A \textit{q}-SDH-based Graph Signature Scheme on Full-Domain Messages with Efficient Protocols

Syh-Yuan Tan, Ioannis Sfyrakis, and Thomas Groß

Newcastle University, Newcastle upon Tyne, United Kingdom
\{syh-yuan.tan, ioannis.sfyrakis, thomas.gross\}@newcastle.ac.uk

Abstract. A graph signature scheme is a digital signature scheme that allows a recipient to obtain a signature on a graph and subsequently prove properties thereof in zero-knowledge proofs of knowledge. While known to be expressive enough to encode statements from NP languages, one main use of graph signatures is in topology certification and confidentiality-preserving security assurance. In this paper, we present an efficient and provably secure graph signature scheme in the standard model with tight reduction. Based on the MoniPoly attribute-based credential system, this new graph signature scheme offers zero-knowledge proofs of possession of the signature itself as well as confidentiality-preserving show proofs on logical statements such as the existence of vertices, graph connectivity or isolation.

1 Introduction

A graph signature scheme is digital signature scheme that operates on a message space of graphs $\mathcal{G}$ and offers efficient proof protocols to assert graph properties, such as connectivity or isolation, in Zero-Knowledge Proofs of Knowledge (ZKPoK) while maintaining confidentiality of further information on the the structure or labeling of the graph itself. Given the versatility of graphs as a data structure, graph signatures are an interesting public-key cryptography primitive for a range of applications, including confidentiality-preserving security assurance, that is, the certification and attestation of topologies of systems of computer systems, first realized as part of cloud security and privacy assurance efforts [17]. Further applications on, for example, social network graphs, state machines, or provenance graphs have been proposed.

While ZKPoK on graphs have been instrumental in the first zero-knowledge proof constructions [6, 16] and transitive signature schemes operating on graph edges have been proposed earlier [23, 2], the first graph signature scheme with a framework of efficient proof protocols was proposed by Groß [18, 19]. This first construction was founded on the SRSA-based Camenisch-Lysyanskaya (CL) signature scheme [11], provably secure in the standard model. The CL signature can be extended to sign multiple messages without increasing the signature size and has been the core engine for attribute-based anonymous credential (ABC) systems, such as IBM’s Identity Mixer [20]. Camenisch and Groß (CG) established a prime encoding [9, 10] to enhance the efficiency of the CL-RSA signature scheme
by rendering the messages as prime exponents. Exploiting the coprimality and non-divisibility of the prime numbers, they showed that the encoded signature can be used to construct a highly efficient ABC system to provide efficient zero-knowledge proofs for AND, OR, and NOT/NAND logical statements.

Groß [18] extended the CG-ABC system into a graph signature scheme which can sign the vertices and edges of a labeled graph in the prime encoded form, while rendering the graph’s algebraic structure accessible to subsequent ZKPoK. Apart from providing zero-knowledge proofs on a grammar of graph predicates [19], the graph signature also supports joint graph issuing protocol that allows the issuer to combine a graph of its choice with recipient’s hidden graph during the signing protocol. This allows a recipient to source different graphs from multiple signers, a feature enabling confidentiality-preserving bootstrapping of signatures on complex graphs. Furthermore, Groß [18] proved the graph signature scheme’s capability to encode and efficiently prove graph 3-colorability implying expressivity in terms of arbitrary statements from NP-languages. Due to these useful features, the graph signature was adopted in the realization of topology certification and infrastructure auditing applications [17]. While the scheme offers useful features, Groß’ original construction of the graph signature scheme retains shortcomings that have yet to be addressed: (i) the message space is restricted to prime numbers, (ii) the encoding needs to be pre-certified, entailing large public keys, (iii) the scheme largely relies on properties inherited from the underlying CL- and CG- schemes [11, 10] and lacks a rigorous security model of the overall proof system. In terms of research gap, the existing SRSA graph signature construction suffers from complex show proofs, where the underlying quadratic-residue special RSA group setup leads to considerable computational and message complexity.

Our Contribution. We propose the first graph signature scheme with efficient protocols based on an elliptic-curve group setup. To that end, we establish a new variant of MoniPoly set commitment scheme [28, 29] that achieves the same efficiency and security as that of the original scheme, while being conducive to encoding graph data structures and proving of their well-formedness. Based on MoniPoly ABC system, we then construct a new graph signature scheme which is provably secure in the standard model with a tight security reduction under the $q$-SDH assumption. In contrast to Groß’ SRSA-based construction [18, 19], our novel graph signature scheme (i) enables graph encoding on arbitrary strings, not just prime numbers, (ii) supports the same expressive range of predicates with shorter proofs, (iii) features rigorous security analysis with respect to general impersonation resilience and unlinkability security requirements.

Organization. §2 introduces the related work to our solution. After listing the preliminaries in §3, we explain in §4 the MoniPoly graph encoding, the SDH-counterpart for Groß’ SRSA-based prime graph encoding. We present the proposed graph signature scheme in §6 and an efficient zero-knowledge proof system for set and graph predicates in §7. §9 evaluates the computational complexity,
compares asymptotically with the original graph signature scheme by Groß, and
details the performance experiments on our implementation.

2 Related Works

Recently, Tan and Groß proposed an efficient ABC system with expressive show
proofs, called MoniPoly [28, 29]. By design, MoniPoly ABC bears a number of
similarities to the Camenisch-Groß-ABC [9, 10]. For instance, MoniPoly ABC is
built on CL-SDH signature scheme [12, 1, 26], the pairing counterpart of CL-RSA
signature scheme [11]. Moreover, MoniPoly ABC features an encoding function
that converts attributes into non-dividable values in \( \mathbb{Z}_p \).

By extending the MoniPoly ABC system’s attribute space to an appropriate
graph encoding, it can yield a graph signature scheme, in principle. The main
technical difficulty in achieving this is to establish the graph data structure in
the credential. Groß’ graph signature scheme [18] achieves this ability by creating
an unambiguous encoding for a graph data set. The graph prime encoding
then yields the capability to produce a graph well-formedness proof to assure the
signed hidden graph is correctly encoded. Unfortunately, the original MoniPoly
ABC cannot achieve the graph well-formedness in the same way, because the
opening value and the graph data set share the same domain. For instance, a
dishonest user can cheat by encoding two vertices \( i, j \) and their edge \( (i, j) \) as:

\[
C = a_i^{(x'+o_i)(x'+i)} a_j^{(x'+o_j)(x'+i)} a_{(i,j)}^{(x'+o_{(i,j)})(x'+i)(x'+j)} \mod p
\]

in a MoniPoly commitment where the opening values \( o_i, o_j, o_{(i,j)} \) are chosen
from the vertex universe. A naive solution might ask the user to compute a NOT
proof with respect to the vertex identifier and labels universes for each opening
value during every show proofs. This naive approach, however, would create a
forbidding overhead and yield an impractical graph signature scheme.

Another related research area is the authenticated data structure [24, 13, 30]
(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 [31, 22, 32]. For instance, the graph
database constructed by Mandal et al. [22] supports query for meta-data, similar
to proving the relationships among vertices in a graph signature scheme. However,
the application scenario of databases from ADS require the data owners
to disclose their data to the database server. Thereby, they enable the database
server to answer the clients’ queries on behalf of the data owners.

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. The owner of a graph signature, hence, responds to
a verifier’s queries on his own, using the signature of his committed graph as
secret input.
3 Preliminaries

Definition 1. Discrete Logarithm Assumption (DLOG). An algorithm \( C \) is said to \((t_{\text{dlog}}, \epsilon_{\text{dlog}})\)-break the DLOG assumption if \( C \) runs in time at most \( t_{\text{dlog}} \) and furthermore:

\[
\Pr[x \in \mathbb{Z}_p : C(g, g^x) = x] \geq \epsilon_{\text{dlog}}
\]

for a negligible probability \( \epsilon_{\text{dlog}} \). We say that the DLOG assumption is \((t_{\text{dlog}}, \epsilon_{\text{dlog}})\)-secure if no algorithm \((t_{\text{dlog}}, \epsilon_{\text{dlog}})\)-solves the DLOG problem.

Definition 2. co-Discrete Logarithm Assumption (co-DLOG) \([14]\). An algorithm \( C \) is said to \((t_{\text{codlog}}, \epsilon_{\text{codlog}})\)-break the co-DLOG assumption if \( C \) runs in time at most \( t_{\text{codlog}} \) and furthermore:

\[
\Pr[x \in \mathbb{Z}_p : C(g_1, g_2, x \in G_1, g_2, g_2 \in G_2) = x] \geq \epsilon_{\text{codlog}}
\]

for a negligible probability \( \epsilon_{\text{codlog}} \). We say that the co-DLOG assumption is \((t_{\text{codlog}}, \epsilon_{\text{codlog}})\)-secure if no algorithm \((t_{\text{codlog}}, \epsilon_{\text{codlog}})\)-solves the co-DLOG problem.

Definition 3. \(q\)-Strong Diffie-Hellman Assumption (SDH) \([26]\). An algorithm \( C \) is said to \((t_{\text{sdh}}, \epsilon_{\text{sdh}})\)-break the SDH assumption if \( C \) runs in time at most \( t_{\text{sdh}} \) and furthermore:

\[
\Pr[x \in \mathbb{Z}_p, c \in \mathbb{Z}_p \setminus \{x\} : C(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x) = (g_1^{x^2}, c)] \geq \epsilon_{\text{sdh}}
\]

for a negligible probability \( \epsilon_{\text{sdh}} \). We say that the SDH assumption is \((t_{\text{sdh}}, \epsilon_{\text{sdh}})\)-secure if no algorithm \((t_{\text{sdh}}, \epsilon_{\text{sdh}})\)-solves the SDH problem.

Definition 4. \(q\)-co-Strong Diffie-Hellman Assumption (co-SDH) \([14]\). An algorithm \( C \) is said to \((t_{\text{cosdh}}, \epsilon_{\text{cosdh}})\)-break the co-SDH assumption if \( C \) runs in time at most \( t_{\text{cosdh}} \) and furthermore:

\[
\Pr[x \in \mathbb{Z}_p, c \in \mathbb{Z}_p \setminus \{x\} : C(g_1, g_1^x, \ldots, g_1^{x^q}, g_2, g_2^x, \ldots, g_2^{x^q}) = (g_1^{x^2}, c)] \geq \epsilon_{\text{cosdh}}
\]

for a negligible probability \( \epsilon_{\text{cosdh}} \). We say that the co-SDH assumption is \((t_{\text{cosdh}}, \epsilon_{\text{cosdh}})\)-secure if no algorithm \((t_{\text{cosdh}}, \epsilon_{\text{cosdh}})\)-solves the co-SDH problem.

3.1 Pedersen Commitment

Pedersen commitment scheme \([25]\) is perfectly hiding and computationally binding under the discrete logarithm assumption. The public parameters \( pk_{PC} = (a, b \in G) \) are based on a group \( G \) of order \( p \) in which the discrete logarithm assumption holds. In order to commit to a message \( m \), one computes

\[
C = \text{Commit}(pk_{PC}, m, r) = a^m b^r
\]

where \( r \in \mathbb{Z}_p^* \) is the randomly selected opening value.
3.2 MoniPoly Set Commitment

MoniPoly set commitment scheme \([28, 29]\) is perfectly hiding and computationally binding under the co-SDH assumption. The public parameter is \(pk_{MP} = (\{a_k = a^{x_k}\}_{k=1}^n \in \mathbb{G}_1, \{X_k = g_2^{x_k}\}_{k=1}^n \in \mathbb{G}_2, \text{MP Encode})\) where \(\text{MP Encode} : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p^{n+1}\) converts a set of \(n\) messages into coefficients for a monic polynomial of degree \(n + 1\). In order to commit to a set of messages \(A = \{m_1, \ldots, m_n-1\}\), one computes:

\[
C = \text{Commit}(pk_{MP}, A, o) = a_0^{(x'+o)} \prod_{k=1}^{n-1} (x'+m_k) = \prod_{k=0}^{n-1} a_k^{m_k}
\]

where \(o \in \mathbb{Z}_p^*\) is the randomly selected opening value and \(\{m_j\} = \text{MP Encode}(A \cup \{o\})\). MoniPoly commitment scheme supports zero-knowledge proofs on set operations:

1. intersection: proves the knowledge of an intersection set \(I = A' \cap A\),
2. difference: proves the knowledge of a difference set \(D = A' - A\),

which give rise to the show proofs in the MoniPoly ABC system \([29]\).

3.3 Camenisch-Lysyanskaya SDH Signature Scheme

The CL-SDH signature scheme \([12, 1, 26]\) is closely related to the BBS signature scheme \([8]\). Since the MoniPoly ABC system \([28, 29]\) is the foundation of our proposed graph signature scheme, we recall the CL-SDH signature variant described by Tan and Groß where the signed messages are the input to an encoding function \(\text{MP Encode}\) as described above:

KeyGen\((1^k, 1^n)\). Construct three cyclic groups \(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\) of order \(p\) based on an elliptic curve whose bilinear pairing is \(e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T\). Select random generators \(a, b, c \in \mathbb{G}_1, g_2 \in \mathbb{G}_2\) and two secret values \(x, x' \in \mathbb{Z}_p\). Compute the values \(X = g_2^n, \{a_i = a^{x''}, X_i = g_2^{x''}\}_{0 \leq i \leq n}\) to output the public key \(pk = (e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, b, c, \{a_i, X_i\}_{0 \leq i \leq n}, X)\) and the secret key \(sk = (x, x')\).

Sign\((pk, sk, \{m_1, \ldots, m_n\})\). On input \(m_1, \ldots, m_n\), choose the random values \(s, t \in \mathbb{Z}_p^*\) to compute:

\[
v = \left(\frac{\prod_{i=0}^n (x'+m_i)}{b^s} c\right)^{\frac{1}{x' + t}}
\]

and output the signature as \(\sigma = (t, s, v)\).

Verify\((pk, \sigma, \{m_1, \ldots, m_n\})\). Given \(\sigma = (t, s, v)\), compute \(\{m_i\}_{0 \leq i \leq n} = \text{MP Encode}(\{m_i\}_{1 \leq i \leq n})\). Output 1 if the following holds:

\[
e(v, X) = e \left(\prod_{i=0}^n a_i^{m_i} b^s c v^{-t}, g_2\right)
\]

and output 0 otherwise.
**Theorem 1.** [26, 29] The CL-SDH signature is strongly existential unforgeable against chosen message attack in the standard model if the SDH problem is intractable.

