacks, if the co-SDH problem is intractable.*

More formally, this theorem translates to the following:

Theorem 15. *If an adversary $\mathcal{A}$ $(t_{\mathsf{imp}}, \varepsilon_{\mathsf{imp}})$-breaks the imp-aca-security of the proposed ReCS system, then there exists an algorithm $\mathcal{C}$ which $(t_{\mathsf{cosdh}}, \varepsilon_{\mathsf{cosdh}})$-breaks the q-co-SDH problem such that:*

$$\frac{\varepsilon_{\mathsf{cosdh}}}{t_{\mathsf{cosdh}}} = \frac{\varepsilon_{\mathsf{imp}}}{t_{\mathsf{imp}}},$$

or an algorithm $\mathcal{C}$ which $(t_{\mathsf{sdh}}, \varepsilon_{\mathsf{sdh}})$-breaks the SDH problem such that:

$$\frac{\varepsilon_{\mathsf{sdh}}}{t_{\mathsf{sdh}}} \geq \frac{1}{3 \cdot 2Nt_{\mathsf{imp}}}$$
$$\frac{6\varepsilon_{\mathsf{sdh}}}{t_{\mathsf{sdh}}} \geq \frac{\varepsilon_{\mathsf{imp}}}{t_{\mathsf{imp}}} - \frac{1 + (q-1)!/p^{q-2}}{t_{\mathsf{imp}} p}$$

where N is the total adversary instance, q is the total credential simulated throughout the game and p is the group order.

We differentiate the ReCS adversary $\mathcal{A}$ into $\mathcal{A} = \{\mathcal{A}_{bind}, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3\}$ corresponding to four different simulation strategies by the challenger $\mathcal{C}$:

- $\mathcal{A}_{bind}$: breaks the binding security of the MoniPoly multi-set commitment.
- $\mathcal{A}_1$: impersonate with credential element $t^* \notin CU \cup CPV \cup HU \cup HPV$ is new to $\mathcal{C}$
- $\mathcal{A}_2$: impersonate with credential element $t^* \in CU \cup CPV \cup HU \cup HPV$ is known to $\mathcal{C}$
- $\mathcal{A}_1$: impersonate with credential elements $s^*, t^* \in CU \cup CPV \cup HU \cup HPV$ are known to $\mathcal{C}$

While $\mathcal{A}_{bind}$ has been described in Theorem 3, we present Lemma 7, 8 and 9 corresponding to the adversaries $\mathcal{A}_1$, $\mathcal{A}_2$ and $\mathcal{A}_3$.

Lemma 7. *If an adversary $\mathcal{A}_1$ $(t_{\mathsf{imp}}, \varepsilon_{\mathsf{imp}})$-breaks the imp-aca-security of the proposed ReCS, then there exists an algorithm $\mathcal{C}$ which $(t_{\mathsf{sdh}}, \varepsilon_{\mathsf{sdh}})$-solves the SDH problem such that:*

$$\varepsilon_{\mathsf{imp}} \leq \sqrt[N]{\sqrt{\varepsilon_{\mathsf{sdh}}} - 1} + \frac{q! + p^{q-1}}{p^q} + 1,$$
$$t_{\mathsf{imp}} \leq t_{\mathsf{sdh}}/2N - T(q^2).$$

where N is the total of adversary instances, q is the number of signature queries allowed, while $T(q^2)$ is the time parametrized by q to setup the simulation environment and to extract the SDH solution.

Proof. Given a q-SDH instance $(g_1, g_1^x, g_1^{x^2}, \ldots, g_1^{x^q}, g_2, g_2^x)$ where q is the maximum number of signature queries allowed, we show that if $\mathcal{A}_1$ exists, there exists an algorithm $\mathcal{C}$ which can output $(g_1^{\frac{1}{x+t}}, t)$ by acting as the simulator for the ReCS as follows:

Game$_0$**.** This is the attack by $\mathcal{A}_1$ on the real N instances of ReCS. Let S_i be the event of a successful impersonation in **Game**$_i$, by assumption, we have:

$$\Pr\left[\mathbf{Game}^{\mathsf{imp-aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}\right] = \Pr[S_0] = \varepsilon_{\mathsf{imp}}. \quad (17)$$

Game$_1$**.** To simulate the environment of the ReCS, for the first (out of the total N) instance, $\mathcal{C}$ uniformly and randomly selects distinct $\{t_{a_i}\}_{i=0}^{L}, t_b, t_c, x', t_1, \ldots, t_q \in \mathbb{Z}_p^*$. Next, let $f(x)$ denotes the polynomial $f(x) = \prod_{k=1}^{q}(x + t_k)$ and $f_u(x)$ denotes the polynomial $f_u(x) = \prod_{k=1, k\neq u}^{q}(x + t_k)$. $\mathcal{C}$ computes master public key as $mpk = (\mathsf{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, \{\{a_{i_k} = g_1^{f(x)t_{a_i}x'^k}, X_{i_k} = g_2^{t_{a_i}x'^k}\}_{i=0}^{L}\}_{k=0}^{n}, b = g_1^{f(x)t_b}, c = g_1^{f(x)t_c}, \{X_i\}_{i=1}^{L}, X = (g_2^x)^{t_{a_0}})$ to implicitly set the master secret key $msk = (x)$. $\mathcal{C}$ sends mpk to the adversary $\mathcal{A}_1$ and creates the four empty lists (CU, CPV, HU, HPV) as defined in the Table III. Since $t_{a_i}, t_b, t_c, x', t_1, \ldots, t_q$ are uniformly random, the distribution of the simulated mpk is the same as that of the original scheme. So, we have:

$$\Pr[S_1] = \Pr[S_0]. \quad (18)$$

Game$_2$**.** In this game, $\mathcal{A}_1$ issues concurrent queries for $\{\pi_u\} \leftarrow \mathcal{O}_{\mathtt{Issuing}}(mpk, \mathcal{G}_u^{\mathtt{U}}, \mathcal{G}_u^{\mathtt{I}}, sid_u)$. Subsequently, $\mathcal{A}_1$ can obtain the corresponding credential $\{cred_u\} \leftarrow \mathcal{O}_{\mathtt{CorrU}}$. Without loss of generality, we assume different $(\mathcal{G}_u^{\mathtt{U}}, \mathcal{G}_u^{\mathtt{I}})$ are used in every sid_u session of the u-th query. If $\mathcal{A}_1$ interacts with the signer simulated by $\mathcal{C}$ without going through oracles, their interaction during an issuing protocol is as follows:

1) $\mathcal{A}_1$ concurrently initializes the initial signing protocol with $\mathcal{C}$ by running the zero-knowledge protocol to prove the knowledge for its hidden graph $\mathcal{G}_u^{\mathtt{U}}$ and the secret exponent $s'_u \in \mathbb{Z}_p^*$. Without loss of generality, we assume $\mathcal{A}_1$ always executes this protocol honestly. Therefore, $\mathcal{C}$ always resets successfully to extract $\mathcal{G}_u^{\mathtt{U}}$ and s'_u.
2) $\mathcal{C}$ chooses a random graph $\mathcal{G}_u^{\mathtt{I}}$ and random $s''_u \in \mathbb{Z}_p^*$ to set:

$$v_u = \prod_{i \in V_u} a_{u,i_0}^{(x'+i)\prod_{m \in f_{\mathcal{V}^{(i)}}}(x'+m)} \prod_{(i,j) \in E_u} a_{u,(i,j)_0}^{(x'+i)(x'+j)\prod_{m \in f_{\mathcal{E}^{(i,j)}}}(x'+m)} b_u^{s'_u + s''_u} c_u$$

where $a_{u,i_0} = g_1^{f_u(x)t_{a_i}}, a_{u,(i,j)_0} = g_1^{f_u(x)t_{a_{(i,j)}}}, b_u = g_1^{f_u(x)t_b}, c_u = g_1^{f_u(x)t_c}$. $\mathcal{C}$ adds the record $(sid_u, \mathcal{G}_u^{\mathtt{U}}, \mathcal{G}_u^{\mathtt{I}}, cred_u = (\sigma_u = (t_u, s_u = s'_u + s''_u, v_u), \mathcal{G}_u^{\mathtt{U}} \cup \mathcal{G}_u^{\mathtt{I}}), \pi_u)$ to CU and returns $\sigma' = (t_u, s''_u, v_u)$ and $\mathcal{G}_u^{\mathtt{I}}$ to $\mathcal{A}_1$.

Since $\mathcal{C}$'s choices of t_u, s''_u are independent from $\mathcal{A}_1$'s view, a collision $v_i = v_j$ for some $i, j \leq q$ in a round of $\mathcal{A}_1$'s concurrent queries happens with a negligible probability. $\mathcal{A}_1$ can formulate $cred_u = (sig_u, \mathcal{G}_u^{\mathtt{U}} \cup \mathcal{G}_u^{\mathtt{I}})$ as in the original signing protocol. This gives:

$$\begin{aligned}\Pr[S_2] &= \Pr[S_1] + \Pr[Col] \\ &\leq \Pr[S_1] + \prod_{i=1}^{q} i/p \\ &\leq \Pr[S_1] + q!/p^q \end{aligned} \quad (19)$$

where $\mathcal{A}_1$ can ask for, at most, q credentials.