4 MoniPoly Graph Encoding

In this section, we offer a brief overview of the graph prime encoding proposed by Groß [18] before presenting our new encoding, namely, the MoniPoly graph encoding. It is conceptually similar to the former in nature and idea, yet supports graph encoding over $\mathbb{Z}_p^*$ instead of over prime numbers.

4.1 Prime Graph Encoding

The prime graph encoding views every vertex and edge as a prime exponent. Camenisch and Groß [10] showed that a prime encoding in general can significantly speed up the show proofs by exploiting the co-primality and divisibility among messages. The main parameters for prime graph encoding are as follows:

- $\mathcal{V}$: Vertex universe
- $\mathcal{E} \subseteq (\mathcal{V} \times \mathcal{V})$: Edge universe
- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$: Graph
- $\Xi_\mathcal{V}$: Vertex identifier universe
- $\Xi_\mathcal{L}$: Labels universe
- $f_\mathcal{V} : \mathcal{V} \rightarrow \mathcal{P}(\Xi_\mathcal{L})$: Labels of a given vertex
- $f_\mathcal{E} : \mathcal{E} \rightarrow \mathcal{P}(\Xi_\mathcal{L})$: Labels of a given edge
- $\chi_\mathcal{V}$: product of all vertex identifiers $\prod_{i \in \Xi_\mathcal{V}} i$
- $\chi_\mathcal{L}$: product of all labels $\prod_{i \in \Xi_\mathcal{L}} i$

where vertex identifiers and label are disjoint:

$$\Xi_\mathcal{V} \cap \Xi_\mathcal{L} = \emptyset \iff \gcd(\chi_\mathcal{V}, \chi_\mathcal{L}) = 1$$

following the fundamental theorem of arithmetic.

In the following explanation, we focus on an unlabeled graph for clarity without loss of generality. In order to encode an unlabeled graph using the prime graph encoding, every vertex $i \in \mathcal{V}$ is mapped to a predefined prime number $e_i$. Next, let $R_i$ be the base for the $i$-th vertex and $R_{(i,j)}$ be the base for the $(i,j)$-th edge, while $m_i = e_i \prod_{k \in f_\mathcal{V}(i)} e_k$ and $m_{(i,j)} = e_i e_j \prod_{k \in f_\mathcal{E}(i,j)} e_k$ denote the full encoding of vertices and edges, respectively. Assuming the signer knows the discrete logarithms of every base with respect to the public key element $S$, a graph can be represented by its vertex exponents $\bar{e}_i = \text{dlog}_S(R_i) m_i$ and its edge exponents $\bar{e}_{(i,j)} = \text{dlog}_S(R_{(i,j)}) m_{(i,j)}$. 
4.2 Encoding Graphs Into the MoniPoly Set Commitment

In the MoniPoly set commitment scheme [28, 29], the messages are converted into a set of monic polynomial coefficients using a conversion function \( \text{MPEncode} : \mathbb{Z}_n \rightarrow \mathbb{Z}_n^{n+1} \) before being committed. Let us view the exponents in MoniPoly as encoded elements such that:

\[
(x' + i) \mod p \Leftrightarrow e_i \mod \phi(N).
\]

Then, \( \text{MPEncode} \) can be used as an encoding for graphs similar to the prime graph encoding.

Recalling from Section 3.2, let the public parameters be \( \{ \alpha_{0k} = \alpha x^k, X_{0k} = X x^k \}_{k=0}^n \). To encode a graph \( G \) of maximum size \( L \), we generate the following additional public parameters:

\[
\{ \{ \alpha_{ik}, X_{ik} \}_{i=1}^L \}_{k=0}^n
\]

Subsequently, to represent a vertex and an edge, respectively, let \( \{ m_{ik} \} = \text{MPEncode}(i, f_V(i)) \) and \( \{ m_{(i,j)k} \} = \text{MPEncode}(i, j, f_E(i, j)) \), we have

\[
\prod_{k=1}^{n_i} \alpha_{ik}^{m_{ik}} \Leftrightarrow R^{m_{i}} \quad \text{and} \quad \prod_{k=1}^{n_{(i,j)}} \alpha_{(i,j)k}^{m_{(i,j)k}} \Leftrightarrow R^{m_{(i,j)}}
\]

The parameters for prime graph encoding can then be adjusted accordingly to suit the MoniPoly encoding. Moreover, the MoniPoly encoding fulfills the requirement that vertex identifiers and labels be disjoint under the fundamental theorem of algebra. Building upon the conceptualization of Groß’ graph signature scheme [18], we define graph well-formedness in the context of the MoniPoly graph encoding as follows:

**Definition 5 (Well-formedness).** We call a graph encoding well-formed if and only if:

1. The encoding only contains MoniPoly encoding representatives \((x' + i) \in \Xi_V \cup \Xi_L\) in the exponents of the base \(a_i\).
2. A base \(a_i\) contains either exactly one vertex identifier \((x' + i) \in \Xi_V\), pair-wise different from other vertex identifiers and zero or more label representatives \((x' + k) \in \Xi_L\), or;
3. contains exactly two vertex identifiers \((x' + i), (x' + j) \in \Xi_V\) and zero or more label representatives \((x' + k) \in \Xi_L\).

Considering the concepts we have introduced so far, we are not yet in the position the MoniPoly encoding to construct a SDH-based graph signature scheme in a way similar to Groß’ SRSA-based graph signature scheme [18, 19]. This is because when an encoded graph is committed as a MoniPoly commitment naively, it loses its graph well-formedness property due to the shared message and opening space. This is unlike the combination of prime encoding and Pedersen commitment proposed by Groß. In the following section, we show how to overcome this problem.
5 Graph Well-formedness from a MoniPoly Commitment

We propose a variant of MoniPoly set commitment scheme featuring an externalized random blinding which is as secure and as efficient as the original scheme [28, 29]. We show that a further extension of this commitment scheme can commit MoniPoly encoded graphs securely. We continue to prove that the extension supports graph well-formedness if it has binding property.

5.1 An Externally-Blinded MoniPoly Set Commitment Scheme

To illustrate why a naive use of the original MoniPoly set commitment scheme fails desired security properties, we first consider the case in which a graph \( G \) has only one vertex \( V = \{i, f \} \). The element \( a_0 \) then holds the MoniPoly encoded graph as exponents. It can be viewed as an insecure MoniPoly commitment \( C' = \prod_{k=0}^{n} a_{m_k} \). Specifically, the opening value for \( C' \) is one of the committed graph elements: there exist finitely many of those and they are not random. Therefore, \( C' \) is not perfect hiding, though the computational binding property remains intact under the SDH assumption. Adding the opening value \( o \) breaks the graph well-formedness as \( (x' + o) \) is not part of the graph encoding.

This problem can be resolved by selecting a random blinding \( o \in \mathbb{Z}_p^* \) to compute the commitment as \( C = C' o \). One can thereby consider \( C = C' o = (x' + o') \mod p \) contains unknown opening value \( o' \in \mathbb{Z}_p^* \). Therefore, computing \( C = \text{Commit}(pk, G, o - x') \) is equivalent to computing \( C = \text{Commit}(pk, G, o - x') \). 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.

We describe our proposed MoniPoly set commitment variant as follows:

**Setup(1^k).** Same as that in Section 3.2.

**Commit(pk, A, o).** Taking as input a message set \( A = \{m_1, \ldots, m_n\} \in \mathbb{Z}_p^* \) and the random opening value \( o \in \mathbb{Z}_p^* \), output the commitment as

\[
C = \left( \prod_{k=1}^{n} (x_{m_k}) \right)^o = \left( \prod_{k=0}^{n} a_{m_k} \right)^o
\]

where \( \{m_k\} = \text{MPEncode}(A) \).

**Open(pk, C, A, o).** Return 1 if \( C = \prod_{k=0}^{n} (a_{m_k})^o \) holds where \( \{m_k\} = \text{MPEncode}(A) \) and return 0 otherwise.
OpenIntersection$(pk, C, A, o, (A', l))$. If $|A' \cap A| \geq l$ holds, return an intersection set $I = A' \cap A$ of length $l$ and a witness such that:

$$W = \left\{ \prod_{m_k \in (A - I)} (x' + m_k) \right\}^o$$

$$= \left( \prod_{k=0}^n a_j \right)^o$$

where $\{w_k\} = \text{MP Encode}(A - I)$. Otherwise, return a null value $\bot$. The correctness can be verified as follows:

$$C = \prod_{m_k \in I} (x' + m_k)$$

$$= \left( \prod_{m_k \in (A - I)} (x' + m_k) \right)^o \prod_{m_k \in I} (x' + m_k)$$

$$= \left( \prod_{m_k \in A} (x' + m_k) \right)^o.$$ 

VerifyIntersection$(pk, C, I, W, (A', l))$. Return 1 if

$$e \left( C \prod_{k=0}^{\lvert A' \rvert} a_k^{m_{1,k}, \cdot}, X_0 \right) = e \left( W \prod_{k=0}^{\lvert A' \rvert - l} a_k^{m_{2,k}, \cdot}, \prod_{k=0}^{l} X_k^{1_k} \right)$$
holds and return 0 otherwise, where \( \{i_k\} = \text{MP Encode}(I) \), \( \{m_{1,k}\} = \text{MP Encode}(A') \) and \( \{m_{2,k}\} = \text{MP Encode}(A' - I) \). The correctness is shown as follows:

\[
\begin{align*}
\text{OpenDifference}(pk, C, A, o, (A', \bar{l})). & \quad \text{If} \ |A' \cap A| \geq \bar{l} \text{ holds, return a difference set} \\
D &= A' - A \text{ of length } \bar{l} \text{ and the witness} \\
&\quad \left(W = \prod_{k=0}^{n-\bar{l}} a_k^{w_k}, \{r_k\}_{k=0}^{\bar{l}-1}\right). \quad \text{The values} \\
&\quad \left(\{w_k\}, \{r_k\}\right) = \text{MP Encode}(A)/\text{MP Encode}(D) \text{ are computed using expanded synthetic division such that} \\
&\quad \{w_k\} \text{ are the coefficients of quotient } q(x') \text{ and } \{r_k\} \text{ are the coefficients of remainder } r(x'). \quad \text{Specifically, let the polynomial divisor be} \\
d(x') = \sum_k d_k x'^k \text{ where } \{d_k\} = \text{MP Encode}(D), \text{ the monic polynomial } f(x') \text{ in} \\
\text{the commitment } C = a_0^f(x') \text{ can be rewritten as} \\
f(x') = d(x')q(x') + r(x'). \text{ Note that} \\
\prod_{k=0}^{\bar{l}-1} a_k^{r_k} \neq 1_{G_1} \text{ whenever } d(x') \text{ cannot divide } f(x'), \text{i.e., the sets } A \text{ and } D \text{ are}
\end{align*}
\]
disjoint. The correctness can be verified from the following:

\[ C = \left( \prod_{x^t \in k} (x^t + m_k) \right)^{a_0} \]
\[ = \left( a_0^{q(x')d(x') + r(x')} \right)^{a_0} \]
\[ = \left( \prod_{k=0}^{\omega \cdot 0} a_k \right)^{d(x')} \left( \prod_{k=0}^{\omega \cdot 0} a_k \right)^{a_0} \]
\[ = W^{d(x')} \left( \prod_{k=0}^{\omega \cdot 0} a_k \right)^{a_0} \]

VerifyDifference(pk, C, D, (W, \{t_k\}_{k=0}^{\omega \cdot 0}, (A', \bar{l}))). Return 1, if the following holds:

\[ e \left( C \left( \prod_{k=0}^{\omega \cdot 0} a_k \right) \prod_{k=0}^{\omega \cdot 0} a_k^{m_{1,k}}, X_0 \right) = e \left( W \prod_{k=0}^{\omega \cdot 0} a_k^{d_{2,k}}, \prod_{k=0}^{\omega \cdot 0} X_{0} \right) \]
\[ \prod_{k=0}^{\omega \cdot 0} a_k^{b_k} \neq 1_{G_1} \]

and return 0 otherwise, where \( \{d_k\} = \text{MP Encode}(D), \{m_{1,k}\} = \text{MP Encode}(A') \) and \( \{m_{2,k}\} = \text{MP Encode}(A' - D) \). The correctness is as follows:

\[ e \left( C \left( \prod_{k=0}^{\omega \cdot 0} a_k \right) \prod_{k=0}^{\omega \cdot 0} a_k^{m_{1,k}}, X_0 \right) \]
\[ = e \left( C \left( \prod_{k=0}^{\omega \cdot 0} a_k \right) ^{a_0}, X_0 \right) e \left( \prod_{k=0}^{\omega \cdot 0} a_k^{m_{1,k}}, X_0 \right) \]
\[ = e \left( a_0^{d(x')q(x') + or(x') - or(x')}, X_0 \right) e \left( \prod_{m_{k \in A'}} (x^t + m_k), X_0 \right) \]
\[ = e \left( a_0^{d(x')q(x')}, X_0 \right) e \left( \prod_{m_{k \in (A' - D)}} (x^t + m_k), X_0 \right) \]
\[ = e \left( \prod_{k=0}^{\omega \cdot 0} a_k^{m_{1,k}}, X_0 \right) e \left( \prod_{k=0}^{\omega \cdot 0} a_k^{d(x')}, X_0 \right) \]
\[ = e \left( W \prod_{k=0}^{\omega \cdot 0} a_k^{m_{2,k}}, \prod_{k=0}^{\omega \cdot 0} X_{0} \right) \]

For the completeness of security analysis, we support the security for the proposed scheme with Theorem 2.
Theorem 2. The externally-blinded MoniPoly set commitment scheme is perfectly hiding and computational binding under the $q$-SDH assumption.

Proof. (Sketch.) The externally-blinded commitment scheme described here is similar to the original MoniPoly commitment scheme [29] such that one can prove its security by adapting the latter’s security proofs. Specifically, we can view the externally-blinded variant as a MoniPoly commitment with a randomized base in the proof such that:

$$C = \left( \prod_{k=0}^{n} a_k^{m_k} \right)^o = \prod_{k=0}^{n} (a_k^o)^{m_k}.$$  

As one can easily see, the new scheme inherits the perfectly hiding property from original MoniPoly scheme. On the other hand, if an adversary outputs $(A, o') \neq (A^*, o'')$ such that

1. $\text{Open}(pk, \text{Commit}(pk, A, o'), A, o') = \text{Open}(pk, \text{Commit}(pk, A^*, o''), A^*, o'')$,
2. $\text{OpenIntersection}(pk, \text{Commit}(pk, A, o'), A, o', (A', l)) = \text{OpenIntersection}(pk, \text{Commit}(pk, A^*, o''), A^*, o'', (A', l))$,
3. $\text{OpenDifference}(pk, \text{Commit}(pk, A, o'), A, o', (A', \bar{l})) = \text{OpenDifference}(pk, \text{Commit}(pk, A^*, o''), A^*, o'', (A', \bar{l}))$,

a $q$-SDH solution can be extracted from the fact that two different sets $(A, o') \neq (A^*, o'')$ yield the same commitment $C$. The extraction is the same as in the original MoniPoly scheme but uses the randomized bases $a_k^o, a_k^{o''}$. Note that we can always derandomize the bases to gain their original form as in the given SDH instance because $o'$ and $o''$ are known. \qed

5.2 An Extended Externally-Blinded MoniPoly Set Commitment

The proposed variant of MoniPoly set commitment can be used to commit a Monipoly encoded graph, if we add parameters $\{\{a_{ik}, X_i\}_{k=0}^{n} \}_{i=1}^{L}$ specific for the MoniPoly encoding to the commitment parameters. While the commitment opening algorithms can be amended accordingly, the security proof requires a considerate modification. In particular, while the perfectly hiding property follows trivially, the computational binding property is now based on the hardness of the co-DLOG problem.

Theorem 3. The extended externally-blinded MoniPoly set commitment is binding if the co-DLOG problem is hard.

Proof. Given a co-DLOG instance $(g_1, h_1 = g_1^x \in G_1, g_2, h_2 = g_2^x \in G_2)$, we construct a challenger $C$ that runs the adversary $A$ of extended MoniPoly set commitment scheme as a sub-routine to find the solution $x$. $C$ sets $\{a_{ik} = g_1^{b_k x^k}, X_{ik} = g_2^{b_k x^k}\}_{k=0}^{n}$ and $\{a_{ik} = h_1^{b_i x^k}, X_{ik} = h_2^{b_i x^k}\}_{k=0}^{n}$ for randomly chosen $b_i, x' \in \mathbb{Z}_p^*$. $C$ publishes $\{\{a_{ik}, X_{ik}\}_{k=0}^{n} \}_{i=1}^{L}$ as the public parameters.
Without loss of generality, we assume a graph is MoniPoly encoded using the bases $a_{i_k}$ in the sequence $i = 1, 2, \ldots, L$. If an adversary can output an extended MoniPoly set commitment $C$ for two different graph data sets $(\mathcal{G}, \mathcal{G}^*)$ such that $|\mathcal{G} \cap \mathcal{G}^*| \geq 2$:

$$
\mathcal{G} = \{V \cup E\} = \{(i, f_V(i), o_i') \in V, (i, j, f_E(i, j), o_{(i,j)}') \in E\},
$$
$$
\mathcal{G}^* = \{V^* \cup E^*\} = \{(i^*, f_V(i^*), o_i'') \in V^*, (i^*, j^*, f_E(i^*, j^*), o_{(i,j)}'') \in E^*\},
$$

where $\{o_i', o_{(i,j)}', o_i'', o_{(i,j)}''\}$ are the opening values for vertices and edges, respectively, a co-DLOG solution can be extracted. This follows from:

$$
C = \prod_{i \in V} \left( a_{i_0} \prod_{w \in f_V(i)} (x^i + w) \right)^{o_i'} \prod_{(i,j) \in E} \left( a_{(i,j)_0} \prod_{w \in f_E(i,j)} (x^{i,j} + w) \right)^{o_{(i,j)}'}
$$
$$
= \prod_{i^* \in V^*} \left( a_{i_0} \prod_{w^* \in f_V(i^*)} (x^{i^*} + w^*) \right)^{o_i''} \prod_{(i^*,j^*) \in E^*} \left( a_{(i,j)_0} \prod_{w^* \in f_E(i^*,j^*)} (x^{i^*,j^*} + w^*) \right)^{o_{(i,j)}}
$$

giving the following equations:

$$
g_1 u = g_1^u h_1^v
$$
$$
g_1 u^* = g_1^u h_1^v^*
$$

Therein, the Challenger $C$ can compute:

$$
x = \frac{u - u^*}{v^* - v} \mod p
$$

to solve the co-DLOG problem. The argument on the OpenIntersection and Open-Difference algorithms is similar to that in the proof of Theorem 2.

Thereafter, whenever we mention a MoniPoly commitment, we mean the extended externally-blinded MoniPoly set commitment scheme.

### 5.3 Graph Composition

The MoniPoly commitment $C = \prod_{i \in V} C_i \prod_{(i,j) \in E} C_{(i,j)}$ on a graph $\mathcal{G}$ is the product of vertex commitments $C_i$ and edge commitments $C_{(i,j)}$, where:

$$
C_i = \left( \prod_{k=0}^{n_i} a_{i_k}^{m_{i_k}} \right)^{o_i}, C_{(i,j)} = \left( \prod_{k=0}^{n_{(i,j)}} a_{(i,j)_k}^{m_{(i,j)_k}} \right)^{o_{(i,j)}}
$$
To prove the composition of the graph commitment $C$, one can run proof of knowledge protocols for the MoniPoly $\text{Open}$ algorithms on every $C_i \in V$ and $C_{(i,j)} \in E$. These proof of knowledge protocols can be combined into a graph composition statement $\text{graph}(C)$:

$$PK\left\{ (\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}, \forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}) : \right.$$ 

$$e(C, X_{0_i}) = \prod_{i \in V} e \left( W'_i \prod_{k=0}^{1} X^{\varepsilon_k}_{0_k} \right) \prod_{(i,j) \in E} e \left( W'_{(i,j)} \prod_{k=0}^{1} X^{\varepsilon_{(i,j)_k}}_{0_k} \right) \right\}.$$ 

The correctness can be verified from the equation below:

$$e(C, X_{0_i}) = \prod_{i \in V'} e \left( a_{i_0}^{-1} \prod_{w \in f_{V'}(i)} (x'+w), X^{r_{(i,j)}}_{0_0} \right) \prod_{(i,j) \in E'} e \left( a_{(i,j)_0}^{-1} \prod_{w \in f_{E'}(i,j)} (x'+w), X^{r_{(i,j)}(x'+i)}_{0_0} \right).$$

We can replace the exponent $i$ for $X_{0_i}$ with a random attribute $w$ without affecting the randomness in the proof as claimed in Lemma 1. The graph composition proof also appears in the $\text{possession}(\sigma, \mathcal{G})$ statement to prove the validity of a graph signature.

**Lemma 1.** The randomization of $C$ in graph is perfectly hiding.

**Proof.** The MoniPoly opening values $o_i, o_{(i,j)}$ turns the $\mathcal{G}_T$ elements:

$$\prod_{i \in V'} e \left( a_{i_0}^{-1} \prod_{w \in f_{V'}(i)} (x'+w), X^{r_{(i,j)}}_{0_0} \right) \prod_{(i,j) \in E'} e \left( a_{(i,j)_0}^{-1} \prod_{w \in f_{E'}(i,j)} (x'+w), X^{r_{(i,j)}(x'+i)}_{0_0} \right)^{o_{(i,j)}}$$

into a Pedersen set commitment (Lemma 14) which is perfectly hiding. When $C$ consists of one vertex or one edge only, the single $\mathcal{G}_T$ element is a MoniPoly commitment and it is perfectly hiding. \hfill \qed

### 5.4 Bootstrapping of MoniPoly-Encoded Graphs

The $\text{graph}$ statement proves that $C$ can be decomposed into MoniPoly commitments $C_i, C_{(i,j)}$ only. In order to prove that $C$ is a commitment for a correctly encoded graph, we need to further show that the $n$ vertex commitments $C_i \in V$ and $m$ edge commitments $C_{(i,j)} \in E$ is correctly encoded. Firstly, instead of selecting a random exponent for the $X_{0_k}$ elements in $\text{graph}$, we set all encoded labels as exponents for the witness $W'_i, W'_{(i,j)}$ and the encoded identifiers as the exponents for the $X_{0_k}$ elements. Secondly, assuming the universe for identifiers ($\mathcal{X}_V$) and labels ($\mathcal{X}_E$) are publicly known, we need to prove the existence of
every commitment’s identifier and labels in $\Xi$ and $\Xi_L$, respectively. Computing the \( n + m \) MoniPoly commitment for $\Xi_V$ and $\Xi_L$ is required because every commitment $C_i$ (resp. $C_{(i,j)}$) is computed on a different base $a_{i_k}$ (resp. $a_{(i,j)_{k}}$).

This complexity overhead can be avoided by an additional step of bootstrapping [18] that switches the bases \( \{a_i\}_{i \in V}, \{a_{(i,j)}\}_{(i,j) \in E} \) to $a_0$ such that:

\[
C_i = \left( \prod_{k=0}^{n_i} a_{0_{i_k}} \right)^{\alpha_{i}} \quad \text{and} \quad C_{(i,j)} = \left( \prod_{k=0}^{m_{(i,j)}} a_{0_{k}} \right)^{\alpha_{(i,j)}}.
\]

The two MoniPoly commitments for $\Xi_V$ and $\Xi_L$ are now computed only once on the base $a_{0_k}$ and referenced by all set membership proofs. Moreover, since \( \{C_i, C_{(i,j)}\} \) are computed on the same base, we can prove the encoding correctness for all the vertices and edges by using only two set membership proofs plus two cumulative product proofs, respectively. We first explain the simpler bootstrap statement as follows:

\[
PK\{ \forall i \in V : \varepsilon_{i_0}, \varepsilon_i, \forall (i, j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \varepsilon_{(i,j)_2} : \\
\left( W'_i, \prod_{k=0}^{1} X_{0_{i_k}}^{\varepsilon_{i_k}} \right) \prod_{(i,j) \in E} \left( W'_{(i,j)}, \prod_{k=0}^{2} X_{0_{k}}^{\varepsilon_{(i,j)_k}} \right) \wedge \\
\left( W_i, \prod_{k=0}^{1} X_{0_{i_k}}^{\varepsilon_{i_k}} \right) \prod_{(i,j) \in E} \left( W_{(i,j)}, \prod_{k=0}^{2} X_{0_{k}}^{\varepsilon_{(i,j)_k}} \right) \wedge \\
\left( W'_{(i,j)}, X_{0_{i_0}} \right) = \prod_{i \in V} \left( W_i, X_{i_0} \right) \prod_{(i,j) \in E} \left( W_{(i,j)}, X_{(i,j)_0} \right) \}
\]

where $\varepsilon_{i,1} = r_i, \varepsilon_{(i,j),2} = r_{(i,j)}; W'_i = a_{i_0} o_{i}^{-1} \prod_{w \in f_{V(i)}} (x_i + w)$ and $W'_{(i,j)} = a_{(i,j)_{0}} o_{(i,j)}^{-1} \prod_{w \in f_{E((i,j))}} (x_i + w)$ for randomly selected $\alpha_{i}, r_{i}, o_{(i,j)}, r_{(i,j)} \in \mathbb{Z}_p$ and commitments \( \{C_i, C_{(i,j)}, W'_i, W'_{(i,j)}, W_i, W_{(i,j)}\} \) are public inputs.

The first statement is the graph($C$) statement while the second and third statements are the bootstrapping statement bootstrap($C$). The MoniPoly commitments \( \{C_i, C_{(i,j)}, W_i, W_{(i,j)}\} \) that appear in the bootstrapping will be used in vertices and edges which are described in the next sections. Note that the bootstrapping can be compressed to result in a more efficient proof of representation.
as follows:

$$PK\left\{ (\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}, \forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \varepsilon_{(i,j)_2}) : \\
e(C, X_{0_0}) = \prod_{i \in V} e \left( W'_i, \prod_{k=0}^{1} X_{0_k}^{\varepsilon_{i_k}} \right) \prod_{(i,j) \in E} e \left( W'_{(i,j)}, \prod_{k=0}^{2} X_{0_k}^{\varepsilon_{(i,j)_k}} \right) \land \\
e \left( \prod_{i \in V} W'_i C_i \prod_{(i,j) \in E} W'_{(i,j)} C_{(i,j)}, X_{0_0} \right) = \\
\prod_{i \in V} e \left( W'_i, X_{i_0} \prod_{k=0}^{1} X_{0_k}^{\varepsilon_{i_k}} \right) \prod_{(i,j) \in E} e \left( W'_{(i,j)}, X_{(i,j)_0} \prod_{k=0}^{2} X_{0_k}^{\varepsilon_{(i,j)_k}} \right) \right\}.$$ 