Game$_3$**.** In this game, $\mathcal{A}_1$ issues concurrent queries for $\{\tilde{\pi}_u\} \leftarrow \mathcal{O}_{\mathtt{Verify}}(mpk, \mathcal{G}_u^{\mathtt{U}}, \mathcal{G}_u^{\mathtt{I}}, \phi, sid_u)$ where it plays the role of verifiers to interact with proves simulated by $\mathcal{C}$ concurrently. When $\mathcal{A}_1$ asks for a show proof on $(\mathcal{G}_u^{\mathtt{U}}, \mathcal{G}_u^{\mathtt{I}}, \phi_u)$, we assume $\mathcal{C}$ already has the appropriate credential $cred_u$ on hand. Else, $\mathcal{C}$ simulates a valid $cred_u$ as in the previous games and adds it to HU before interacting with $\mathcal{A}_1$. If the protocol ends successfully, the showing transcript $\tilde{\pi}_u$ is added to HPV. Subsequently, $\mathcal{A}_1$ can also query for the corresponding credentials $cred_u \leftarrow \mathcal{O}_{\mathtt{CorrP}}(\tilde{\pi}_u)$. Therefore, we have:

$$\Pr[S_3] = \Pr[S_2]. \quad (20)$$

Game$_4$**.** $\mathcal{A}_1$ decided to impersonate a user whose graph $\mathcal{G}^* = \{V^*, E^*, f_\mathcal{V}, f_\mathcal{E}\}$ can satisfy its intended predicate ϕ^* such that $\phi^*(\mathcal{G}^*) = 1$. If there exits $\mathcal{G}_u^{\mathtt{U}}, \mathcal{G}_u^{\mathtt{I}} \in CU \cup CPV$ such that $\phi^*(\mathcal{G}_u^{\mathtt{U}} \cup \mathcal{G}_u^{\mathtt{I}}) = 1$, $\mathcal{C}$ aborts. This means the `possession` predicate is excluded as `possession` always returns true for all corrupted credentials. $\mathcal{A}_1$ is still allowed to query the oracles as in the previous games but the queries are bounded by the restrictions on $(\mathcal{G}^*, \phi^*)$ above. Essentially, $\mathcal{A}_1$ can query for the transcripts involving $(\mathcal{G}^*, \phi^*)$ but it is not allowed to recover the corresponding $cred^*$ from $\mathcal{O}_{\mathtt{CorrU}}$ and $\mathcal{O}_{\mathtt{CorrP}}$.

Finally, if $\mathcal{A}_1$ completes a showing protocol with ϕ^* such that the verifier $\mathcal{C}$ returns 1, $\mathcal{C}$ obtains a valid showing protocol transcript $\tilde{\pi}_1^*$. $\mathcal{C}$ resets $\mathcal{A}_1$ to the beginning of the game and activates it in parallel using the $N-1$ instances $\{mpk_i\}_{i=2}^N$ randomized through random self-reduction [43] of the given SDH instance. In essence, the polynomial $f(x)$ in the i-th instance is constructed as $f(xx_i)$ and we have $X_i = X^{x_i}$ for randomly selected $\{x_i \in \mathbb{Z}_p^*\}_{2 \le i \le N}$. If $\mathcal{C}$ obtains another valid transcript $\tilde{\pi}_2^*$ from one of the $N-1$ instances, it can extract the secret exponents (r^*, y^*, t^*, s^*) and subsequently recover the credential elements (v_1^*, v_2^*). Let

$$Z = \sum_{i \in V^*} t_{a_i}(x'+i) \prod_{m \in f_\mathcal{V}} (x'+m) + \sum_{(i,j) \in E^*} t_{a_{(i,j)}}(x'+i)(x'+j) \prod_{m \in f_\mathcal{E}} (x'+m) + t_b s^* + t_c,$$

$\mathcal{C}$ can now reconstruct $cred_1^* = (t^*, s^*, v^* = v_1^*, \mathcal{G}^*)$ used by $\mathcal{A}_1$ in $\tilde{\pi}_1^*$ where $v^* = g_1^{\frac{f(x)Z}{x+t^*}}$. Since $\mathcal{A}_1$ must output $t^* \notin \{t_1, \ldots, t_q\}$, if $v^* \notin CU \cup CPV \cup HU \cup HPV$, $\mathcal{C}$ can construct a polynomial $c(x)$ of degree $n-1$ such that $f(x) = c(x)(x+t^*) + r$ to compute:

$$\begin{aligned} v^{*1/Zr} g_1^{-\frac{c(x)}{r}} &= g_1^{\frac{f(x)Z}{Zr(x+t^*)} - \frac{c(x)}{r}} \\ &= g_1^{\frac{c(x)(x+t^*)+r}{r(x+t^*)} - \frac{c(x)}{r}} = g_1^{\frac{1}{x+t^*}} \end{aligned}$$

and output $(g_1^{\frac{1}{x+t^*}}, t^*)$ as the solution for the SDH instance. On the other hand, if we have $v^* \in CU \cup CPV \cup HU \cup HPV$, $\mathcal{C}$ can extract the discrete logarithm x to break the SDH assumption.

By Multi-Instance Reset Lemma [44], we obtain:

$$\Pr[S_4] \le \Pr[S_3] + \sqrt[N]{\sqrt{\varepsilon_{\mathsf{sdh}}} - 1} + 1/p + 1. \tag{21}$$

Summing up the probability from (17) to (21), we have $\varepsilon_{\mathsf{imp}} \le \sqrt[N]{\sqrt{\varepsilon_{\mathsf{sdh}}} - 1} + 1/p + 1 + q!/p^q$ as required. The time taken by $\mathcal{C}$ is at least $2Nt_{\mathsf{imp}}$ due to reset and interacting with N parallel impersonation instances, in addition to the environment setup and the final SDH solution extraction that cost $T(q^2)$.

□

Lemma 8. *If an adversary $\mathcal{A}_2$ $(t_{\mathsf{imp}}, \varepsilon_{\mathsf{imp}})$-breaks the imp-aca-security of the proposed ReCS, then there exists an algorithm $\mathcal{C}$ which $(t_{\mathsf{sdh}}, \varepsilon_{\mathsf{sdh}})$-solves the SDH problem such that:*

$$\varepsilon_{\mathsf{imp}} \le \sqrt[N]{\sqrt{\varepsilon_{\mathsf{sdh}}} - 1} + \frac{q! + p^{q-1}}{p^q} + 1,$$
$$t_{\mathsf{imp}} \le t_{\mathsf{sdh}}/2N - T(q^2).$$

where N is the total of adversary instances, q is the number of signature queries allowed, while $T(q^2)$ is the time parametrized by q to setup the simulation environment and to extract the SDH solution.

Proof. Given a q-SDH instance $(g_1, g_1^x, g_1^{x^2}, \ldots, g_1^{x^q}, g_2, g_2^x)$ where q is the number of signature queries allowed, there exists an algorithm $\mathcal{C}$ which can output $(g_1^{\frac{1}{x+t}}, t)$ by acting as the simulator for the ReCS as follows:

Game$_0$**.** This is the same as the **Game**$_0$ in Lemma 7 where we have:

$$\Pr\left[\mathbf{Game}_{\mathsf{ReCS},\mathcal{A}}^{\mathsf{imp-aca}} = \mathsf{Win}\right] = \Pr[S_0] = \varepsilon_{\mathsf{imp}}. \tag{22}$$

Game$_1$. The setting for $\{a_{i_k}, X_{i_k}\}, X$ is the same as the **Game**$_1$ in Lemma 7 but not for (b, c). Let $f_{u',u}(x)$ denotes the polynomial $f_{u',u}(x) = \prod_{k=1, k \ne u', u}^q (x + t_k)$. $\mathcal{C}$ uniformly selects random distinct $s_1, \ldots, s_q \in \mathbb{Z}_p^*$ to set $(b = g_1^{f(x)t_b - \sum_{u=1}^q f_u(x)}, c = g_1^{f(x)t_c + \sum_{u=1}^q s_u f_u(x)})$. $\mathcal{C}$ then sends mpk to $\mathcal{A}_2$. This gives:

$$\Pr[S_1] \le \Pr[S_0]. \tag{23}$$

Game$_2$. This is the same as the **Game**$_2$ in Lemma 7 except that, after resetting $\mathcal{A}_2$, $\mathcal{C}$ simulates the pre-signature $\sigma'_u = (t_u, s''_u, v_u)$ for $(\mathcal{G}_u^{\mathtt{U}}, \mathcal{G}_u^{\mathtt{I}})$ such that:

$$v_u = \prod_{i \in V_u} a_{u,i_0}^{(x'+i)\prod_{m \in f_\mathcal{V}(i)}(x'+m)} \prod_{(i,j) \in E_u} a_{u,(i,j)_0}^{(x'+i)(x'+j)\prod_{m \in f_\mathcal{E}(i,j)}(x'+m)} b_u^{s'_u + s''_u} c_u$$

where $s''_u = s_u - s'_u$. When the protocol ends, $\mathcal{A}_2$ obtains the graph signature $\sigma_u = (t_u, s_u = s'_u + s''_u, v_u)$ and a graph $\mathcal{G}_u^{\mathtt{I}}$. As $\mathcal{C}$ simulates issuing perfectly, we have:

$$\Pr[S_2] \le \Pr[S_1] + q!/p^q. \tag{24}$$

where $\mathcal{A}_1$ can make, at most, q signature queries.

Game$_3$ This is the same as that in Lemma 7 and we have:

$$\Pr[S_3] = \Pr[S_2]. \tag{25}$$

Game$_4$**.** Similar to the **Game**$_4$ in Lemma 7, $\mathcal{C}$ can extract the elements (t^*, s^*, v^*) of σ^* from the Multi-Instance Reset Lemma. Since $\mathcal{A}_2$ must output $t^* = t_u \in \{t_1, \ldots, t_q\}$ but $s^* \ne s_u \in \{s_1, \ldots, s_q\}$ for an $u \in \{1, \ldots, q\}$, v^* is in the form:

$$v^* = \left(g_1^{f(x)Z + \sum_{u'=1, u' \ne u}^q (s_{u'} - s^*) f_{u',u}(x) + (s_u - s^*) f_u(x)}\right)^{1/(x+t_u)}.$$

where

$$Z = \sum_{i \in V} (t_{a_i}(x'+i) \prod_{m \in f_\mathcal{V}(i)} (x'+m)) + \sum_{(i,j) \in E} (t_{a_{(i,j)}}(x'+i)(x'+j) \prod_{m \in f_\mathcal{V}(i)} (x'+m)) + s^* t_b + t_c.$$

$\mathcal{C}$ proceeds to compute $c(x)$ of degree $q-2$ and $r \in \mathbb{Z}_p^*$ from the knowledge of $\{t_1, \ldots, t_q\}$ such that $f_u(x) = c(x)(x +$

$t_u) + r$. Moreover, it will be the case $v^* \notin CU \cup CPV \cup HU \cup HPV$ or $\mathcal{C}$ can extract x to solve the SDH problem. Subsequently, $\mathcal{C}$ calculates:

$$\left(v^*/g_1^{f_u(x)Z+\sum_{u'=1,u'\neq u}^{q}(s_{u'}-s^*)f_{u',u}(x)+c(x)(s_u-s^*)}\right)^{\frac{1}{r(s_u-s^*)}}$$
$$= g_1^{\frac{(f_u(x)-c(x)(x+t_u))(s_u-s^*)}{r(s_u-s^*)(x+t_u)}}$$
$$= g_1^{\frac{1}{x+t_u}}$$

and outputs $(g_1^{\frac{1}{x+t_u}}, t_u)$ as the solution for the SDH instance. Therefore, we have:

$$\Pr[S_4] \leq \Pr[S_3] + \sqrt[N]{\sqrt{\varepsilon_{\mathsf{sdh}}} - 1} + 1/p + 1 \tag{26}$$

and summing up the probability from (23) to (26), we have $\varepsilon_{\mathsf{imp}} \leq \sqrt[N]{\sqrt{\varepsilon_{\mathsf{sdh}}} - 1} + 1/p + 1 + q!/p^q$ as required. The time taken by $\mathcal{C}$ is at least $2Nt_{\mathsf{imp}}$ due to reset and interacting with N parallel impersonation instances, in addition to the environment setup and the final SDH solution extraction that cost $T(q^2)$.

□

Lemma 9. *If an adversary $\mathcal{A}_3$ $(t_{\mathsf{imp}}, \varepsilon_{\mathsf{imp}})$-breaks the imp-aca-security of the proposed graph signature system, then there exists an algorithm $\mathcal{C}$ which $(t_{\mathsf{sdh}}, \varepsilon_{\mathsf{sdh}})$-solves the SDH problem such that:*

$$\varepsilon_{\mathsf{imp}} \leq \sqrt[N]{\sqrt{\varepsilon_{\mathsf{sdh}}} - 1} + \frac{q! + p^{q-1}}{p^q} + 1,$$
$$t_{\mathsf{imp}} \leq t_{\mathsf{sdh}}/2N - T(q^2).$$

where N is the total of adversary instances, q is the number of signature queries allowed, while $T(q^2)$ is the time parametrized by q to setup the simulation environment and to extract the SDH solution.

Proof. Given a q-SDH instance $(g_1, g_1^x, g_1^{x^2}, \ldots, g_1^{x^q}, g_2, g_2^x)$ where q is the number of signature queries allowed, there exists an algorithm $\mathcal{C}$ which can output $(g_1^{\frac{1}{x+t}}, t)$ by acting as the simulator for the ReCS as follows:

Game$_0$**.** This is the same as the **Game**$_0$ in Lemma 7:

$$\Pr\left[\mathbf{Game}^{\mathsf{imp-aca}}_{\mathsf{ReCS},\mathcal{A}} = \mathsf{Win}\right] = \Pr[S_0] = \varepsilon_{\mathsf{imp}}. \tag{27}$$

Game$_1$. The precomputations and setting are the same as the **Game**$_1$ in Lemma 7 except for $(\{\{a_{i_k} = g_1^{(f(x)t_{a_i} - \sum_{k=1}^{q} f_k(x))x'^k}\}_{i=0}^{L}\}_{k=0}^{n}$, $b = g_1^{f(x)t_b - \sum_{u=1}^{q} f_u(x)}$, $c = g_1^{f(x)t_c + \sum_{u=1}^{q} z_u f_u(x)})$ where the random $z_1, \ldots, z_q \in \mathbb{Z}_p^*$ are uniformly distributed. This gives:

$$\Pr[S_1] \leq \Pr[S_0]. \tag{28}$$

Game$_2$. This is the same as the **Game**$_2$ in Lemma 7 except that, after resetting $\mathcal{A}_3$, $\mathcal{C}$ simulates the pre-signature $\sigma'_u = (t_u, s''_u, v_u)$ for $(\mathcal{G}_u^{\mathtt{U}}, \mathcal{G}_u^{\mathtt{I}})$ by letting

$$s_u = z_u - \sum_{i \in V_u} t_{a_i}(x'+i) \prod_{m \in f_{\mathcal{V}}(i)} (x'+m) -$$
$$\sum_{(i,j) \in E_u} t_{a_{(i,j)}}(x'+i)(x'+j) \prod_{m \in f_{\mathcal{E}}(i,j)} (x'+m)$$

to simulate

$$v_u = \prod_{i \in V_u} a_{u,i_0}^{(x'+i)\prod_{w \in f_{\mathcal{V}}(i)}(x'+w)}$$
$$\prod_{(i,j) \in E_u} a_{u,(i,j)_0}^{(x'+i)(x'+j)\prod_{w \in f_{\mathcal{E}}(i,j)}(x'+w)} b_u^{s'_u + s''_u} c_u$$

for $s''_u = s_u - s'_u$. When the protocol ends, $\mathcal{A}_3$ obtains the credential as $\sigma_u = (t_u, s_u, v_u)$ on the graph $\mathcal{G}_u^{\mathtt{U}} \cup \mathcal{G}_u^{\mathtt{I}}$. As $\mathcal{C}$ simulates the issuing perfectly, we have:

$$\Pr[S_2] \leq \Pr[S_1] + q!/p^q. \tag{29}$$

where $\mathcal{A}_1$ can ask for, at most, q credentials.

Game$_3$ This is the same as that in Lemma 7:

$$\Pr[S_3] = \Pr[S_2]. \tag{30}$$

Game$_4$**.** By definition, $\mathcal{A}_3$ must output $t^* = t_u \in \{t_1, \ldots, t_q\}$ and $s^* = s_u \in \{s_1, \ldots, s_q\}$ for a $u \in \{1, \ldots, q\}$. Note that it must be the case $v^* \notin CU \cup CPV \cup HU \cup HPV$ or x can be extracted to solve the SDH problem. In the unlikely case of $(\mathcal{G}^*, s^*, t^*, v^*) \in HU \cup HPV$ which happens with probability $1/p$, $\mathcal{C}$ aborts. Similar to the **Game**$_4$ in Lemma 7, $\mathcal{C}$ can extract the signature elements (t^*, s^*, v^*) through the Multi-Instance Reset Lemma where v^* is in the form:

$$v^* = g_1^{f_u(x)(Z+s^*t_b+t_c)+\sum_{u'=1,u'\neq u}^{q}(z_k-z^*)f_{u',u}(x)+(z_u-z^*)f_u(x)}$$

for

$$Z = \sum_{i \in V} t_{a_i}(x'+i) \prod_{m \in f_{\mathcal{V}}(i)} (x'+m) -$$
$$\sum_{(i,j) \in E} t_{a_{(i,j)}}(x'+i)(x'+j) \prod_{m \in f_{\mathcal{E}}(i,j)} (x'+m)$$

and $z^* = s^* + Z$. $\mathcal{C}$ proceeds to compute $c(x)$ of degree $q-2$ and the remainder $r \in \mathbb{Z}_p^*$ from the knowledge of $\{t_1, \ldots, t_q\}$ such that $f_u(x) = c(x)(x+t_u)+r$. Subsequently, $\mathcal{C}$ calculates:

$$\left(v^* g_1^{-f_u(x)(Z+s^*t_b+t_c)-\sum_{k=1,k\neq u}^{q}(z_k-z^*)f_{k,u}(x)+(z_k-z^*)c(x)}\right)^{\frac{1}{r(z_u-z^*)}}$$
$$= g_1^{\frac{(f_u(x)-c(x)(x+t_u))(z_u-z^*)}{(x+t_u)r(z_u-z^*)}}$$
$$= g_1^{\frac{1}{x+t_u}}$$

and outputs $(g_1^{\frac{1}{x+t_u}}, t_u)$ as the solution for the SDH instance. Therefore, we have:

$$\Pr[S_4] \leq \Pr[S_3] + \sqrt[N]{\sqrt{\varepsilon_{\mathsf{sdh}}} - 1} + 1 \tag{31}$$

and summing up the probability from (27) to (31), we have $\varepsilon_{\mathsf{imp}} \leq \sqrt[N]{\sqrt{\varepsilon_{\mathsf{sdh}}} - 1} + 1/p + 1 + q!/p^q$ as required. The time taken by $\mathcal{C}$ is at least $2Nt_{\mathsf{imp}}$ due to reset and interacting with N parallel impersonation instances, in addition to the environment setup and the final SDH solution extraction which cost $T(q^2)$.

□

Combining Theorem 3, Lemmas 7, 8, and 9 gives Theorem 15 as required.