**Lemma 2.** The randomization of $C$ in bootstrap is perfectly hiding.

**Proof.** The proof is the same as Lemma 1. \qed

### 5.5 Vertex Composition

The core idea in the vertices statement is to prove the computation correctness for the cumulative products of all vertex identifiers $i$ and labels $f_V(i)$, respectively, in a graph $G$. Also, vertices proves the cumulative products are a subset of vertex identifier universe (e.g. $i \in \Xi_V$) and label universe (e.g. $f_V(i) \in \Xi_L$), respectively. We note that the proof semantics of this proof are different from the original graph signature scheme [18] as the latter’s vertex composition statement only decomposed the graph signature into singleton commitments on encodings of individual vertices.
Let \( V = \bigcup \{ V_1, \ldots, V_k \} \) be the vertex set in a graph \( G \), the vertices statement is described as the following protocol:

\[
PK\{(\forall i \in V : \varepsilon_{b_i}, \varepsilon_{l_i}) : e\left(\prod_{i \in V} C_i, X_{0_0}\right) = \prod_{l=1}^{\ell} e\left(W_{l}, \prod_{k=0}^{1} X_{e\varepsilon_{l_k}}\right) \land \\
e(W_{V_1}, X_{0_0}) = e\left(a_{0_0}, \prod_{k=0}^{1} X_{e\varepsilon_{l_k}}\right) \land e(a_{0_0}, W_{L_1}) = e(W_1, X_{0_0}) \land \\
e(W_{V_2}, X_{0_0}) = e\left(W_{V_1}, \prod_{k=0}^{1} X_{e\varepsilon_{l_k}}\right) \land e(W_{V_2}, X_{0_0}) = e\left(W_{V_2}, \prod_{k=0}^{1} X_{e\varepsilon_{l_k}}\right) \land \\
\cdots \land e(a_{0_0}, W_{V_1}) = e\left(W_{V, l-i}, \prod_{k=0}^{1} X_{e\varepsilon_{l_k}}\right) \land e\left(\prod_{k=0}^{1} a_{0_{l_k}}, X_{0_0}\right) = e\left(W_{\varepsilon_{l-i}} \setminus V, W_{V_1}\right) \land \\
e(a_{0_0}, W_{L_2}) = e(W_2, W_{L_1}) \land e(a_{0_0}, W_{L_2}) = e(W_3, W_{L_2}) \land \\
\cdots \land e(a_{0_0}, W_{L_1}) = e\left(W_{l}, W_{L_1-1}\right) \land e\left(\prod_{k=0}^{1} a_{0_{l_k}}, X_{0_0}\right) = e\left(W_{\varepsilon_{l-i}} \setminus f_v(v), W_{L_1}\right)
\]

where \( \{m_{v_k}\} = \text{MP Encode}(\varepsilon_{v}), \{m_{L_k}\} = \text{MP Encode}(\varepsilon_{L}), W_{l} = \prod_{k=0}^{f_v(i_l)} a_{0_{k}}, \{\varepsilon_{l_0}, \varepsilon_{l_1}\} = r_i \times \text{MP Encode}(\{i_l\}) \) and \( \{m_{i_l}\} = q_i \cdot r_i^{-1} \times \text{MP Encode}(f_v(i_l)) \) for randomly selected \( q_i, r_i \in \mathbb{Z}_p^\ast \). The public inputs \( (W_1, \ldots, W_\ell) \) are witnesses for the vertex labels, \( (W_{V_1}, \ldots, W_{V_\ell}, W_{\varepsilon_{l-i}}) \) are witnesses for the cumulative product of vertex identifiers while \( (W_{L_1}, \ldots, W_{L_\ell}, W_{\varepsilon_{l-i}} \setminus f_v(v)) \) are witnesses for the cumulative product of vertex labels. The first statement is the bootstrap statement while the correctness for the cumulative products can be verified from the following equations:

\[
e(W_{V_1}, X_{0_0}) = e\left(a_{0_0}, \prod_{k=0}^{1} X_{e\varepsilon_{l_k}}\right) = e\left(\prod_{k=0}^{1} a_{0_{l_k}}, X_{0_0}\right) = e\left(W_{V, l-i}, \prod_{k=0}^{1} X_{e\varepsilon_{l_k}}\right)
\]

and,

\[
e(a_{0_0}, W_{L_1}) = e\left(a_{0_0}, X_{0_0}\right) = e\left(\prod_{k=0}^{1} a_{0_{l_k}}, X_{0_0}\right) = e\left(W_{l}, W_{L_1-1}\right)
\]
The proofs of cumulative product for the vertex identifiers and labels implicitly prove the pair-wise differences for every vertex. Let \( W_{V_0} = a_{0_V}, W_{L_0} = X_{0_L} \), simplifying the pairing notations in the proof above gives:

\[
PK\left\{ \left( \forall i \in V : \varepsilon_i, \varepsilon_i \right) : e\left( \prod_{i \in V} C_i, X_0 \right) = \ell e\left( W_i, \prod_{k=0}^{l} X_{0_k}^\varepsilon_i \right) \right\}
\]

\[
e\left( \prod_{k=0}^{\ell} a_{0_k} \prod_{l=1}^{\ell} W_{V_l}, X_0 \right) = \prod_{l=1}^{\ell} e\left( W_{V_{l-1}}, \prod_{k=0}^{l} X_{0_k}^\varepsilon_i \right) e\left( a_{0_l} \prod_{k=0}^{l-1} X_{0_k}^\varepsilon_i \right)
\]

\[
e\left( \prod_{k=0}^{\ell} a_{0_k}, X_0 \right) e\left( a_{0_l}, \prod_{l=1}^{\ell} W_{L_l} \right) = \prod_{l=1}^{\ell} e\left( W_l, W_{L_{l-1}} \right) e\left( W_{\Xi_{L \setminus f_V(V)}}, W_{L_l} \right)
\]

to establish the vertices(V) statement\(^1\) where \( C_i \) links to the graph signature. In the subsequent sections, we use \( \varepsilon_i[C_i] \in \Xi_V \) and \( W_i[C_i] \subseteq \Xi_L \) as the short form for the last two statements. They are read as \( \varepsilon_i \) encoded in commitment \( C_i \) is a member of \( \Xi_V \), and correspondingly, \( W_i \) encoded in commitment \( C_i \) is a witness for a member of \( \Xi_L \).

**Lemma 3.** The randomization of \((W_{V_1}, \ldots, W_{V_\ell}, W_{\Xi_{V \setminus V}})\) and \((W_{L_1}, \ldots, W_{L_\ell}, W_{\Xi_{L \setminus f_V(V)}})\) in vertices are perfectly hiding.

**Proof.** The random values \( r_l, o_l \) turn the witnesses in the cumulative products:

\[
W_{V_l} = \left( a_{0_0}^{\prod_{k=1}^{l} (x'+i_k)} \right)^{\prod_{k=1}^{l} r_k}, W_{\Xi_{V \setminus V}} = \left( a_{0_0}^{\prod_{k=1}^{l} (x'+i_k)} \right)^{\prod_{k=1}^{l} r_k^{-1}}
\]

and

\[
W_{L_l} = \left( X_0^{\prod_{w \in f_V(i_k)} (x'+w)} \right)^{\prod_{k=1}^{l} o_k r_k^{-1}}, W_{\Xi_{L \setminus E}} = \left( a_{0_0}^{\prod_{k=1}^{l} (x'+i_k)} \right)^{\prod_{k=1}^{l} o_k^{-1} r_k}
\]

into MoniPoly commitments which are perfectly hiding. \( \square \)

### 5.6 Edge Composition

The edge composition works in the similar way as the vertex composition and is, again, conceptually different from the edge composition statement of the

\(^1\) The first statement can be combined with the second and third statements to save another \( \ell + 2 \) pairing operations but we do not present the proof as such for clarify purposes.
original graph signature scheme [18]. Here, we need to run at least three proofs of cumulative products: one for the even-indexed edge identifiers, one for the odd-indexed edge identifiers and one for the edge labels. This is because all but the first and the last vertex identifiers appear at least twice in the edges. The number of set membership proof increases with respect to the number of branches in a graph. A simple way to separate the edges is to traverse the entire graph and record the edge identifiers into vertex sets \( E_1, \ldots, E_\ell \) such that every vertex identifier is unique in its own set. We first consider the scenario of an undirected acyclic graph. The protocol below establishes the statement \( \text{edges}(E) \):

\[
PK\left\{(\forall (i, j) \in E : \varepsilon_{(i, j)_0}, \varepsilon_{(i, j)_1}, \varepsilon_{(i, j)_2}) : \\
\prod_{(i, j) \in E} C_{(i, j)}, X_{0_0} = \prod_{(i, j) \in E} e\left(W_{(i, j)}, \prod_{j=0}^{2} X_j^{\varepsilon_{(i, j)_j}}\right) \land \\
\varepsilon_{(i, j)_0}[C_{(i, j)_0}] \in \Xi_V \land \cdots \land \varepsilon_{(i, j)_2}[C_{(i, j)_2}] \in \Xi_V \land \\
W_{(i, j)}[C_{(i, j)_E}] \in \Xi_L \right\}
\]

where the first statement is the bootstrap statement.

**Lemma 4.** The randomization of \( \{W_{V_1}, \ldots, W_{V_{|E_1|}}, W_{\Xi_V \setminus E_{k=1}}\} \) and \( \{W_{L_1}, \ldots, W_{\Xi_L \setminus \mathcal{E}(V)}\} \) in \( \text{edges} \) are perfectly hiding.

**Proof.** The proof is the similar to that of Lemma 3.

In addition, the edge identifier in a directed acyclic graph is represented by three vertex identifiers. For instance, \((i, j, j)\) represents a directed edge from \(i\) to \(j\). We can construct the edge composition proof for directed acyclic graph with a minor modification to its undirected counterpart. To be precise, the prover runs the proof of undirected version using \((i, j)\) extracted from \((i, j, j)\), in addition to proving \(j\) is a valid vertex identifier from \(\Xi_V\). We describe the proof as follows:

\[
PK\left\{(\forall (i, j) \in E : \varepsilon_{(i, j)_0}, \varepsilon_{(i, j)_1}, \varepsilon_{(i, j)_2}, \varepsilon_{(i, j)_3}, \varepsilon_{j_0}, \varepsilon_{j_1}) : \\
e\left(\prod_{(i, j) \in E} C_{(i, j)_j}, X_{0_0}\right) = \prod_{(i, j) \in E} e\left(W_{(i, j)_j}, \prod_{k=0}^{3} X_{0_k}^{\varepsilon_{(i, j)_k}}\right) \land \\
e\left(a_{0_0}, \prod_{(i, j) \in E} X_{0_k}^{\varepsilon_{(i, j)_k}}\right) = \prod_{(i, j) \in E} e\left(W_{(i, j)_j}, a_{0_1} X_{0_0}^{\varepsilon_{j_0}}\right) \land \\
W_{(i, j)}[C_{(i, j)_E}] \in \Xi_V \land \cdots \land W_{(i, j)}[C_{(i, j)_E}] \in \Xi_V \land \varepsilon_{j}[C_{(i, j)_j}] \in \Xi_V \land \\
W_{(i, j)}[C_{(i, j)_E}] \in \Xi_L \right\}
\]
where an additional proof of cumulative product is needed for every extra vertices \( j \in E \). At this point, it is clear that whether a graph is acyclic or cyclic does not have an impact on our edges statement. In the subsequent sections, we assume the graph is always an undirected graph in order to ease the explanation.

5.7 Graph Well-formedness

Combining the statements above, we can construct a proof of well-formedness for an encoded graph as below:

\[
PK\{ (\forall i \in V : \epsilon_i \in \mathbb{Z}, \forall (i, j) \in E : \epsilon_{(i,j)} \in \mathbb{Z}, \epsilon_{(i,j)} \neq \epsilon_{(i',j')}, \forall vertices(V) \land edges(E)) \}
\]

**Theorem 4.** The compound proof of knowledge of graph, bootstrap, vertices and edges establishes the well-formedness according to Def. 5 of an encoded graph.

**Proof.** Lemma 1 to Lemma 4 reduce the hardness of breaking the perfect hiding property in the proofs to breaking that of MoniPoly set commitment and Pedersen set commitment. This implicitly reduces the hardness of breaking the graph well-formedness to the hardness of breaking the binding property of the two commitment schemes which assume the hardness of the co-DLOG problem and DLOG problem, respectively. This is because graph, vertices and edges prove the required properties listed in Def. 5. Specifically, the bootstrap statement links graph to the vertices and edges statements, whose set membership proofs and cumulative product proofs prove that: (1) all committed exponent must fall in a system defined universes, (2) all committed exponents are pair-wise different with respect to the system defined universes, (3) every vertex (resp. edge) contains one identifier (resp. two identifiers) and zero or more labels. Therefore, the graph well-formedness holds if co-DLOG and DLOG problems are hard. \( \square \)

6 A New Graph Signature Scheme

6.1 Interface

We adapt the scheme interface from the MoniPoly anonymous attribute-based credential (ABC) system [28, 29] for our new graph signature scheme\(^2\) In the graph signature context, the access control policy \( \phi_{stmt} \) is termed as the predicate an we view a graph signature as six algorithms \( GS = \{ \text{KeyGen}_S, \text{KeyGen}_U, \text{Obtain}, \text{Sign}, \text{Prove}, \text{Verify} \} \).

\(^2\) While Groß’ graph signature scheme [18] was the first and only one of its kind and bears similarities to the Camenisch-Groß ABC system [10], both systems inherited their security properties from the underlying CL signature scheme and its proof system and were not proven rigorously for their overall systems. As we aim for a tight reduction proof, a properly defined interface and security notions, such as that by MoniPoly ABC system [29] appear to be a sound starting point.
To fully exploit the features of a graph signature scheme, we further divide the signing process (Obtain, Sign) into three parts: 1) begins with an initial signing (InitObtain, InitSign), 2) optionally continues to at least one intermediate signing for graph accumulation (IntermObtain, IntermSign), 3) ends with a final signing (FinalObtain, FinalSign). If a user runs the initial signing followed by the final signing, it resembles Groß’s single signing [18]. If there is at least an intermediate signing in between the initial and final signings, it realizes Groß’s joint graph signing [18]. This design makes the support of multi-signer applications a trivial effort, while still conforming to our rigorous security model. The scheme interface is described as follows:

1. **KeyGen}_S)(1^k, 1^L, 1^n) → (pk_S, sk_S): This algorithm is executed by the signer (S). On the input of the security parameter k, maximum supported graph size L and the graph attributes upper bound n, it generates a key pair (pk_S, sk_S).
2. **KeyGen}_U(pk_S) → (sk_U): This algorithm is executed by the user to generate a user (U) secret key pair sk_U.
3. (Obtain(pk_S, A_U), Sign(pk_S, sk_S, A_S)) → (σ or ⊥): The signing protocol outputs a valid graph signature σ for two message sets (A_U, A_S) if it completes successfully. Otherwise, it outputs a null value ⊥. The signing protocol starts with the initial signing and ends with the final signing:
   (a) (InitObtain(pk_S, {sk_U,s′}), InitSign(pk_S, sk_S, {G_S})) → (σ_{init} or ⊥): The signing process always starts with this initial signing protocol. It signs a committed user secret key sk_U and a graph data set G_S given by the signer. User private input is a set A_U = {sk_U,s′} which contains the user secret key sk_U and its opening value s′. Signer private inputs are the signer secret key sk_S and a graph data set A_S = G_S. At the end of the protocol, the algorithm outputs a valid initial signature σ_{init} and a null value ⊥ otherwise.
   (b) (IntermObtain(pk_S, {sk_U,s′,G_U, {o_i,o_{(i,j)}}}), IntermSign(pk_S, sk_S, {G_S})) → (σ_{interm} or ⊥): This optional intermediary signing protocol signs a committed user secret key sk_U, a committed hidden graph data set G_U and a graph data set G_S given by the signer. User private input is a set A_U = {sk_U,s′,G_U, {o_i,o_{(i,j)}}} which contains the user secret key sk_U and his hidden graph G_U where s′ and {o_i,o_{(i,j)}} are their opening values, respectively. Signer private inputs are the signer secret key sk_S and a graph data set A_S = G_S. At the end of the protocol, the algorithm outputs a valid intermediate signature σ_{interm} and a null value ⊥ otherwise.
   (c) (FinalObtain(pk_S, {sk_U,s′,G_U, {o_i,o_{(i,j)}}}), FinalSign(pk_S, sk_S, ⊥)) → (σ or ⊥): The signing process always ends with this final signing protocol. It signs a committed user private key sk_U and a committed hidden graph G_U. User private input is a set A_U = {sk_U,s′,G_U, {o_i,o_{(i,j)}}} which contains the user secret key sk_U and his hidden graph G_U where s′ and {o_i,o_{(i,j)}} are their opening values, respectively. Signer private input is the signer secret key sk_S. At the end of the protocol, the algorithm outputs a valid graph signature σ and a null value ⊥ otherwise.
4. \( (\text{Prove}(pk_S, \sigma, \phi_{\text{stmt}}), \text{Verify}(pk_S, \phi_{\text{stmt}})) \rightarrow (b) \): This interactive showing protocol establishes a show proof for the predicate \( \phi_{\text{stmt}} \) requested by verifier such that \( \phi_{\text{stmt}}(A) = 1 \). If \( \phi_{\text{stmt}}(A) = 0 \), the graph signature holder aborts and \( \text{Verify} \) outputs \( b = 0 \). At the end of the protocol, \( \text{Verify} \) outputs \( b = 1 \) if it accepts prover’s proof and outputs \( b = 0 \) otherwise.