The detailed connection proof is at below:

$$PK\Bigg\{(\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}), (\forall (i,j) \in E \setminus E' \} : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}),$$
$$\{\varepsilon_{l,0}\}_{l=2}^{\ell-1}, \{\varepsilon_{l,1}\}_{l=1}^{\ell-1}, \varepsilon, \rho, \omega, \tau, \gamma) :$$
$$\prod_{i \in V} \mathsf{e}\left(W_i, X_{0_1}^{\varepsilon_{i_1}} X_{0_0}^{\varepsilon_{i_0}}\right) \prod_{(i,j) \in E \setminus E'} \mathsf{e}\left(W_{(i,j)}, X_{0_1}^{\varepsilon_{(i,j)_1}} X_{0_0}^{\varepsilon_{(i,j)_0}}\right). \tag{32}$$
$$\mathsf{e}\left(W'_1, (X_{0_2} X_{0_1}^{i^*})^{\varepsilon} (X_{0_1} X_{0_0}^{i^*})^{\varepsilon_{1,1}}\right). \tag{33}$$
$$\mathsf{e}\left(W'_2, X_{0_2}^{\varepsilon} X_{0_1}^{\varepsilon_{1,1}} X_{0_1}^{\varepsilon_{2,1}} X_{0_0}^{\varepsilon_{2,0}}\right) \cdots\cdots \tag{34}$$
$$\cdots \mathsf{e}\left(W'_{\ell-1}, X_{0_2}^{\varepsilon} X_{0_1}^{\varepsilon_{\ell-2,1}} X_{0_1}^{\varepsilon_{\ell-1,1}} X_{0_0}^{\varepsilon_{\ell-1,0}}\right). \tag{35}$$
$$\mathsf{e}\left(W'_\ell, (X_{0_2} X_{0_1}^{j^*})^{\varepsilon} (X_{0_1} X_{0_0}^{j^*})^{\varepsilon_{\ell-1,1}}\right). \tag{36}$$
$$\mathsf{e}\left(b^\rho c^\omega v'^{-\tau}, X_{0_0}\right) = \mathsf{e}(v'^{\gamma}, X) \wedge \tag{37}$$
$$\mathsf{e}\left(C_{\mathcal{V}} \prod_{l=1}^{\ell-2} W_{V_l}, X_{0_0}\right) = \tag{38}$$
$$\prod_{l=1}^{\ell-1} \mathsf{e}\left(W_{V_{l-1}}, X_{0_1}^{\varepsilon} X_{0_0}^{\varepsilon_{l,1}}\right) \mathsf{e}\left(a_{0_0}^{-1} W_{\mathcal{V}}, W_{V_{\ell-1}}\right) \Bigg\} \tag{39}$$

In this proof, Clauses (32) to (37) represent the `possession` predicate while Clause (39) is the proof of cumulative product for all vertex identifiers in E'. To be precise, Clause (32) shows the correctness for $\mathcal{G} \setminus E'$ and Clauses (33) to (36) show the correctness for E'. Specifically, Clause (33) verifies correctness of the first edge $E'_1 \in E'$, Clauses (34) and (35) verifies the correctness of $E'_2, \ldots, E'_{\ell-1}$ while Clause (36) verifies the correctness of the last edge $E'_\ell \in E$. Equality is proven, as the matching edges share the same randomness $r_{(i^*,j^*)} \in \mathbb{Z}_p^*$ where $W'_k = a_{(i_k,j_k)_0}^{r r_{(i^*,j^*)}^{-1} \prod_{m \in f_{\mathcal{E}}(i_k,j_k)}(x'+m)}$ for $1 \geq k \geq \ell$. Lastly, Clause (39) is the set membership proof to prove that it is the vertices but not labels that connect the edges. To be precise, it shows that the cumulative product of encoded vertex identifiers in E' is a subset of the vertex domain $\mathcal{V}$ where

$$W_{V_0} = a_{0_0}, \{W_{V_l} = a_{0_0}^{r_{(i^*,j^*)}^l \prod_{k=1}^{l}(x'+j_k)}\}_{l=1}^{\ell-2},$$
$$W_{V_{\ell-1}} = X_{0_0}^{r_{(i^*,j^*)}^{\ell-1} \prod_{k=1}^{\ell-1}(x'+j_k)}, W_{\mathcal{V}} = C_{\mathcal{V}}^{1/r_{(i^*,j^*)}^{\ell-1} \prod_{k=1}^{\ell-1}(x'+j_k)}$$

are witnesses for the cumulative product and $C_{\mathcal{V}} = a_{0_0}^{\prod_{m \in \mathcal{V}}(x'+m)} = \mathsf{SC.Commit}(pk, \mathtt{G2MS}(\mathcal{V}), 1^{|\mathcal{V}|})$ is a MoniPoly commitment.

The below executes the protocol for $\mathtt{cover}_{(\mathcal{G}',\ell)}(\mathcal{G})$, where V_i represents the set of vertex identifier and associated labels for vertex i and $E_{(i,j)}$ represents the set of edge identifiers and associated labels for edge (i,j).

$$PK\Bigg\{(\forall i \in V \setminus I : \varepsilon_{i_0}, \varepsilon_{i_1}), (\forall (i,j) \in E \setminus I : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}),$$
$$(\forall i \in I : \{\varepsilon_{i_k}\}_{k=0}^{|V'_i|}), (\forall (i,j) \in I : \{\varepsilon_{(i,j)_k}\}_{k=0}^{|E'_{(i,j)}|}),$$
$$\rho, \omega, \tau, \gamma, (\forall i \in V' \setminus I : \varepsilon_{i_0}, \varepsilon_{i_1}), \tag{40}$$
$$(\forall i \in V' : \alpha_i), (\forall (i,j) \in E'_{(i,j)} : \alpha_{(i,j)})) :$$
$$\prod_{i \in I} \mathsf{e}\left(W'_i, \prod_{k=0}^{|V'_i|} X_{0_k}^{\varepsilon_{i_k}}\right) \prod_{(i,j) \in I} \mathsf{e}\left(W'_{(i,j)}, \prod_{k=0}^{|E'_{(i,j)}|} X_{0_k}^{\varepsilon_{(i,j)_k}}\right). \tag{41}$$
$$\prod_{i \in (V \setminus I)} \mathsf{e}\left(W_i, X_{0_1}^{\varepsilon_{i_1}} X_{0_0}^{\varepsilon_{i_0}}\right) \prod_{(i,j) \in (E \setminus I)} \mathsf{e}\left(W_{(i,j)}, X_{0_1}^{\varepsilon_{(i,j)_1}} X_{0_0}^{\varepsilon_{(i,j)_0}}\right). \tag{42}$$
$$\mathsf{e}\left(b^\rho c^\omega v'^{-\tau}, X_{0_0}\right) = \mathsf{e}(v'^{\gamma}, X) \wedge \tag{43}$$
$$\prod_{i \in I} \mathsf{e}\left(a_{0_0}, \prod_{k=0}^{|V'_i|} X_{0_k}^{\varepsilon_{i_k}}\right) \prod_{(i,j) \in I} \mathsf{e}\left(a_{0_0}, \prod_{k=0}^{|E'_{(i,j)}|} X_{0_k}^{\varepsilon_{(i,j)_k}}\right). \tag{44}$$
$$\prod_{i \in (V' \setminus I)} \mathsf{e}\left(W'_i, X_{0_1}^{\varepsilon_{i_1}} X_{0_0}^{\varepsilon_{i_0}}\right) \prod_{(i,j) \in (E' \setminus I)} \mathsf{e}\left(W'_{(i,j)}, X_{0_1}^{\varepsilon_{(i,j)_1}} X_{0_0}^{\varepsilon_{(i,j)_0}}\right) = \tag{45}$$
$$\mathsf{e}\left(\prod_{i \in V'} C_i^{\alpha_i} \prod_{(i,j) \in E'} C_{(i,j)}^{\alpha_{(i,j)}}, X_{0_0}\right) \Bigg\} \tag{46}$$

where clauses (41) to (43) are the possession predicate. To be precise, clause (41) represents the intersecting multi-set $I = \mathcal{G}' \cap \mathcal{G}$ such that

$$W'_i = a_{i_0}^{r_i^{-1} r \prod_{m \in V_i \setminus V'_i}(x'+m)}, W'_{(i,j)} = a_{(i,j)_0}^{r_i^{-1} r \prod_{m \in E_{(i,j)} \setminus E'_{(i,j)}}(x'+m)},$$
$$\prod_{k=0}^{|V'_i|} X_{0_k}^{\varepsilon_{i_k}} = X_{0_0}^{r_i \prod_{m \in V'_i}(x'+m)}, \prod_{k=0}^{|E'_{(i,j)}|} X_{0_k}^{\varepsilon_{(i,j)_k}} = X_{0_0}^{r_{(i,j)} \prod_{m \in E'_{(i,j)}}(x'+m)}$$

for randomly selected $r, \{r_i, r_{(i,j)}\} \in \mathbb{Z}_p^*$. The non-intersected portion $\mathcal{G} \setminus I$ is then represented by line (42). Similar to the `possession` proof, clause (44) proves the correctness of randomized credential sig' where $\omega = r, \rho = sr, \tau = ty, \gamma = y$ for randomly selected $y \in \mathbb{Z}_p^*$. The clauses (44), (45) and (46) show that $I \subseteq \mathcal{G}'$ and $|I| = \ell$ where clause (44) corresponds to the correctness for I taken from clause (41) while clause (45) addresses the non-intersecting portion $\mathcal{G}' \setminus I$:

$$W'_i = a_{0_0}^{o_i r_i^{-1} \prod_{m \in V'_i \setminus m_i^*}(x'+m)}, W'_{(i,j)} = a_{0_0}^{o_{(i,j)} r_{(i,j)}^{-1} \prod_{m \in E'_{(i,j)} \setminus m_{(i,j)}^*}(x'+m)}$$
$$X_{0_1}^{\varepsilon_{i_1}} X_{0_0}^{\varepsilon_{i_0}} = X_{0_0}^{r_i(x'+m_i^*)}, X_{0_1}^{\varepsilon_{(i,j)_1}} X_{0_0}^{\varepsilon_{(i,j)_0}} = X_{0_0}^{r_{(i,j)}(x'+m_{(i,j)}^*)}$$

for randomly chosen $m_i^* \in V'_i$ and $m_{(i,j)}^* \in E'_{(i,j)}$. Lastly, we have the following in clause (46):

$$\prod_{i \in V'} C_i^{\alpha_i} \prod_{(i,j) \in E'} C_{(i,j)}^{\alpha_{(i,j)}}$$
$$= \prod_{i \in I} C_i^{r_i} \prod_{i \in V' \setminus I} C_i^{o_i} \prod_{(i,j) \in I} C_{(i,j)}^{r_{(i,j)}} \prod_{(i,j) \in E' \setminus I} C_{(i,j)}^{o_{(i,j)}}$$

such that MoniPoly MSC commitments

$$C_i = a_{0_0}^{\prod_{m\in V_i'}(x'+m)} = \mathsf{MSC.Commit}(mpk, V_i', 1) \text{ and}$$

$$C_{(i,j)} = a_{0_0}^{\prod_{m\in E'_{(i,j)}}(x'+m)} = \mathsf{MSC.Commit}(mpk, E'_{(i,j)}, 1)$$

are computed by the verifier using the base a_{0_0} while $o_i, o_{(i,j)} \in \mathbb{Z}_p^*$ are randomly chosen by the prover.

Our ReCS is capable to support more proofs such as the graph disjointness and graph isolation proposed in the Groß' SRSA graph signature scheme [32], [33]. To achieve this, we introduce two new predicates `vertices` and `edges` that can isolate vertices and edges, respectively, from a graph.

Remark 3. While the the proof predicates presented here cover the same range as the ones proposed by Groß' SRSA graph signature scheme [32], [33], our predicates have different semantics. This is because the MoniPoly set commitment scheme and the underlying monic polynomial structure enable us to implicitly cover aspects that the original graph signature scheme needed to model as explicit protocols. For instance, Groß' `partition` predicate is implicitly realized by our `vertices` predicate. The `disjoint` predicate in Groß' scheme proves the pair-wise differences of vertices and edges in a graph, but this is indirectly done by our `vertices` and `edges` predicates.

`vertices` : proves the correctness of the vertex composition, that is, all cumulated vertex identifiers are subsets of the identifier universe. Intuitively, this is a possession proof plus a proof of cumulative product for all vertex identifiers in the signed graph:

$$PK\Bigg\{\{\varepsilon_{l_0}, \varepsilon_{l_1}\}_{l=1}^{\ell=|V|}, (\forall (i,j)\in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), \rho, \omega, \tau, \gamma) :$$

$$\prod_{l=1}^{\ell} \mathsf{e}\left(W_l, \prod_{k=0}^{1} X_{0_k}^{\varepsilon_{l_k}}\right) \prod_{(i,j)\in E} \mathsf{e}\left(W_{(i,j)}, X_{(i,j)_1}^{\varepsilon_{(i,j)_1}} X_{(i,j)_0}^{\varepsilon_{(i,j)_0}}\right) \cdot \tag{47}$$

$$\mathsf{e}\left(b^{\rho} c^{\omega} v'^{-\tau}, X_{0_0}\right) = \mathsf{e}(v'^{\gamma}, X) \wedge \tag{48}$$

$$\mathsf{e}\left(a_{0_0}^{\Xi_{\mathcal{V}}} \prod_{l=1}^{\ell-1} W_{\mathcal{V}_l}, X_{0_0}\right) = \prod_{l=1}^{\ell} \mathsf{e}\left(W_{\mathcal{V}_{l-1}}, \prod_{k=0}^{1} X_{0_k}^{\varepsilon_{l_k}}\right) \mathsf{e}\left(a_{0_0}^{-1} W_{\mathcal{V}\setminus V}, W_{\mathcal{V}_\ell}\right)\Bigg\} \tag{49}$$

where $W_l = a_{0_0}^{r_l^{-1}\prod_{w\in f_{\mathcal{V}}(i_l)}(x'+w)}, \varepsilon_{l_0} = r_l, \varepsilon_{l_1} = r_l i_l$ for randomly selected $r_l \in \mathbb{Z}_p^*$. The clause (49) is a proof of cumulative product for all vertex identifiers $\{i_l\}$ in $X_{0_1}^{\varepsilon_{l_1}} X_{0_0}^{\varepsilon_{l_0}} = X_{0_0}^{r_l(x'+i_l)}$ with witnesses $W_{V_0} = a_{0_0}, \{W_{V_l} = a_{0_0}^{\prod_{k=1}^{l} r_k(x'+i_k)}\}_{l=1}^{\ell-1}, W_{V_\ell} = X_{0_0}^{\prod_{k=1}^{\ell} r_k(x'+i_k)}$ where $r_l \in \mathbb{Z}_p^*$ are randomly selected by the prover. Clause (49) also contains the set membership proof with witnesses $W_{\mathcal{V}} = C_{\mathcal{V}}^{1/\prod_{k=1}^{\ell} r_k(x'+i_k)}$ where $C_{\mathcal{V}} = a_{0_0}^{\prod_{i\in\mathcal{V}}(x'+i)} = \mathtt{MSC.Commit}(mpk, \mathcal{V}), \{1\}^{|\mathcal{V}|})$ is the MoniPoly MSC commitment on all encoded vertices in $\mathcal{V}$ using the base a_{0_0}. At clause (47), specific bases $X_{(i,j)_k}$ are used for the edges but not required for the vertices because a successful set membership proof implicitly verifies every encoded vertex identifier is unique in the cumulative product. The public inputs $W_{\mathcal{V}_1}, \ldots, W_{\mathcal{V}_\ell}$ are witnesses for the cumulative product of encoded vertex identifiers with $W_{\mathcal{V}_0} = a_{0_0}$.

`edges` : proves the correctness of the edge composition, that is, all cumulated edge identifiers and labels are subsets of the identifiers and labels universes. Intuitively, this is a `possession` proof plus a proof of cumulative product for all edge identifiers in the signed graph:

$$PK\Bigg\{(\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}), \{\{\varepsilon_{k,l_0}, \varepsilon_{k,l_1}, \varepsilon_{k,l_2}\}_{l=1}^{|E_k|}\}_{k=1}^{\ell}, \rho, \omega, \tau, \gamma) :$$

$$\prod_{i\in V} \mathsf{e}\left(W_i, X_{i_1}^{\varepsilon_{i_1}} X_{i_0}^{\varepsilon_{i_0}}\right) \prod_{k=1}^{\ell} \prod_{l=1}^{|E_k|} \mathsf{e}\left(W_{k,l}, X_{0_2}^{\varepsilon_{k,l_2}} X_{0_1}^{\varepsilon_{k,l_1}} X_{0_0}^{\varepsilon_{k,l_0}}\right) \cdot \tag{50}$$

$$\mathsf{e}\left(b^{\rho} c^{\omega} v'^{-\tau}, X_{0_0}\right) = \mathsf{e}(v'^{\gamma}, X) \wedge \tag{51}$$

$$\mathsf{e}\left(\prod_{k=1}^{\ell} C_{\mathcal{V}} \prod_{l=1}^{|E_k|-1} W_{V_{k,l}}, X_{0_0}\right) =$$

$$\prod_{k=1}^{\ell} \prod_{l=1}^{|E_k|} \mathsf{e}\left(W_{V_{k,l-1}}, X_{0_2}^{\varepsilon_{k,l_2}} X_{0_1}^{\varepsilon_{k,l_1}} X_{0_0}^{\varepsilon_{k,l_0}}\right) \mathsf{e}\left(a_{0_0}^{-1} W_{\mathcal{V}_k}, W_{V_{k,|E_k|}}\right)\Bigg\} \tag{52}$$

where clauses (50) and (51) answers the `possession` predicate with witnesses

$$W_i = a_{0_0}^{rr_i^{-1}\prod_{m\in f_{\mathcal{V}}(i)}(x'+m)},$$
$$W_{k,l} = a_{0_0}^{rr_{k,l}^{-1}(x'+j_{k,l})\prod_{m\in f_{\mathcal{E}}(i_{k,l},j_{k,l})}(x'+m)}$$

and $\{\{r_{k,l}\}_{l=1}^{|E_k|}\}_{k=1}^{\ell} \in \mathbb{Z}_p^*$ are randomly selected by the prover. All edge identifiers $\{E_1 = \{E_{1,1} = \{i_{1,1}, j_{1,1}\}, \ldots, E_{1,|E_1|}\}, \ldots, E_\ell\} = \bigcup_{(i,j)\in E} i, j$ in the signed graph $\mathcal{G}$ are extracted by $X_{0_2}^{\varepsilon_{k,l_2}} X_{0_1}^{\varepsilon_{k,l_1}} X_{0_0}^{\varepsilon_{k,l_0}} = X_{0_0}^{r_{k,l}(x'+i_{k,l})(x'+j_{k,l})}$. Every vertex identifier i in each set E_k is unique such that $(E_k \setminus \{i\}) \cap \{i\} = \emptyset$. At clause (50), specific bases X_{i_k} are used for the vertices but not the edges because the cumulative products proof and set membership proof at clause (52) confirm a node with two vertex identifiers must be an edge. The witnesses are constructed in the similar way as in the vertices predicate such that $\{W_{V_{k,0}} = a_{0_0}, \{W_{V_{k,l}} = a_{0_0}^{\prod_{u=1}^{l} r_{k,u}(x'+i_{k,u})(x'+j_{k,u})}\}_{l=1}^{|E_k|-1}, W_{V_{k,|E_k|}} = X_{0_0}^{\prod_{l=1}^{|E_k|} r_{k,l}(x'+i_{k,l})(x'+j_{k,l})}, W_{\mathcal{V}_k} = C_{\mathcal{V}}^{1/\prod_{l=1}^{|E_k|} r_{k,l}(x'+i_{k,l})(x'+j_{k,l})}\}_{k=1}^{\ell}$.

`disjoint` $_{(\mathcal{G}')}$. When a verifier requests a show proof for the predicate $\mathtt{disjoint}_{(\mathcal{G}')}$, i.e., showing a sub-graph $\mathcal{G}' = (V', E', f_{\mathcal{V}}, f_{\mathcal{E}})$ is not inside the signed graph $\mathcal{G}$ such that $\mathcal{G}' \cap \mathcal{G} = \emptyset$, the prover can make use of MoniPoly `MSC.VerifyDifference` algorithm. As edge identifiers must have appeared in the vertex identifiers, it is sufficient to show only $V' \cap V = \emptyset$:

$$PK\Bigg\{((\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}), \tag{53}$$
$$(\forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), (\forall i \in V' : \alpha_i),$$
$$\rho, \omega, \tau, \gamma) :$$
$$\texttt{vertices} \ \wedge \ \mathsf{e}\left(a_{0_0}, W_{V_\ell}\right) = \mathsf{e}\left(W_V, X_{0_0}^{\prod_{i \in V'}(x'+i)}\right) \mathsf{e}\left(R, X_{0_0}\right) \ \wedge \ R \neq 1_{\mathbb{G}_1} \tag{54}$$
$$\left(\forall i \in V' : \mathsf{e}\left(R, X_{0_0}\right) = \mathsf{e}\left(W_i, X_{0_1} X_{0_0}^i\right) \mathsf{e}\left(a_{0_0}^{\alpha_i}, X_{0_0}\right) \ \wedge \ a_{0_0}^{\alpha_i} \neq 1_{\mathbb{G}_1}\right) \tag{55}$$

where $X_{0_0}^{\prod_{i \in V'}(x'+i)}$ in clause (54) is computed by the verifier. The `vertices` proof provides the cumulative product $W_{V_\ell} = X_{0_0}^{\prod_{k=1}^{\ell} r_k (x'+i_k)}$ for the set membership proof where $W_V = a_{0_0}^{q(x') \prod_{k=1}^{\ell} r_k}, R = a_{0_0}^{r(x') \prod_{k=1}^{\ell} r_k}$ are the witnesses. The witnesses $\{W_i = a_{0_0}^{q_i(x') \prod_{k=1}^{\ell} r_k}\}$ and secret remainders $\{\alpha_i = r_i(x') \prod_{k=1}^{\ell} r_k\}$ at clause (55) are computed through MoniPoly `MSC.OpenDifference` where every remainder $r_i(x')$ is a constant.