In the subsequent sections, we will use \( \text{Obtain} \) (resp. \( \text{Sign} \)) to represent all three sub-modules.

6.2 Security Requirements

The graph signature scheme is governed by the security requirements \textit{impersonation resilience} and \textit{unlinkability} (for graphs and protocol), whose general forms were first introduced in the MoniPoly ABC [28, 29].

\textit{Impersonation Resilience}. We define our security model as the security against impersonation under active and concurrent attacks (\textit{imp-aca}) in the game between an adversary \( A \) and a challenger \( C \) as follows.

\textbf{Game 1 (imp – aca(\( A, C \))}

1. \textit{Setup}: \( C \) runs \( \text{KeyGen}(1^k, 1^n) \) and sends \( pk_S \) to \( A \).
2. \textit{Phase 1}: \( A \) is able to issue concurrent queries to the \( \text{Obtain}, \text{Prove} \) and \( \text{Verify} \) oracles where he plays the role of user, prover and verifier, respectively. \( A \) can query a set \( A_{U_i} \) (resp., \( \phi_i \)) of his choice to \( \text{Obtain} \) (resp., \( \text{Prove} \) and \( \text{Verify} \)) in the \( i \)-th query. \( A \) can also issue queries to the \( \text{SignTranscript} \) oracle which takes in \( A_{U_i} \) and returns its signing protocol transcript.
3. \textit{Challenge}: \( A \) outputs the challenge data set \( A_{\ast U} \) and its corresponding predicate \( \phi^\ast \) such that \( \phi^\ast(A_{U_i}) = 0 \) and \( \phi^\ast(A_{\ast U}) = 1 \) for every \( A_{U_i} \) queried to the \( \text{Obtain} \) oracle during Phase 1.
4. \textit{Phase 2}: \( A \) can continue to query the oracles as in Phase 1 with the restriction that it cannot query a data set \( A_{U_i} \) to \( \text{Obtain} \) such that \( \phi^\ast(A_{U_i}) = 1 \).
5. \textit{Impersonate}: \( A \) completes a showing protocol as the prover with \( C \) as the verifier for the predicate \( \phi^\ast(A_{\ast U}) = 1 \). \( A \) wins the game if \( C \) outputs 1.

\textbf{Definition 6.} An adversary \( A \) is said to \( (t_{\text{imp}}, \varepsilon_{\text{imp}}) \)-break the security against impersonation under active and concurrent attacks (\textit{imp-aca}) of a graph signature scheme if \( A \) runs in time at most \( t_{\text{imp}} \) and wins in Game 1 such that:

\[ \Pr[(A, \text{Verify}(pk, \phi^\ast)) = 1] \geq \varepsilon_{\text{imp}} \]

for a negligible probability \( \varepsilon_{\text{imp}} \). We say that a graph signature scheme is \textit{imp-aca}-secure if no adversary \( (t_{\text{imp}}, \varepsilon_{\text{imp}}) \)-wins Game 1.
Unlinkability. The security model for graph unlinkability under active and concurrent attacks (gunl-aca) is defined as a game between an adversary $A$ and a challenger $C$. The gunl-aca security does not consider the collusion in between $A$ and signer. We argue that such security is sufficient for a graph signature scheme because the signer is always a trusted party.

Game 2 (gunl – aca$(A,C)$)

1. **Setup:** $C$ runs KeyGen and sends $pk_S, sk_S$ to $A$.
2. **Phase 1:** $A$ is able to issue concurrent queries to the Commit, Obtain, Prove and Verify oracles where he plays the role of committer, user, signer, prover and verifier, respectively, on any hidden set $A_{U_0}$ of his choice in the $i$-th query. $A$ can also issue queries to an additional oracle, namely, Corrupt that when queried with a protocol session identifier, returns the entire internal state of a user in a signing protocol, or the entire internal state of a prover in the showing protocol.
3. **Challenge:** $A$ decides the two non-empty hidden set $A_{U_0} = \{ sk_{U_0}, s'_0, G_{U_0} = (V_0 \cup E_0, \{ o_0, o_{1,j} \} ) \}$, $A_{U_1} = \{ sk_{U_1}, s'_1, G_{U_1} = (V_1 \cup E_1, \{ o_1, o_{1,j} \} ) \}$ with $|V_0| = |V_1|, |E_0| = |E_1|$ and the predicate $\phi^*$ which he wishes to challenge such that $\phi^*(A_{U_0}) = \phi^*(A_{U_1}) = 1$. $A$ is allowed to select $A_{U_0}, A_{U_1}$ from the existing queries to Obtain in Phase 1. $C$ responds by randomly choosing a challenge bit $b \in \{0,1\}$ and interacts as the user with $A$ as the signer to complete the protocols in the order:

   - $(\text{InitObtain}(pk_S, \{ sk_{U_0}, s'_0 \} ), \text{InitSign}(pk_S, sk_S, \{ G_{U_0} \} )) \rightarrow \sigma_{\text{init},b}$,
   - $(\text{FinalObtain}(pk_S, A_{U_0}), \text{FinalSign}(pk_S, sk_S, \bot) ) \rightarrow \sigma_b$,
   - $(\text{InitObtain}(pk_S, \{ sk_{U_1}, s'_1 \} ), \text{InitSign}(pk_S, sk_S, \{ G_{U_1} \} )) \rightarrow \sigma_{\text{init},1-b}$,
   - $(\text{FinalObtain}(pk_S, A_{U_1}), \text{FinalSign}(pk_S, sk_S, \bot) ) \rightarrow \sigma_{1-b}$,

where $sk_{U_0}, s'_0$ are randomly chosen by $C$. If there are $n$ times intermediary signing in between, the challenge graph is formatted as $G_{U_b} = \bigcup \{ G_{b_0}, \ldots, G_{b_n} \}$ such that $G_{b,i} = (V_{b,i} \cup E_{b,i}), |V_{b,i}| = |V_{1-b,i}|, |E_{b,i}| = |E_{1-b,i}|$ for $0 \leq i \leq n$.

The initial signer signs the graphs $G_{b_0}, G_{1-b,0}$ and the $i$-th intermediary signer signs on the graphs $G_{b,i}, G_{1-b,i}$. Subsequently, $C$ interacts as the prover with $A$ as the verifier for polynomially many times as requested by $A$ to complete the protocols in the same order:

   - $(\text{Prove}(pk_S, \sigma_b, \phi^*), \text{Verify}(pk_S, \phi^*) ) \rightarrow 1$,
   - $(\text{Prove}(pk_S, \sigma_{1-b}, \phi^*), \text{Verify}(pk_S, \phi^*) ) \rightarrow 1$.

4. **Phase 2:** $A$ can continue to query the oracles as in Phase 1 except querying the transcripts of the challenged signing and presentation protocols to Corrupt.
5. **Guess:** $A$ outputs a guess $b'$ and wins the game if $b' = b$.  

23
Definition 7. An adversary $A$ is said to $(t_{\text{gunl}}, \varepsilon_{\text{gunl}})$-break the security of graph unlinkability under active and concurrent attacks (gunl-aca) of a graph signature scheme if $A$ runs in time at most $t_{\text{gunl}}$ and wins in Game 2 such that:

$$|\Pr[ b = b'] - \frac{1}{2}| \geq \varepsilon_{\text{gunl}}$$

for a negligible probability $\varepsilon_{\text{gunl}}$. We say that a graph signature is gunl-aca-secure if no adversary $(t_{\text{gunl}}, \varepsilon_{\text{gunl}})$-wins Game 2.

The protocol unlinkability under active and concurrent attack (punl-aca) is defined as a game between an adversary $A$ and a challenger $C$ as follows:

Game 3 (punl-aca($A, C$))

1. **Setup:** Same to that of graph unlinkability game.
2. **Phase 1:** Same to that of graph unlinkability game.
3. **Challenge:** Same as that of graph unlinkability game except that $C$ responds by randomly choosing two challenge bits $b_1, b_2 \in \{0, 1\}$ and interacts as the user with $A$ as the signer to complete the protocols in the order:

$$(\text{InitObtain}(pk_s, \{sk_U, s'\}), \text{InitSign}(pk_s, sk_s, \{G_{Ub_1}\})) \rightarrow \sigma_{\text{init}, b_1},$$

$$(\text{FinalObtain}(pk_s, A_{Ub_1}), \text{FinalSign}(pk_s, sk_s, \bot)) \rightarrow \sigma_{b_1},$$

$$(\text{InitObtain}(pk_s, \{sk_U, s'\}), \text{InitSign}(pk_s, sk_s, \{G_{Ub_2}\})) \rightarrow \sigma_{\text{init}, b_2},$$

$$(\text{FinalObtain}(pk_s, A_{Ub_2}), \text{FinalSign}(pk_s, sk_s, \bot)) \rightarrow \sigma_{b_2}.$$

If there are $n$ intermediary signing protocols, they also follow the order above. Subsequently, $C$ interacts as the prover with $A$ as the verifier for polynomially many times as requested by $A$ to complete the protocols in the same order:

$$(\text{Prove}(pk_s, \sigma_{b_2}, \phi^*), \text{Verify}(pk_s, \phi^*)) \rightarrow 1,$$

$$(\text{Prove}(pk_s, \sigma_{1-b_2}, \phi^*), \text{Verify}(pk_s, \phi^*)) \rightarrow 1.$$

4. **Phase 2:** Same to that of graph unlinkability game.
5. **Guess:** $A$ outputs a guessed pair of signing protocol transcript $\pi_{(O,S)}$ and show proof transcript $\pi_{(P,V)}$ and wins the game if the pair is under the same credential such that $\sigma_{\pi_{(O,S)}} = \sigma_{\pi_{(P,V)}}$.

Definition 8. An adversary $A$ is said to $(t_{\text{punl}}, \varepsilon_{\text{punl}})$-break the security of the protocol unlinkability under active and concurrent attacks (punl-aca) of a graph signature scheme if $A$ runs in time at most $t_{\text{punl}}$ and wins in Game 3 such that:

$$|\Pr[\sigma_{\pi_{(O,S)}} = \sigma_{\pi_{(P,V)}}] - \frac{1}{2}| \geq \varepsilon_{\text{punl}}$$

for a negligible probability $\varepsilon_{\text{punl}}$. We say that an ABC system is punl-aca-secure if no adversary $(t_{\text{punl}}, \varepsilon_{\text{punl}})$-wins Game 3.
6.3 Construction

In this section, we first describe the KeyGen of the graph signature, followed by the three graph signing protocols. At a quick glance, we show the structure of the graph signature, which is a SDH-CL signature on the MoniPoly encoded graph:

\[ v^{x+t} = h^{sk_u} \cdot a_{i,0}^{(x'+i)} \prod_{k \in f_v(x'+k)} \ldots a_{(i,j),0}^{(x'+i)(x'+j)} \prod_{k \in f_e(x'+k)} \ldots b^c \]

\( \forall \text{ vertices } i \)

\( \forall \text{ edges } (i,j) \)

6.3.1 Key Generation

**Signer S.** Construct three cyclic groups \( G_1, G_2, G_T \) of order \( p \) based on an elliptic curve whose bilinear pairing is \( e : G_1 \times G_2 \rightarrow G_T \). Choose two random secret values \( x, x' \in \mathbb{Z}_p^* \) and select random generators \( b, c, d, h \in G_1, \{a_i, X_i \in G_2\}_{i=0}^L \) to compute the values \( \{a_{ik} = a_i^{x_k}, X_{ik} = X_i^{x_k}\}_{i=0}^L \). Define the function \( \text{MPEncode} : \mathbb{Z}_p^n \rightarrow \mathbb{Z}_p^{n+1} \) that converts a set of \( n \) attributes into coefficients for a monic polynomial of degree \( n + 1 \). The public key is \( pk = (e, G_1, G_2, G_T, p, b, c, d, h, \{a_{ik}, X_{ik}\}_{i=0}^L, X, \Xi_V, \Xi_L) \) and the secret key is \( sk = (x, x') \) where \( \Xi_V \) is the list of vertex identifiers and \( \Xi_L \) is the list of vertex and edge labels. The parameter \( L \) is the maximum vertices and edges allowed in a graph, while \( n \) indicates that each vertex can have no more than \( n - 1 \) labels and each edge can have no more than \( n - 2 \) labels.

**User U.** The user generates the user secret key as \( sk_U \in \mathbb{Z}_p^* \).

6.3.2 Graph Signing As the signing protocols require the signers to transfer the graphs to the user, we assume a secure channel is always in place.

**Initial Signing.** The initial signing protocol \( \text{InitObtain/InitSign} \) begins with a user providing a proof of representation for the Pedersen commitment \( C \) of his secret key \( sk_U \). If the proof is verified, using the CL-SDH signature, signer signs on \( C \) together with an assigned graph \( G_S \) decided by the signer. If the returned graph signature \( \sigma' \) is a valid signature on \( C \) and \( G_S \), the user finalizes the graph signature as \( \sigma_{\text{init}} \).

We assume the signer knows the discrete logarithms of bases \( d, a_{0a}, a_{1a}, \ldots, a_{La} \) with respect to \( b \). Without loss of generality, we also assume the signer always utilize the bases in an incremental sequence such that \( a_{1a}, \ldots, a_{La} \). For \( G_S \), let the vertex exponents be \( \bar{e}_i = d \log_b(a_{i,0})(x' + i) \prod_{w \in f_v(i)} (x' + w) \) while the edge exponents be \( \bar{e}_{(i,j)} = d \log_b(a_{(i,j),0})(x' + i)(x' + j) \prod_{w \in f_e(i,j)} (x' + w) \). The initial graph signing protocol is as follows:

1. User randomly selects \( s' \in \mathbb{Z}_p^* \) and interacts with signer to prove the well-formedness of his hidden graph:

\( PK\{(\zeta, \rho) : C = h^\zeta b^\rho\} \)

where \( \zeta = sk_U \) and \( \rho = s' \).
2. If the proof is verified, signer signs on \( C \) and an assigned graph \( G_S \) as:

\[
v = \left( C b^{s''+\cdots+\epsilon_i+\cdots+\epsilon_{(i,j)}+\cdots+d\log d} \right)^{(x+t)^{-1}}
\]

for randomly selected \( s'', t \in \mathbb{Z}_p^* \). Signer returns \( (\sigma' = (t, s'', v), G_S) \) to the user.

3. If \( \sigma' \) is a valid SDH-CL signature on \( C \) and \( G_S \), user finalizes his graph signature as \( \sigma_{init} = (t, s, v, G_U) \) where \( s = s' + s'' \mod p \) and hidden graph \( G_U = G_S \).

**Remark 1.** The signing protocol above dispense with a secure channel by employing an additional encryption step. The Pedersen commitment of the user secret key \( C = h^{sk_U b^s} \) can be treated as an ElGamal public key. The signer can perform a hybrid ElGamal encryption on the signature and graphs to hides all the information transferred to the user. This approach is also applicable to the intermediary and final signing protocols.

**Intermediary Signing.** The intermediary signing protocol is useful for the applications which require the user to approach different signers to gather the needed graphs. This protocol is the same as above except it begins with a user providing a proof of possession for a blinded initial signature \( \sigma'_{init} = (t, s, v) \) such that:

\[
v' = v r_1 y^{-1}, s' = s r_1 \mod p, t' = t y \mod p
\]

where \( r_1, y \in \mathbb{Z}_p^* \) are the random blinding factors. The correctness for the blinded signature \( \sigma'_{init} \) can be verified as follows:

\[
e \left( h^{sk_U r_1} \prod_{i \in V} C_i^r \prod_{(i,j) \in E} C_{(i,j)}^{r_{(i,j)}} b^s d^{i} v^{t-t'}, X_{0_0} \right)
\]

\[
= e \left( h^{sk_U} \prod_{i \in V} C_i \prod_{(i,j) \in E} C_{(i,j)} b^s d^{i} (v^{-1})^{-t y}, X_{0_0} \right)^{r_1}
\]

\[
= e(v(x+t) v^{-t}, X_{0_0})^{r_1}
\]

\[
= e(v^y, X)
\]

where \( G_U = (V \cup E) \). The user also explicitly proves the correctness for the Pedersen commitment of his secret key \( sk_U \):

\[
C_1 = h^{sk_U r_1 b^{sr_1}}
\]

and the MoniPoly commitment of his hidden graph \( G_U \):

\[
C_i = a_{\alpha_i} a_{\alpha_i (x'+i)} \prod_{w \in f_U(i)} (x'+w) C_{(i,j)} = a_{\alpha_{(i,j)} (x'+i)} (x'+j) \prod_{w \in f_U(i,j)} (x'+w),
\]

\[
C_2 = \prod_{i \in V_U} C_i^{r_{(i,j)}^{-1}} \prod_{(i,j) \in E_U} C_{(i,j)}^{r_{(i,j)}^{-1}}
\]

26
where $r_1, o_{i_1}, o_{(i,j)_1} \in \mathbb{Z}_p^*$ are the random blinding factors. We describe the intermediary signing protocol as follows:

1. User randomly selects $r_1, y, o_{i_1}, o_{(i,j)_1}, r_i, r_{(i,j)} \in \mathbb{Z}_p^*$ and interacts with the signer to prove the possession of $\sigma_{\text{init}}$ and the representation of his hidden graph $G_U$:

   $$PK\left\{ (\forall i \in V_U : \varepsilon_{i_0}, \varepsilon_{i_1}, \varepsilon_i), (\forall (i,j) \in E_U : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \varepsilon_{(i,j)}), \zeta, \rho, \omega, \tau, \gamma \right\} :$$

   $$e(C_1C_2d^\omega b^{\tau - \gamma}, X_{0_i}) = e(u^{\gamma'}, X) \land C_1C_2 = h^\omega b^\rho \prod_{i \in V_U} C_i^{\varepsilon_i} \prod_{(i,j) \in E_U} C_{(i,j)}^{\varepsilon_{(i,j)}} \land$$

   $$e\left( \prod_{i \in V_U} C_i \prod_{(i,j) \in E_U} C_{(i,j)}, X_{0_i} \right) =$$

   $$\prod_{i \in V_U} e\left( W'_i, \prod_{i \in E_U} X_{0_i} \right) \prod_{(i,j) \in E_U} e\left( W'_{(i,j)}, \prod_{k = 0}^{1} X_{0_{(i,j)_k}} \right)$$

   where $\zeta = sk_U r_1, \rho = sr_1, \omega = r_1, \tau = ty, \gamma = y, \varepsilon_{i_0} = r_i, \varepsilon_{i_1} = r_1, \varepsilon_i = r_1 a_{i_1}^{-1}, \varepsilon_{(i,j)_1} = r_{(i,j)_1}, \varepsilon_{(i,j)_0} = r_{(i,j)} i, \varepsilon_{(i,j)} = r_1 a_{(i,j)}^{-1}$, while the witnesses are

   $$W'_i = a_{i_0} r_i^{-1} \Pi_{o \in fV(i)}(x' + w)$$

   and

   $$W'_{(i,j)} = a_{(i,j)_0} r_{(i,j)_1}^{-1} \Pi_{o \in fE(i,j)}(x' + w).$$

   Note that the exponent $i$ for vertices and edges can always be swapped with a randomly selected attribute $w$.

2. If the proof is verified, signer signs on $C_1, \{C_i, C_{(i,j)}\}$ and an assigned graph $G_S$ as:

   $$v = \left( C_1 \prod_{i \in V_U} C_i \prod_{(i,j) \in E_U} C_{(i,j)}b^{s'' + \cdots + \varepsilon_{i_1} + \cdots + \varepsilon_{(i,j)} + \cdots + \log_b(d)} \right)^{(x+t)^{-1}}$$

   for randomly selected $s''$, $t \in \mathbb{Z}_p^*$. Signer returns $(\sigma' = (t, s'', v), G_S)$ to the user.

3. If $\sigma'$ is a valid SDH-CL signature on $C_1, \{C_i, C_{(i,j)}\}$ and $G_S$, user updates his secret key as $sk_{U1}$ and graph signature as $\sigma_{\text{interm}} = (t, s_{r_1} + s'' + v, G, \{o_{i_1}, o_{(i,j)_1}\})$

   where $G = G_U + G_S$.

**Remark 2.** A user can interact with more than one intermediate signer to accumulate the required graphs.

**Final Signing.** The final signing protocol is the same as the intermediary signing protocol except the signer does not assign a new graph. Similarly, the user also proves the correctness for the Pedersen commitment of his secret key $sk_U$:

$$C_1 = h^sk_{U2}b^{sr_2}$$

27
and the MoniPoly commitment of his hidden graph \( G = G_U \cup G_S \):

\[
\forall i \in V_U : C_i = a_{i_0} \alpha_{i_1} (x' + i) \prod_{w \in f_U(i)} (x' + w),
\]

\[
\forall i \in V_S : C_i = a_{i_0} \alpha_{i_1} (y' + i) \prod_{w \in f_S(i)} (x' + w),
\]

\[
\forall i \in E_U : C_{(i,j)} = a_{(i,j)_0} \alpha_{(i,j)_1} (x' + i)(x' + j) \prod_{w \in f_E(i,j)} (x' + w),
\]

\[
\forall i \in E_S : C_{(i,j)} = a_{(i,j)_0} \alpha_{(i,j)_1} (x' + i)(x' + j) \prod_{w \in f_E(i,j)} (x' + w),
\]

\[
C_2 = \prod_{i \in V} C_i^{\tau_2 \alpha_{(i,j)_2}} \prod_{(i,j) \in E} C_{(i,j)}^{\tau_2 \alpha_{(i,j)_2}}
\]

where \( \tau, \alpha_{(i,j)_2}, \alpha_{(i,j)_1} \in \mathbb{Z}_p^* \) are the random blinding factor. We describe the final signing protocol as follows:

1. User randomly selects \( r, \alpha_{i_1}, \alpha_{(i,j)_2}, r_i, r_{i,j} \in \mathbb{Z}_p^{*} \) and interacts with the signer to prove the possession of \( \sigma_{\text{interm}} = (t, s, v, \{ \alpha_{i_1}, \alpha_{(i,j)_1} \}) \) and the representation of his hidden graph:

\[
PK\left\{ ((\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}, \varepsilon_i), (\forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \varepsilon_{(i,j)}), \zeta, \rho, \omega, \tau, \gamma) :
\right. \]

\[
\begin{align*}
& e\left( C_1 C_2 d^{\varepsilon_{i_1}} v'^{-\tau}, X_{0_0} \right) = e\left( v^{\tau}, X \right) \land C_1 C_2 = h^\varepsilon v^\rho \prod_{i \in V} C_i^{\varepsilon_i} \prod_{(i,j) \in E} C_{(i,j)}^{\varepsilon_{(i,j)}} \land \\
& \prod_{i \in V} \left( W_i^{r_i} \prod_{k=0}^{1} X_{0_k}^{\varepsilon_{i,k}} \right) \prod_{(i,j) \in E} \left( W_{(i,j)}^{r_{(i,j)}} \prod_{k=0}^{1} X_{0_k}^{\varepsilon_{(i,j),k}} \right)
\end{align*}
\]

where \( \varepsilon_{i_1} = r_i, \varepsilon_{i_0} = r_i, \varepsilon_{(i,j)_1} = r_{(i,j)}, \varepsilon_{(i,j)_2} = r_{(i,j)} \). We also have the witnesses as follows:

\[
\forall i \in V_U : W_i = a_{i_0} \alpha_{i_1} \gamma_{r_i} \prod_{w \in f_U(i)} (x' + w),
\]

\[
\forall i \in V_S : W_i = a_{i_0} \alpha_{i_1} \gamma_{r_i} \prod_{w \in f_S(i)} (x' + w),
\]

\[
\forall i \in E_U : W_{(i,j)} = a_{(i,j)_0} \alpha_{(i,j)_1} \gamma_{r_{(i,j)}} \prod_{w \in f_E(i,j)} (x' + w),
\]

\[
\forall i \in E_S : W_{(i,j)} = a_{(i,j)_0} \alpha_{(i,j)_1} \gamma_{r_{(i,j)}} \prod_{w \in f_E(i,j)} (x' + w)
\]

such that the exponent \( i \) for vertices and edges can be swapped with a randomly selected attribute \( w \).

2. If the proof is verified, signer signs on \( C_1 \) and \( \{ C_1, C_{(i,j)} \} \) as:

\[
v = \left( C_1 \prod_{i \in V} C_i \prod_{(i,j) \in E} C_{(i,j)} b^{x'' + \text{diag}_b(c)} \right)^{(x+t)^{-1}}
\]
for randomly selected $s''$, $t \in \mathbb{Z}_p^*$. Signer returns $(\sigma' = (t, s'', v))$ to the user.

3. If $\sigma'$ is a valid SDH-CL signature on $C_1$ and $\{C_i, C_{(i,j)}\}$, user finalizes his secret key as $sk_{U2}$ and graph signature as

$$\sigma = (t, sr_2 + s'', v, \mathcal{G}, \{o_{i_1}, o_{(i,j)_1}, o_{(i,j)_2}\}, \{o_{i_2}, o_{(j,i)_2}\})$$

where $\mathcal{G} = \mathcal{G}_{U} + \mathcal{G}_{S}$.

**Remark 3.** The proposed scheme can be modified to let the final signer assign a dummy graph that contains information such as the final signer’s identity, signing date and a unique signature ID for the purpose of revocation.

### 6.4 Security of the Graph Signature Scheme

#### 6.4.1 Impersonation Resilience

We establish the security of the graph signature scheme by constructing a reduction to the (co-)SDH problem. To achieve tight security reduction, we make use of Multi-Instance Reset Lemma [21] as the knowledge extractor which requires the adversary $\mathcal{A}$ to run $N$ parallel instances of impersonation under active and concurrent attacks. The challenger $\mathcal{C}$ can fulfill this requirement by simulating the $N-1$ instances from its given SDH instance which is random self-reducible [7]. Since this is obvious, we describe only the simulation for a single instance of impersonation under active and concurrent attacks in the security proofs.

**Theorem 5.** If an adversary $\mathcal{A} (t_{imp}^{imp}, \varepsilon_{imp})$-breaks the imp-aca-security of the proposed graph signature scheme, then there exists an algorithm $\mathcal{C}$ which $(t_{cosdh}^{cosdh}, \varepsilon_{cosdh})$-breaks the co-SDH problem such that:

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

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

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

$$t_{imp} \leq t_{sdh}/2N - T(q^2).$$

where $N$ is the total adversary instance, $q = Q_{(O, S)} + Q_{(P, V)}$ is the total query made to the Obtain and Verify oracles, while $T(q^2)$ is the time parameterized by $q$ to setup the simulation environment and to extract the SDH solution. Consider the dominant time elements $t_{imp}$ and $t_{sdh}$ only, we have:

$$\left(1 - \left(1 - \varepsilon_{imp} + \frac{1 + (q - 1)!}{p^{q-2}}\right)^N\right)^2 \leq \varepsilon_{sdh}, 2N t_{imp} \approx t_{sdh}.$$
Let $N = (\varepsilon_{\text{imp}} - \frac{1 + (q-1)!/p^{q-2}}{p})^{-1}$, we get $\varepsilon_{\text{sdh}} \geq (1 - e^{-1})^2 \geq 1/3$ and the success ratio is:

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

which gives a tight reduction.