`isolated` $_{(i,j)}$. This predicate allows a prover to prove the disjointness of two vertex identifiers $\{i^*, j^*\}$ in the signed graph $\mathcal{G} = (V, E, f_{\mathcal{V}}, f_{\mathcal{E}})$. Specifically, a bi-partition variant of edges is executed followed by a `disjoint` to show that i^* and j^* are from the edge sets E_{i^*} and E_{j^*}, respectively, such that $E = \{E_{i^*} \cup E_{j^*}\}$ and $(E_{j^*} \cap E_{i^*}) = \emptyset$:

$$PK\Bigg\{(\{\{\varepsilon_{i,k,l_0}, \varepsilon_{i,k,l_1}, \varepsilon_{i,k,l_2}\}_{l=1}^{|E_{i^*,k}|}\}_{k=1}^{\ell_{i^*}}, \{\{\varepsilon_{j,k,l_0}, \varepsilon_{j,k,l_1}, \varepsilon_{j,k,l_2}\}_{l=1}^{|E_{j^*,k}|}\}_{k=1}^{\ell}$$
$$(\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}), (\forall \bar{j} \in E_{j^*} : \alpha_{\bar{j}_0}, \alpha_{\bar{j}_1}, \beta_{\bar{j}}), \rho, \omega, \tau, \gamma) :$$
$$\prod_{k=1}^{\ell_{i^*}} \prod_{l=1}^{|E_{i^*,k}|} \mathsf{e}\left(W_{i,k,l}, X_{0_2}^{\varepsilon_{i,k,l_2}} X_{0_1}^{\varepsilon_{i,k,l_1}} X_{0_0}^{\varepsilon_{i,k,l_0}}\right) \cdot \tag{56}$$
$$\prod_{k=1}^{\ell_{j^*}} \prod_{l=1}^{|E_{j^*,k}|} \mathsf{e}\left(W_{j,k,l}, X_{0_2}^{\varepsilon_{j,k,l_2}} X_{0_1}^{\varepsilon_{j,k,l_1}} X_{0_0}^{\varepsilon_{j,k,l_0}}\right) \cdot \tag{57}$$
$$\prod_{i \in V} \mathsf{e}\left(W_i, X_{i_1}^{\varepsilon_{i_1}} X_{i_0}^{\varepsilon_{i_0}}\right) \mathsf{e}\left(b^\rho c^\omega v'^{-\tau}, X_{0_0}\right) = \mathsf{e}(v'^\gamma, X) \ \wedge \tag{58}$$
$$\mathsf{e}\left(a_{0_0}, X_{0_2}^{\varepsilon_{i^*,k,l_2}} X_{0_1}^{\varepsilon_{i^*,k,l_1}} X_{0_0}^{\varepsilon_{i^*,k,l_0}}\right) = \mathsf{e}(W_{i^*}, X_{0_1} X_{0_0}^{i^*}) \ \wedge \tag{59}$$
$$\mathsf{e}\left(a_{0_0}, X_{0_2}^{\varepsilon_{j^*,k,l_2}} X_{0_1}^{\varepsilon_{j^*,k,l_1}} X_{0_0}^{\varepsilon_{j^*,k,l_0}}\right) = \mathsf{e}(W_{j^*}, X_{0_1} X_{0_0}^{j^*}) \ \wedge \tag{60}$$
$$\mathsf{e}\left(C_{\mathcal{V}}^{\ell_{i^*}} \prod_{k=1}^{\ell_{i^*}} \prod_{l=1}^{|E_{i^*,k}|-1} W_{V_{i^*,k,l}}, X_{0_0}\right) =$$
$$\prod_{k=1}^{\ell_{i^*}} \prod_{l=1}^{|E_{i^*,k}|} \mathsf{e}\left(W_{V_{i^*,k,l-1}}, X_{0_2}^{\varepsilon_{i,k,l_2}} X_{0_1}^{\varepsilon_{i,k,l_1}} X_{0_0}^{\varepsilon_{i,k,l_0}}\right) \mathsf{e}\left(a_{0_0}^{-1} W_{\mathcal{V}_{i^*,k}}, W_{V_{k,|E_{i^*,k}|}}\right. \tag{61}$$
$$\mathsf{e}\left(\prod_{k=1}^{\ell_{i^*}} W_{E_{i^*,k}}, X_{0_0}\right) = \prod_{k=1}^{\ell_{i^*}} \mathsf{e}\left(W_{E_{i^*,k-1}}, W_{V_{k,|E_{i^*,k}|}}\right) \ \wedge \tag{62}$$
$$\mathsf{e}\left(C_{\mathcal{V}}^{\ell_{j^*}} \prod_{k=1}^{\ell_{j^*}} \prod_{l=1}^{|E_{j^*,k}|-1} W_{V_{j^*,k,l}}, X_{0_0}\right) =$$
$$\prod_{k=1}^{\ell_{j^*}} \prod_{l=1}^{|E_{j^*,k}|} \mathsf{e}\left(W_{V_{j^*,k,l-1}}, X_{0_2}^{\varepsilon_{j,k,l_2}} X_{0_1}^{\varepsilon_{j,k,l_1}} X_{0_0}^{\varepsilon_{j,k,l_0}}\right) \mathsf{e}\left(a_{0_0}^{-1} W_{\mathcal{V}_{j^*,k}}, W_{V_{k,|E_{j^*,}}}\right. \tag{63}$$
$$\mathsf{e}\left(\prod_{k=1}^{\ell_{j^*}} W_{E_{j^*,k}}, X_{0_0}\right) = \prod_{k=1}^{\ell_{j^*}} \mathsf{e}\left(W_{E_{j^*,k-1}}, W_{V_{k,|E_{j^*,k}|}}\right) \ \wedge \tag{64}$$
$$\mathsf{e}(W_{E_{j^*,\ell_{j^*}}}, X_{0_0}) \mathsf{e}\left(\prod_{l=0}^{|E_{j^*}|-1} W_{V_l}, X_{0_0}\right) =$$
$$\prod_{l=0}^{|E_{j^*}|} \mathsf{e}\left(W_{V_{l-1}}, X_{0_1}^{\alpha_{\bar{j}_1}} X_{0_0}^{\alpha_{\bar{j}_0}}\right) \mathsf{e}\left(a_{0_0}^{-1} W_{\mathcal{V}}, W_{V_{|E_{j^*}|}}\right) \ \wedge \tag{65}$$
$$\mathsf{e}\left(W_{E_{i^*,\ell_{i^*}}}, X_{0_0}\right) = \mathsf{e}\left(W_{E_{j^*,\ell_{j^*}}}, W\right) \mathsf{e}\left(R, X_{0_0}\right) \ \wedge \ R \neq 1_{\mathbb{G}_1} \ \wedge \tag{66}$$
$$\left(\forall \bar{j} \in E_{j^*} : \mathsf{e}\left(R, X_{0_0}\right) = \mathsf{e}\left(W_{\bar{j}}, X_{0_1}^{\alpha_{\bar{j}_1}} X_{0_0}^{\alpha_{\bar{j}_0}}\right) \mathsf{e}\left(a_{0_0}^{\beta_{\bar{j}}}, X_{0_0}\right) \ \wedge \ a_{0_0}^{\beta_{\bar{j}}} \neq 1_{\mathbb{G}}\right. \tag{67}$$

where clauses (56) to (58) are the `possession` proof where

the first two clauses represent the graph bi-partition such that

$$W_{i,k,l} = a_{(i,j)_0}^{rr_{(i,j)}^{-1} \prod_{w \in f_{\mathcal{E}}(i,j)}(x'+w)},$$
$$X_{0_2}^{\varepsilon_{i,k,l_2}} X_{0_1}^{\varepsilon_{i,k,l_1}} X_{0_0}^{\varepsilon_{i,k,l_0}} = X_{0_0}^{r_{(i,j)}(x'+i)(x'+j)}$$

for an edge $(i,j) \in E_{i^*,k} \subseteq E_{i^*}$ with $E_{i^*,k} = \{(i_l, j_l)\}_{l=1}^{|E_{i^*,k}|}$ and $E_{i^*} = \{E_{i^*,k}\}_{k=1}^{|E_{i^*}|}$. The values $W_{j,k,l}$ and $X_{0_2}^{\varepsilon_{j,k,l_2}} X_{0_1}^{\varepsilon_{j,k,l_1}} X_{0_0}^{\varepsilon_{j,k,l_0}}$ are constructed analogously. Next, the clauses (59) and (60) show that the identifiers i^* and j^* are from two edges $E_{i^*,k}, E_{j^*,k}$, respectively. Subsequently, these two edges are shown to fall in the bi-partition $E_{i^*,k} \in E_{i^*}$ and $E_{j^*,k} \in E_{j^*}$, respectively, by the proof of cumulative products for E_{i^*} (resp. E_{j^*}) at clauses (61) and (62) (resp. clauses (63) and (64)). Particularly, clause (61) shows the correctness of cumulated product for all encoded edge identifiers $(i,j) \in E_{i^*,k}$ while clause (62) show the correctness of cumulated product for all sets $E_{i^*,k} \in E_{i^*}$. The same information with respect to E_{j^*} is delivered by clauses (63) and (64). Sourcing the cumulated product $W_{E_{j^*,\ell_{j^*}}}$ for encoded edge identifiers from clause (64), clause (65) extracts the edge identifiers as vertex identifiers $\bar{j} \in E_{j^*}$ in $X_{0_1}^{\alpha_{\bar{j}_1}} X_{0_0}^{\alpha_{\bar{j}_0}} = X_{0_0}^{r_{\bar{j}}(x'+\bar{j})}$ to be used as the divisor at clause (67) such that $\prod_{\bar{j} \in E_{j^*}} r_{\bar{j}} = \prod_{l=1}^{\ell_{j^*}} \prod_{k=1}^{|E_{j^*,k}|} r_{(i,j)}$. In the case of $|E_{j^*}| > |E_{i^*}|$, the edge identifiers which acts as the divisor for the disjoint proof in equations (65) to (67) are sourced from $W_{E_{i^*,\ell_{j^*}}}$ instead.