Similar to the approach in the security proofs for MoniPoly graph signature scheme [29], we categorize the way an adversary impersonates in Table 1.

<table>
<thead>
<tr>
<th>Type $\mathcal{G}$</th>
<th>MP Encode($\mathcal{G}$)</th>
<th>$s$</th>
<th>$t$</th>
<th>$v$</th>
<th>Adversary Assumption</th>
<th>Lemma</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1</td>
<td>** **</td>
<td>$A_{\text{bind}}$</td>
<td>co-DLOG</td>
<td>Theorem 3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 0 0</td>
<td>0 0 0 0</td>
<td>$A_1$</td>
<td>SDH</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 0 0</td>
<td>0 0 1 0</td>
<td>$A_1$</td>
<td>DLOG</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 0 0</td>
<td>0 0 1 1</td>
<td>$A_2$</td>
<td>SDH</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 0 0</td>
<td>0 1 0 0</td>
<td>$A_1$</td>
<td>SDH</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 0 0</td>
<td>0 1 0 1</td>
<td>$A_1$</td>
<td>DLOG</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 0 0</td>
<td>1 0 1 0</td>
<td>$A_1$</td>
<td>SDH</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 0 0</td>
<td>1 1 1 0</td>
<td>$A_3$</td>
<td>SDH</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 0 0</td>
<td>1 1 1 1</td>
<td>$A_3$</td>
<td>DLOG</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9 1 1</td>
<td>0 0 0 0</td>
<td>$A_1$</td>
<td>SDH</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 1 1</td>
<td>0 0 1 0</td>
<td>$A_1$</td>
<td>DLOG</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 1 1</td>
<td>0 1 0 0</td>
<td>$A_2$</td>
<td>SDH</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12 1 1</td>
<td>0 1 1 0</td>
<td>$A_2$</td>
<td>DLOG</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13 1 1</td>
<td>1 0 0 0</td>
<td>$A_1$</td>
<td>SDH</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 1 1</td>
<td>1 0 1 0</td>
<td>$A_1$</td>
<td>N/A</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15 1 1</td>
<td>1 1 0 0</td>
<td>$A_3$</td>
<td>SDH</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 1 1</td>
<td>1 1 1 1</td>
<td>$A_3$</td>
<td>N/A</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: * = 1 or 0, 1 = queried, 0 = not queried, N/A = not available

We present Lemma 5, 6 and 7 corresponding to the adversaries $A_1$, $A_2$ and $A_3$ as follows.

**Lemma 5.** If an adversary $A_1 (t_{\text{imp}}, \varepsilon_{\text{imp}})$-breaks the imp-aca-security of the proposed graph signature scheme, then there exists an algorithm $C$ which $(t_{\text{sdh}}, \varepsilon_{\text{sdh}})$-solves the SDH problem such that:

$$\varepsilon_{\text{imp}} \leq N \sqrt{\varepsilon_{\text{sdh}}} - 1 + \frac{1 + (q-1)!/p^{q-2}}{p} + 1,$$

$$t_{\text{imp}} \leq t_{\text{sdh}}/2N - T(q^2).$$
where \( N \) is the total of adversary instances, \( q = Q_{(O,S)} + Q_{(P,V)} \) is the number of queries made to the Obtain and Verify oracles, while \( T(q^2) \) is the time parameterized by \( q \) to setup the simulation environment and to extract the SDH solution.

**Proof.** Given a \( q \)-SDH instance \((g_1, g_1^1, g_1^2, \ldots, g_1^n, g_2, g_2^2)\) where \( q = Q_{(O,S)} + Q_{(P,V)} \) is the maximum number of queries \( A_1 \) can issue to the Obtain and Verify oracles, we show that if \( A_1 \) exists, there exists an algorithm \( C \) which can output \((g_1^1, t)\) by acting as the simulator for the graph signature scheme as follows:

**Game 0.** This is the attack by \( A_1 \) on the real \( N \) instances of graph signature scheme. Let \( S \) be the event of a successful impersonation, by assumption, we have:

\[
\Pr[S_0] = \varepsilon_{imp}.
\]  

(1)

**Game 1.** In order to simulate the environment of the graph signature scheme, \( C \) uniformly and randomly selects distinct \( t_a, \{t_a\}_{i=1}^n, t_b, t_c, t_d, t_h, x' \), \( t_1, \ldots, t_q \in \mathbb{Z}_p^* \). Next, let \( f(x) \) denotes the polynomial \( f(x) = \prod_{k=1}^q (x + t_k) = \sum_{k=0}^q \rho_k x^k \) and \( f_u(x) \) denotes the polynomial \( f_u(x) = \prod_{k=1, k\neq u}^q (x + t_k) = \sum_{k=1}^q \lambda_k x^k \).

Let \( g_1^i(x) = \prod_{k=0}^q (g_1^k)^{\rho_k} \), \( C \) sends \((e, G_1, G_2, G_T, p, \{a_{t_a} = g_1^i(t_a x^k), X_{a_k} = g_2^{2k} \}_{k=0}^n, \{a_{t_b}, X_{t_b} \}_{i=1}^n, b = g_1^{f(x) t_h}, c = g_1^{f(x) t_h}, d = g_1^{f(x) t_h}, \tilde{h} = g_1^{f(x) t_h}, X = g_2^n)\) as the public key to \( A_1 \). \( C \) creates two empty lists \( L_{(FO, FS)} \) and \( L_{(P,V)} \) where the former stores the corrupted final signatures simulated during the final signing protocol while the latter stores the non-corrupted signatures simulated during the showing protocol. \( C \) also creates two empty lists \( L_{(IO, IS)} \) and \( L_{(IO, FS)} \) to store the corrupted initial signatures and intermediary signatures simulated during the initial and intermediary signing protocols, respectively. Since \( t_a, t_b, t_c, t_d, t_h, x' \) are uniformly random, the distribution of the simulated public key (and the corresponding random self-reducible \([7]\) \( N - 1 \) instances) is the same as that of the original scheme. So, we have:

\[
\Pr[S_1] = \Pr[S_0].
\]

(2)

**Game 2.** In this game, \( A_1 \) plays the role of multiple users to concurrently interact with the initial signer simulated by \( C \). Without loss of generality, we assume every user \( u \) uses different data set \( A_u = \{sk_{u}, s'_{u}, \perp, \perp\} \). If the \( u \)-th session of an initial signing protocol ends successfully, \( C \) produces a signature \( \sigma_{init} \) for \( A_1 \). Their interaction is as follows:

1. \( A_1 \) concurrently initializes the initial signing protocol with \( C \) by running the zero-knowledge protocol:

\[
PK\{(\zeta, \rho), C = h_\zeta b^\rho\}
\]

Without loss of generality, we assume \( A_1 \) always execute this protocol honestly. Therefore, \( C \) always reset successfully and can extract the secret exponents \( \zeta = sk_{u}, \rho = s'_{u} \) used by \( A_1 \) in the protocol above.

31
2. C chooses a random value $s''_u \in \mathbb{Z}_p^*$ and a graph $\mathcal{G}_{S_u} = \{V_{S_u} \cup E_{S_u}\}$ to set:

$$v_u = h_u^{s''_u} \prod_{i \in V_{S_u}} a_{u,i}^{m_u(i,i)} \prod_{(i,j) \in E_{S_u}} b_{u}^{m_u(i,j)}  + s''_u d_u$$

where $a_{u,i} = g_1^{f_u(x)u^{x^i}}$, $b_u = g_1^{f_u(x)t_u}$, $d_u = g_1^{f_u(x)t_u}$, $\{m_u, (i,j)\} = \text{MPEncode}(i \in V_{S_u})$ and $\{m_u(i,j)\} = \text{MPEncode}((i,j) \in E_{S_u})$. C adds the record $(t_u, s_u = s''_u, v_u, A_u, \mathcal{G}_{S_u})$ to $L_{(iO,iS)}$ and returns $\sigma''_u = (t_u, s''_u, v_u, \mathcal{G}_{S_u})$ as the signature to $A_1$.

Since C’s choices of $t_u, s''_u$ are independent of $A$’s view, a collision $v_i = v_j$ for some $i, j \leq q$ in $A$’s concurrent queries happens with a negligible probability of $\Pr[Col] = 1/p$ in which $A_1$ can compute the discrete logarithm $x$. Else, C simulates the init\text{Sign} oracle perfectly for every concurrent query and $A_1$ can formulate its signature $\sigma_{init_u} = (t_u, s_u = s''_u, v_u, \mathcal{G}_{U_u}, \mathcal{G}_{S_u})$ as in the original initial signing protocol. This gives:

$$\Pr[S_2] = \Pr[S_1] + \Pr[Col]$$

$$\leq \Pr[S_1] + \prod_{i=1}^{q-1} \frac{i}{p}$$

$$\leq \Pr[S_1] + (q - 1)!/p^{q-1} \tag{3}$$

where $A_1$ can make, at most, another $q - 1$ initial signature queries.

Game 3. In this game, $A_1$ plays the role of multiple users to concurrently interact with the intermediary signer simulated by C. Without loss of generality, we assume every user $u$ has different graph $A_u = \{sk_u, s'_u, \mathcal{G}_{U_u}, \{o_u, i\}, \mathcal{G}_{S_u}\}$. If the $u$-th session of an signing protocol ends successfully, C produces a signature $\sigma_{interm}$ for $A_1$. Their interaction is as follows:

1. $A_1$ concurrently initializes the signing protocol with C by running the zero-knowledge protocol:

$$PK \left\{(i \in V_{U_u}, \varepsilon_{i0}, \varepsilon_{i1}, \varepsilon_{i}), (i, j) \in E_{U_u}, \varepsilon_{i,j0}, \varepsilon_{i,j1}, \varepsilon_{i,j}, \zeta, \rho, \omega, \tau, \gamma) : \right.$$ 

$$e(C_1 C_2 d_u^u u^{\tau}, X_{00}) = e(u^{\rho}, X) \wedge C_1 C_2 = h^{\xi \psi} \prod_{i \in V_{U_u}} C_i^{\varepsilon_{i0}} \prod_{(i,j) \in E_{U_u}} C_i^{\varepsilon_{i,j0}} \wedge$$

$$e \left( \prod_{i \in V_{U_u}} C_i \prod_{(i,j) \in E_{U_u}} C_{i,j}, X_{00} \right) =$$

$$\prod_{i \in V_{U_u}} e \left( W_i^\varepsilon, \prod_{k=0}^{1} X_{0k}^{\varepsilon_{i,k}} \right) \prod_{(i,j) \in E_{U_u}} e \left( W_{i,j}^\varepsilon, \prod_{k=0}^{1} X_{0k}^{\varepsilon_{i,j,k}} \right) \right\}$$

32
Without loss of generality, we assume $A_1$ always execute this protocol honestly. Therefore, $C$ always reset successfully and can extract the secret exponents $\zeta = s_{ku_1}r_{u,1}, \rho = s_u r_{u,1}, \omega = r_{u,1}, \tau = t_u, \gamma = y_u, \varepsilon_i = r_u, i, \varepsilon_0 = r_u, i_0, \varepsilon_1 = r_u, o_{u,i_1}, \varepsilon_{(i,j)} = r_u, (i,j), \varepsilon_{(i,j)0} = r_u, (i,j)i_u, \varepsilon_{(i,j)} = r_u, o_{u,(i,j)}$, used by $A_1$ in the protocol above.

2. Using $t_u = t_u'/y_u$ or $s_u = s_u'/r_u, 1$, $C$ can search in $L_{(O, IS)}$ for the $G_{u} u$, of this intermediary signature. Next, $C$ chooses a random value $s''_u \in \mathbb{Z}_p^*$ and a graph $G_{u} = \{V_{u} u, E_{u} u\}$ to set:

$$v_u = h^s_{ku_1} r_{u,1} \prod_{i \in V_{u} u} \left( \prod_{k=0}^{n} a_{u,i,k} \right)^{o_{u,i_1}} \prod_{(i, j) \in E_{u} u} \left( \prod_{k=0}^{n} m_{u,(i,j),k} \right)^{o_{u,(i,j)1}} \prod_{i \in V_{u} u} \prod_{k=0}^{n} a_{u,i,k} \prod_{(i, j) \in E_{u} u} m_{u,(i,j),k} b_u s''_u r_{u,1} + s''_u d_u$$

where $a_{u,i,k} = g_1^{t_u(x)} t_u, x^k_i$, $b_u = g_1^{s''_u t_u}, d_u = g_1^{s''_u}$, $C$ adds the record $(t_u, s_u = s_u r_{u,1} + s''_u, v_u, A_u, G_{u} u)$ to $L_{(O, IS)}$ and returns $s''_u = (t_u, s''_u, v_u, G_{u} u)$ as the signature to $A_1$.

Since $C$’s choices of $t_u, s''_u$ are independent of $A$’s view, a collision $v_i = v_j$ for some $i, j \leq q$ in $A$’s concurrent queries happens with a negligible probability of $Pr[Col] = 1/p$ in which $A_1$ can compute the discrete logarithm $x$. Else, $C$ simulates the IntermSign oracle perfectly for every concurrent query and $A_1$ can formulate its signature $\sigma_{interm} = (t_u, s_u = s_u r_{u,1} + s''_u, v_u, G_{u} u) = G_{u} u + G_{u} s, \{o_{u,i}, o_{u,(i,j)}\})$ as in the original signing protocol. This gives:

$$Pr[S_3] = Pr[S_2] + Pr[Col]$$

$$\leq Pr[S_2] + \prod_{i=1}^{q-2} i/p$$

$$\leq Pr[S_2] + (q - 2)!/p^{q-2}$$

(4)

where $A_1$ can make at most, another, $q - 2$ intermediary signature queries. The initial signature will not collide with intermediary signature because $C$ can always avoid this by choosing another $s''_u$.