While `isolated` hardly find a useful scenario in social graph, it is useful in application with graph data in general, such as topology attestation [33].

In the prior SRSA graph signature scheme by Groß [32], the cover predicate works differently compared to ours. The SRSA $\mathsf{cover}_{(V')}$ checks whether the queried vertex set V' covers all vertices in the committed graph $V \in \mathcal{G}$ such that $V \subseteq V'$. On the contrary, our $\mathsf{cover}_{(\mathcal{G}',\ell)}$ predicate checks whether a queried graph $\mathcal{G}'$ overlaps the committed graph $\mathcal{G}$ such that $|\mathcal{G}' \cap \mathcal{G}| \geq \ell$. This difference occurs in other predicates as well where ours take in more general inputs (i.e., graph and threshold) and therefore are more computing expensive. To achieve a purposeful comparison for protocol complexity and size, we consider the SRSA predicates that take in the same inputs instead of vertices V' only.

For the protocol complexity, we use the number of point multiplications M_1 in $\mathbb{G}_1$ (resp. M_x for $\mathbb{G}_x$) as a benchmarking unit. We set the conversion parameters [8] at 128-bit security level to $1M_2 = 2M_1, 1M_T = 6M_1$, a pairing $1P = 9M_1$ and a modular exponentiation in SRSA $1E = 5M_1$. We illustrate the complexity comparison for the PoK predicates in Table VIII and Figure 3a. In order to ease the presentation, we assume that every vertex $V_i \in V$ and edge $E_i \in E$ has the same number of labels $l = 10$. The number of edges is twice the number of vertices, $|E| = 2|V|$ and the threshold is $\ell = |V|/10$. Figure 3a shows that the computational complexity of the new q-SDH graph signature scheme is consistently lower than that of the SRSA-based graph signature scheme [32]. We note that the complexity comparison only considers pure graph operations, not the cost for zero-knowledge hash-to-prime predicates to make general identifiers usable by the SRSA scheme, a considerable overhead.

We also compare the proofs size of the two schemes in Table IX. We consider 128-bit security level by using the BLS12-461 curve parameters in Section VIII-B for our scheme while setting the RSA modulus N to have $|N| = 3072$. The size of for the responses in our scheme is fixed to the size of group order $|p| = 308$ and that of the SRSA scheme is set to $|N|$ only. We apply a conservative setting for the range proof parameters $(t = 160, l = 80, s = 80)$.

While the relational credential system is based on a new MoniPoly multi-set commitment scheme, it is useful to consider set commitment schemes and their properties as context. We introduce a different set commitment scheme in the MoniPoly framework in Sections P. This variants differ in the approach to blinding compared to the original MoniPoly set commitment scheme [8]. The scheme presented in Section P is most closely related to the multi-set commitment scheme defined in this paper.

N. General Interface

A set commitment scheme SC is a tuple of seven algorithms [8]:

$$\mathsf{SC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{OpenIntersection}, \mathsf{VerifyIntersection}, \mathsf{OpenDifference}, \mathsf{VerifyDifference})$$

1) `Setup` $(1^k, n) \rightarrow (pk, sk)$. A pair of public and secret keys (pk, sk) are generated by a trusted authority based on the security parameter input 1^k. The message domain $\mathcal{D}$ is defined and $n-1$ is the maximum messages allowed. If n is fixed, sk can be discarded.
2) `Commit` $(pk, A) \rightarrow (C)$. On the input of pk and a message set $A \in \mathcal{D}^{n-1}$, select a random opening value $o \in_R \mathcal{D}$, output the commitment C.
3) `Open` $(pk, C, A, o) \rightarrow b$. Return $b = 1$ if C is a valid commitment to A with the opening value o under pk, and return $b = 0$ otherwise.
4) `OpenIntersection` $(pk, C, A, o, (A', l)) \rightarrow (I, W)$ or $\perp$. If $|A' \cap A| \geq l$ holds, return an intersection set $I = A' \cap A$ of length l with the corresponding witness W, and return an error $\perp$ otherwise.
5) `VerifyIntersection` $(pk, C, (I, W), (A', l)) \rightarrow b$. Return $b = 1$ if W is a witness for S being the intersection set of length l for A' and the set committed to in C, and return $b = 0$ otherwise.
6) `OpenDifference` $(pk, C, A, o, (A', \bar{l})) \rightarrow (D, W)$. If $|A' - A| \geq \bar{l}$ holds, return the difference set $D = A' - A$ of length $\bar{l}$ with the corresponding witness W, and return $\perp$ otherwise.
7) `VerifyDifference` $(pk, C, (D, W), (A', \bar{l})) \rightarrow b$. Return $b = 1$ if W is the witness for D being the difference set of length $\bar{l}$ for A' and the set committed to in C, and return $b = 0$ otherwise.

If a set commitment scheme is perfectly hiding and computational binding, we say that the set commitment scheme is secure.

TABLE VII: Complexity for our proofs of knowledge predicate.

Predicate	Point Mult. ($\mathbb{G}_1$)	Point Mult. ($\mathbb{G}_2$)	Mult. ($\mathbb{G}_T$)	Pairing
possession	$(l+1)n + (l+2)m + 15$	$6(n+m)$	$n+m$	$n+m+2$
cover	$(l+1)(n''+n) + (l+2)(m''+m) + l(n'+m') + 2n' + 3m' + 15$	$6(n+m+n''+m'')$	$n+m+n'+m'-2$	$n+m+n''+m''+3$
connected	$(l+1)n + (l+2)m - 2\ell + 15 + \lvert\mathcal{V}\rvert + (\ell^2+\ell)/2$	$6(n+m) + 13\ell - 8$	$n+m+\ell-1$	$n+m+\ell+3$

Note: l: label, n: $|V|$, m: $|E|$, n': $|V'|$, m': $|E'|$, n'': $|V'-V|$, m'': $|E'-E|$, ℓ: threshold

TABLE VIII: PoK protocol complexity comparison for SRSA-based and MoniPoly graph signature schemes.

Predicate	SRSA [32], [33]		Ours (Point Mult. $\mathbb{G}_1$)
	RSA Mod. Exp.	Approx. Point Mult. $\mathbb{G}_1$	
possession	$10(n+m)+58$	$50(n+m)+290$	$(l+28)n + (l+29)m + 33$
cover	$10n + 10m + 14\ell + 68$	$50n + 50m + 70\ell + 345$	$(l+28)n + (l+29)m + (l+8)n' + (l+9)m' + (l+22)n'' + (l+23)m'' + 30$
connected	$10n + 10m + 104\ell + 107$	$50n + 50m + 520\ell + 535$	$(l+28)n + (l+29)m + 17\ell + 61$

Note: l: label, n: $|V|$, m: $|E|$, n': $|V'|$, m': $|E'|$, n'': $|V'-V|$, m'': $|E'-E|$, ℓ: threshold

TABLE IX: PoK protocol size comparison for SRSA-based and MoniPoly graph signature schemes.

Pedicate	SRSA [33], [32]	Ours
possession	$(3(n+m)+12)N + 7t + 2b + 3(l+s)$	$(n+m+3)G_1 + (n+m)G_2 + (2(n+m)+5)p$
cover	$(3(n+m)+12+22n'+24m')N + 7t + 2b + 3(l+s)$	$(n+m+n''+m''+4)G_1 + (n+m+n''+m'')G_2 + (2(n+m+n''+m'')+n'+m'+5)p$
connected	$(3(n+m)-10+44\ell)N + 7t + 2b + 3(l+s)$	$(n+m+\ell+2)G_1 + (n+m+1)G_2 + (2(n+m-\ell)+2(\ell-1)+7)p$

Note: l: label, n: $|V|$, m: $|E|$, n': $|V'|$, m': $|E'|$, n'': $|V'-V|$, m'': $|E'-E|$, ℓ: threshold

Definition 24. *A set commitment scheme is perfectly hiding if every commitment $C = \texttt{Commit}(pk, A)$ is uniformly distributed such that there exists an $o' \neq o$ for all $A' \neq A$ where $\texttt{Open}(pk, C, A', o') = 1$.*

Definition 25. *An adversary $\mathcal{A}$ is said to $(t_{\mathsf{bind}}, \varepsilon_{\mathsf{bind}})$-break the binding security of a set commitment scheme if $\mathcal{A}$ runs in time at most t_{bind} and furthermore:*

$$\Pr[\mathsf{Open}(pk, C, A_1, o_1) = \mathsf{Open}(pk, C, A_2, o_2) = 1] \geq \varepsilon_{\mathsf{bind}}$$

for any two pairs $(A_1, o_1), (A_2, o_2)$ output by $\mathcal{A}$. We say that a set commitment scheme is $(t_{\mathsf{bind}}, \varepsilon_{\mathsf{bind}})$-secure wrt. binding if no adversary $(t_{\mathsf{bind}}, \varepsilon_{\mathsf{bind}})$-breaks the binding security of the set commitment scheme.

O. `MPEncode` Algorithm

Figure 5 quotes the MoniPoly encoding `MPEncode()`.

P. Variant of MoniPoly Set Commitment

It is not hard to see that computing our variant $C = \mathsf{Commit}(pk, \mathcal{G})$ with an opening value o is equivalent to computing an original $C = \texttt{Commit}(pk, \mathcal{G})$ with an opening value $o - x'$. Since finding x' yields an intractable DLOG problem and finding two different opening values that produce the same C breaks the co-SDH assumption, the externally-blinded MoniPoly variant is as secure as the original scheme, as in Theorem 16.

Fig. 5: `MPEncode()`: Algorithm by Tan and Groß [8] to convert an attribute set into polynomial coefficients.

Input: Attribute set $A = \{m_0, \ldots, m_{n-1}\}$ and prime order p.
Output: $L = \{\mathsf{m}_0, \ldots, \mathsf{m}_n\}$.
Post-conditions: $\prod_{i=0}^{n-1}(x' + m_i) = \sum_{i=0}^{n} \mathsf{m}_i x'^i$

$L[|A|+1] \leftarrow 1$
if $|A| = 1$ **then**
  $L[0] \leftarrow A[0]$
  return L
end if
$L[0] \leftarrow A[0] \times A[1] \mod p$
$L[1] \leftarrow A[0] + A[1] \mod p$
for $i \leftarrow 2$ to $|A|$ **do**
  for $j \leftarrow i$ to 0 **do**
    if $j = i$ **then**
      $L[i] \leftarrow L[i-1] + A[i]$
    else if $j = 1$ **then**
      $L[j] \leftarrow L[j] \times A[i] + L[j-1]$
      $L[0] \leftarrow L[0] \times A[i]$
    else
      $L[j] \leftarrow L[j] \times A[i] + L[j-1]$
    end if
  end for
end for
return L

`Setup` (1^κ). Same as that in Section M.

`Commit` (pk, A). Taking as input a message set

$A = \{m_1, \ldots, m_n\} \in \mathbb{Z}_p^*$, select a random opening value $o \in_R \mathbb{Z}_p^*$, output the commitment as $C = a_0^{o \prod_{k=1}^{n}(x'+m_k)}$.

`Open` (pk, C, A, o). Return 1 if $\mathsf{e}(C, X_0) = \mathsf{e}(a_0^{o \prod_{m \in A \setminus m^*}(x'+m)}, X_1 \; X_0^{m^*})$ holds and return 0 otherwise for a randomly selected $m^* \in A$.

`OpenIntersection` $(pk, C, A, o, (A', \ell))$. If $|A' \cap A| \geq \ell$ holds, return an intersection set $I = A' \cap A$ of length ℓ and a witness W such that $W = a_0^{o \prod_{m \in (A-I)}(x'+m)}$. Otherwise, return a null value $\perp$.

`VerifyIntersection` $(pk, C, I, W, (A', \ell))$. Return 1 if

$$\mathsf{e}(C a_0^{\prod_{m \in A'}(x'+m)}, X_0) = \mathsf{e}(W a_0^{\prod_{m \in A' \setminus I}(x'+m)}, X_0^{\prod_{m \in I}(x'+m)})$$

holds and return 0 otherwise.

`OpenDifference` $(pk, C, A, o, (A', \bar{\ell}))$. If $|A' \cap A| \geq \bar{\ell}$ holds, return a difference set $D = A' - A$ of length $\bar{\ell}$ and the witness $\left(W = a_0^{q(x')}, \{\mathsf{r}_k\}_{k=0}^{\bar{\ell}-1}, \left\{W_i = a_0^{q_i(x')}, \mathsf{r}_i\right\}_{i \in D}\right)$. Specifically, let the polynomial divisor be $d(x') = \prod_{m \in D}(x'+m)$, the monic polynomial $f(x') = o \prod_{m \in A}(x'+m)$ in the commitment $C = a_0^{f(x')}$ can be rewritten[3] as $f(x') = d(x')q(x') + r(x')$ while $r(x') = (x'+i)q_i(x') + r_i(x')$ for every $i \in A'$ where $r_i(x') = \mathsf{r}_i$ is a constant.

`VerifyDifference` $(pk, C, D, (W, \{\mathsf{r}_k\}_{k=0}^{\bar{\ell}-1}, \{W_i, \mathsf{r}_i\}_{i \in A'}), (A', \bar{\ell}))$. Return 1 if the following conditions hold:

$$\mathsf{e}\left(C a_0^{-r(x')} a_0^{\prod_{m \in A'}(x'+m)}, X_0\right) = \mathsf{e}\left(W a_0^{\prod_{m \in A' \setminus D}(x'+m)}, X_0^{d(x')}\right) \wedge a_0^{r(x')} \neq 1_{\mathbb{G}_1} \wedge \left(\forall i \in D : \mathsf{e}\left(a_0^{r(x')} a_0^{-\mathsf{r}_i}, X_0\right) = \mathsf{e}\left(W_i, X_1 X_0^i\right) \wedge a_0^{\mathsf{r}_i} \neq 1_{\mathbb{G}_1}\right)$$

and return 0 otherwise.

Remark 4. The set difference algorithm in the original MoniPoly commitment scheme [8] can verify whether $d(x')$ and $r(x')$ have a common monic divisor by using only the coefficients $\{\mathsf{d}_k, \mathsf{r}_k\}$ because its opening value randomizes only the coefficients in $r(x')$. However, in our variant, the opening value also randomizes the secret value x', making it impossible to verify this condition as x' is not known.

Theorem 16. The MoniPoly set commitment scheme above is perfectly hiding and computational binding under the q-SDH assumption.

Proof. As o is randomly selected, a_0^o and $C = (a_0^o)^{\prod_{k=1}^{n}(x'+m_k)}$ have uniform distribution over $\mathbb{G}_1$. For every C and every set $A = \{m_1, \ldots, m_n\} \in \mathbb{Z}_p^n \setminus -x'$, there exists a unique opening value $o \in \mathbb{Z}_p^*$ such that:

$$\mathsf{dlog}_{a_0}(C) = o \prod_{k=1}^{n}(x'+m_k) \mod p$$
$$o = \frac{\mathsf{dlog}_{a_0}(C)}{\prod_{k=1}^{n}(x'+m_k)} \mod p$$

where $x' \in \mathbb{Z}_p^*$ is not known. Therefore, our MoniPoly set commitment variant is perfectly hiding.

Next, given a co-SDH challenge $(g_1, g_1^{x'}, \ldots, g_1^{x'^q}, g_2, g_2^{x'}, \ldots, g_2^{x'^q})$, we show that a challenger $\mathcal{C}$ can find a solution $(g^{\frac{1}{x'+b}}, b)$ by running an adversary $\mathcal{A}$ which can break the computational binding of our MoniPoly set commitment scheme. Let $a_0 = g_1, \ldots, a_q = g_1^{x'^q}, X_0 = g_2, \ldots, X_q = g_2^{x'^q}$, if $\mathcal{A}$ outputs $(A, o) \neq (A^*, o^*)$ such that:

$$a_0^{o \prod_{m \in A}(x'+m)} = C = a_0^{o^* \prod_{m^* \in A^*}(x'+m^*)},$$

we show that a co-SDH solution can be extracted. Without loss of generality, assuming $(o, a \in A) \neq (o^*, b \in A^*)$ and $b \notin A$, $\mathcal{C}$ can extract:

$$a_0^{o(x'+a)\prod_{m \in A \setminus a}(x'+m)} = a_0^{o^*(x'+b)\prod_{m^* \in A^* \setminus b}(x'+m^*)}$$
$$a_0^{\frac{o(x'+a)\prod_{m \in A \setminus a}(x'+m)}{x'+b}} = a_0^{o^* \prod_{m^* \in A^* \setminus b}(x'+m^*)}$$
$$a_0^{\frac{o((x'+b)q(x')+r(x'))}{x'+b}} = a_0^{o^* \prod_{m^* \in A^* \setminus b}(x'+m^*)}$$
$$a_0^{\frac{or(x')}{x'+b}} = \frac{a_0^{o^* \prod_{m^* \in A^* \setminus b}(x'+m^*)}}{a_0^{oq(x')}}$$
$$a_0^{\frac{1}{x'+b}} = \left(\frac{a_0^{o^* \prod_{m^* \in A^* \setminus b}(x'+m^*)}}{a_0^{oq(x')}}\right)^{\frac{1}{or(x')}}$$

where $q(x')$ is the quotient polynomial and $r(x')$ is the remainder. As o is known and $\deg(r(x')) = 0$, $\mathcal{C}$ can compute $\frac{1}{or(x')} \mod p$ and the denominator as $a_0^{oq(x')} = \left(\prod_{k=0}^{\deg(q(x'))} a_k^{\mathsf{w}_k}\right)^o$ where $\{\mathsf{w}_k\}$ are the coefficients for $q(x')$.

When we have $|A \cap A'| = \ell = |A^* \cap A'|$ for a query set A', it is a special case of the pair $(A, o) \neq (A^*, o^*)$ that can fulfil the `OpenIntersection` algorithm. As it must be $|\overline{A \cap A^*}| \geq 1$, we have $\ell \leq \min(|A|, |A^*|)$ with $|A| \neq |A^*|$; or $\ell \leq |A| - 1$ with $|A| = |A^*|$. Also, $|A' \cap \bar{A}| = \bar{\ell} = |A' \cap \bar{A^*}|$ is a special case that can fulfil `OpenDifference`. As it must be $|\overline{A \cap A^*}| \geq 1$, we have $\bar{\ell} \leq \min(|A|, |A^*|)$ when $|A| \neq |A^*|$; or $\bar{\ell} \leq |A|$ when $|A| = |A^*|$. □

[3]Note that $r(x') \neq 0$ and therefore $a_0^{r(x')} \neq 1_{\mathbb{G}_1}$ whenever $d(x')$ cannot divide $f(x')$, i.e., the sets A and D are disjoint.