**Game4.** In this game, $A_1$ plays the role of multiple users to concurrently interact with the final signer simulated by $C$. Without loss of generality, we assume every user $u$ has different graph $A_u = \{s_{ku_u}, s''_u, G_{u} u, \{o_{u,i}, o_{u,(i,j)}\}\}$. If the $u$-th session of an signing protocol ends successfully, $C$ produces a graph signature $\sigma$ for $A_1$. Their interaction is as follows:
1. \(A_1\) concurrently initializes the signing protocol with \(C\) by running the zero-knowledge protocol:

\[
\begin{align*}
PK \left\{ (\forall i \in V_u : \varepsilon_{i_0}, \varepsilon_{i_1}, \varepsilon_{i}), \langle \forall (i, j) \in E_u : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \varepsilon_{(i,j)} \rangle, \zeta, \rho, \omega, \tau, \gamma : \\
\left( C_1 C_2 t^{2u'^{\tau}} \cdot X_0 \right) &= \left( C_1 C_2 h^{t^u} \prod_{i \in V_u} C_i^{\varepsilon_{i}} \prod_{(i,j) \in E_u} C_{(i,j)}^{\varepsilon_{(i,j)}} \right) \text{and} \\
\left( \prod_{i \in V_u} \left( W'_i \prod_{k=0}^{1} X_{0k}^{\varepsilon_{i_k}} \right) \prod_{(i,j) \in E_u} \left( W'_{(i,j)} \prod_{k=0}^{1} X_{0k}^{\varepsilon_{(i,j)k}} \right) \right) \right) \\
\end{align*}
\]

Without loss of generality, we assume \(A_1\) always execute this protocol honestly. Therefore, \(C\) always reset successfully and can extract the secret exponents \(\zeta = sk_{x_0}, u, \rho = s'_{y_0}, \omega = r_{u,2}, \tau = t'_u, \gamma = y_u, \varepsilon_{i_0} = r_{u,i}, \varepsilon_{i_1} = r_{u,i}, \varepsilon_{i_2} = r_{u,2}\varepsilon_{(i,j)_1} = r_{u,(i,j)}, \varepsilon_{(i,j)_0} = r_{u,(i,j)}\varepsilon_{(i,j)} = r_{u,2}\varepsilon_{(i,j)}\) used by \(A_1\) in the protocol above.

2. Using \(t_u = t'_u/y_u\) or \(s_u = s''_{u}/r_{u,2}\), \(C\) can search in \(L_{(IOL,iS)}\) for \(\{G_u, a_{u,1}, u_{(i,j)}\}\) of this final signature. Next, \(C\) chooses a random value \(s''_{u} \in \mathbb{Z}_p^+\) to set:

\[
\begin{align*}
v_u &= b_u^{sk_{x_0}r_{u,1}} \prod_{i \in V_u} \left( \prod_{k=0}^{n} a_{u,i_k}^{m_{u,i_k}} \right)^{\Theta_{u,i_1} \Theta_{u,i_2}} \prod_{(i,j) \in E_u} \left( \prod_{k=0}^{n} a_{u,(i,j)_k}^{m_{u,(i,j)_k}} \right)^{\Theta_{u,(i,j)_1} \Theta_{u,(i,j)_2}}, \\
&= b_u^{s''_{y_0} + s''_{u} c_u}, \\
\end{align*}
\]

where \(a_{u,i_k} = b_{u}^{f_{x_0}(x) t_u \cdot x} \cdot b_{u}^{f_{x_0}(x) t_u} \cdot c_u = g_{1}^{f_{x_0}(x) t_u}\). If \((t_u, s_u, v_u, A_u) \in L_{(PV)}\), \(C\) removes it from \(L_{(PV)}\). \(C\) adds the record to \(L_{(FO,FS)}\). \(C\) returns \(\sigma_u = (t_u, s''_{u}, v_u)\) as the signature to \(A_1\).

Since \(C\)'s choices of \(t_u, s''_{u}\) are independent of \(A\)'s view, a collision \(v_i = v_j\) for some \(i, j \leq q\) in \(A\)'s concurrent queries happens with a negligible probability of \(Pr[Col] = 1/p\) in which \(A_1\) can compute the discrete logarithm \(x\). Else, \(C\) simulates the FinalSign oracle perfectly for every concurrent query and \(A_1\) can formulate its signature \(\sigma_u = (t_u, s_u = s''_{u} r_{u,2} + s''_{u}, v_u, G_u, \{a_{u,i}, a_{u,i_2}, a_{u,(i,j)}, a_{u,(i,j)_2}\}, \{a_{u,i_1}, a_{u,(i,j)}\}) as in the original signing protocol. This gives:

\[
\begin{align*}
Pr[S_4] &= Pr[S_3] + Pr[Col] \\
&\leq Pr[S_3] + \sum_{i=1}^{q-2} i/p \\
&\leq Pr[S_3] + (q-2)!/p^{q-2}.
\end{align*}
\]
where \( \mathcal{A}_1 \) can make, at most, another \( q - 2 \) final signature queries. The final signature will not collide with the initial signature because they use different bases \( c \) and \( d \), respectively.

**Game5.** In this game, \( \mathcal{A}_1 \) plays the role of multiple provers to concurrently interact with the verifier simulated by \( \mathcal{C} \). Without loss of generality, we assume every prover \( u \) uses a valid \( \sigma_u \) to run its show proof on \( \phi_{\text{stmt}} \) such that \( \phi_{\text{stmt}}(A_u) = 1 \). \( \mathcal{C} \) always simulates the \( \text{Verify} \) oracle correctly and this gives:

\[
\Pr[S_u] = \Pr[S_4].
\]  

(6)

**Game6.** In this game, \( \mathcal{A}_1 \) plays the role of verifier to concurrently interact with multiple provers simulated by \( \mathcal{C} \). When \( \mathcal{A}_1 \) asks for a show proof on \( \phi_{\text{stmt}} \), \( \mathcal{C} \) interacts with \( \mathcal{A}_1 \) using a \( \sigma_u \) such that \( \phi_{\text{stmt}}(A_u) = 1 \). We assume \( \mathcal{C} \) already has the appropriate signatures on his hand for these queries. Else, \( \mathcal{C} \) simulates a valid \( \sigma_u \) as in the previous games and adds it to \( L_{(P,V)} \) before interacting with \( \mathcal{A}_1 \). This gives:

\[
\Pr[S_u] = \Pr[S_5].
\]  

(7)

**Game7.** In this game, \( \mathcal{A}_1 \) wants to impersonate the prover whose data set is \( A^* \neq A_i \in L_{(FO,FS)} \) using the predicate \( \phi^*_{\text{stmt}} \) such that \( \phi^*_{\text{stmt}}(A^*) = 1 \) and \( \phi^*_{\text{stmt}}(A_i) = 0 \). \( \mathcal{A}_1 \) is still allowed to query the oracles as in previous games but with the restriction \( \phi^*_{\text{stmt}}(A_i) \neq 1 \) for \( A_i \) to the \( \text{Obtain} \) oracle. Finally, if \( \mathcal{A}_1 \) completes a show proof for \( A^* \) such that \( (A^1_{\text{Proof}}(pk, \cdot, \phi^*_{\text{stmt}}(A^*)), \mathcal{C}^{\text{Verify}}(pk, \phi^*_{\text{stmt}}(A^*))) = 1 \), \( \mathcal{C} \) resets \( \mathcal{A}_1 \) to the time where it has just sent the witnesses. If the show is proof verified again, \( \mathcal{C} \) can obtain two valid transcripts and recover the secret exponents to extract the signature elements \( (t^*, s^*, v^*) \).

Since \( \mathcal{A}_1 \) must output \( t^* \notin \{t_1, \ldots, t_q\} \), if \( v^* \notin L_{(FO,FS)} \cup L_{(P,V)} \), \( \mathcal{C} \) can construct a polynomial \( c(x) \) of degree \( n - 1 \) such that \( f(x) = c(x)(x + t^*) + r \) to compute:

\[
v^{*1/((t_h + b_h, t_i + \sum_{i=1}^n m_i^h x_i^{(h)}) \sum_{(l,j) \in E} (t_{(l,j)} + \sum_{k=0}^n m_{(l,j)}^h x_i^{(h)}) + t_h s^* + t_c))} - \frac{c(x)}{g_1}\]

\[
= g_1 \frac{c(x)(x+t^*)+r - c(x)}{v^{*1/(t_h + b_h, t_i + \sum_{i=1}^n m_i^h x_i^{(h)}) \sum_{(l,j) \in E} (t_{(l,j)} + \sum_{k=0}^n m_{(l,j)}^h x_i^{(h)}) + t_h s^* + t_c)} - \frac{c(x)}{g_1}
\]

\[
= g_1 \frac{1}{v^{*1/(t_h + b_h, t_i + \sum_{i=1}^n m_i^h x_i^{(h)}) \sum_{(l,j) \in E} (t_{(l,j)} + \sum_{k=0}^n m_{(l,j)}^h x_i^{(h)}) + t_h s^* + t_c) - v^{*1/(t_h + b_h, t_i + \sum_{i=1}^n m_i^h x_i^{(h)}) \sum_{(l,j) \in E} (t_{(l,j)} + \sum_{k=0}^n m_{(l,j)}^h x_i^{(h)}) + t_h s^* + t_c)}}
\]

and output \( (g^{v^{*1/(t_h + b_h, t_i + \sum_{i=1}^n m_i^h x_i^{(h)}) \sum_{(l,j) \in E} (t_{(l,j)} + \sum_{k=0}^n m_{(l,j)}^h x_i^{(h)}) + t_h s^* + t_c)}, t^*) \) as the solution for the SDH instance. On the other hand, if we have \( v^* \in L_{(FO,FS)} \cup L_{(P,V)} \), \( \mathcal{C} \) can extract the discrete logarithm \( x \) to break the SDH assumption.

Let \( \Pr[\text{Acc}] \) be the probability of \( \mathcal{C} \) outputs 1 in the showing protocol with \( \mathcal{A}_1 \), and \( \Pr[\text{Res}] \) the probability of \( \mathcal{C} \) resets successfully, by Multi-Instance
Reset Lemma [21], we have:

\[
\Pr[S_7] \leq \Pr[S_6] + \Pr[Acc]
\leq \Pr[S_6] + \sqrt{\Pr[Res]} - 1 + 1/p + 1
\leq \Pr[S_6] + \sqrt{\varepsilon_{sdh}} - 1 + 1/p + 1.
\] (8)

Summing up the probability from (1) to (8), we have \( \varepsilon_{\text{imp}} \leq \sqrt{\varepsilon_{sdh}} - 1 + 1/p + 1 + (q-1)/p^{q-1} \) as required. The time taken by \( C \) is at least \( 2NT_{\text{imp}} \) 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 6.** If an adversary \( A_2 \) \( (t_{\text{imp}}, \varepsilon_{\text{imp}}) \)-breaks the \( \text{imp-aca} \)-security of the proposed graph signature scheme, then there exists an algorithm \( C \) which \( (t_{\text{sdh}}, \varepsilon_{\text{sdh}}) \)-solves the SDH problem such that:

\[
\varepsilon_{\text{imp}} \leq \sqrt{\varepsilon_{sdh}} - 1 + \frac{1+(q-1)/p^{q-2}}{p} + 1,
\]

\[
t_{\text{imp}} \leq t_{\text{sdh}}/2N - T(q^2).
\]

where \( N \) is the total of adversary instances, \( q = Q_{(O,S)} + Q_{(P,V)} \) is the number of queries made to the Obtain and Verify oracles, while \( T(q^2) \) is the time parameterized by \( q \) to setup the simulation environment and to extract the SDH solution.

**Proof.** Given a \( q \)-SDH instance \((g_1, g_1^2, g_1^3, \ldots, g_1^q, g_2, g_2^2)\) where \( q = Q_{(O,S)} + Q_{(P,V)} \) is the maximum number of queries \( A_2 \) can issue to the Obtain and Verify oracles, there exists an algorithm \( C \) which can output \((g_{x+t}, t)\) by acting as the simulator for the graph signature scheme as follows:

**Game\textsubscript{0}.** This is the same as the **Game\textsubscript{0}** in Lemma 5 where we have:

\[
\Pr[S_0] = \varepsilon_{\text{imp}}.
\] (9)

**Game\textsubscript{1}.** This is the same as the **Game\textsubscript{1}** in Lemma 5 except that \( C \) additionally checks whether \( X = d_q^{x_0} \) for \( u \in \{1, \ldots, q\} \). If such \( t_u \) is found, \( C \) outputs the solution of the SDH instance using the discrete logarithm \( x = t_u \). Let \( f_{u', u}(x) = \prod_{k=1, k \neq u',u}^{O} (x + t_k) = \sum_{k=0}^{O} \gamma_k x^k \). \( C \) uniformly selects random distinct \( s_1, \ldots, s_q \in \mathbb{Z}_{p}^* \) and sends \((e, \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T, p, \{a_{0_k} = g_1^{(x_0)x_0^k}, X_{0_k} = g_2^{x_0^k}\}) \) to \( A_2 \). This gives:

\[
\Pr[S_1] \leq \Pr[S_0].
\] (10)

**Game\textsubscript{2}.** This is the same as the **Game\textsubscript{2}** in Lemma 5 except that, after resetting \( A_2 \), \( C \) simulates the signature \( \sigma'_u = (t_u, s''_u, v_u) \) for \( A_u = \{sk_u, s''_u, \perp, \perp\} \) and
\[ \mathcal{G}_{S_u} = \{V_{S_u} \cup E_{S_u}\} \] such that:

\[
v_i = h_u^{sk_{U,u}r_u,1} \prod_{i \in V_{S_u}} \left( \prod_{k=0}^{n} a_{u,i,k}^{m_u,i_k} \right)^{a_{u,i,1}} \prod_{(i,j) \in E_{S_u}} \left( \prod_{k=0}^{n} a_{u,(i,j),k}^{m_u,(i,j)_k} \right)^{a_{u,(i,j),1}} b_u^{s_u' + (s_u - s_u') c_u} d_u
\]

where \( s_u' = s_u - s_u' \). When the protocol ends, \( A_2 \) can compile the intermediary signature as \( \sigma_{init_u} = (t_u, s_u = s_u' + s_u', v_u, \mathcal{G}_{U_u} = \mathcal{G}_{S_u}) \). As \( C \) simulates the \textit{InitSign} oracle perfectly, we have:

\[
\Pr[S_2] \leq \Pr[S_1] + (q - 1)/p^{q-1}.
\]

where \( A_1 \) can make, at most, another \( q - 1 \) initial signature queries.

**Game3.** This is the same as the \textbf{Game3} in Lemma 5 except that, after resetting \( A_2, C \) simulates the signature \( \sigma_u' = (t_u, s_u, v_u) \) for \( A_u = \{sk_{U,u}, s_u', \mathcal{G}_{U_u}, \{o_u,i, o_{u,(i,j)}\}\} \) and \( \mathcal{G}_{S_u} \) such that:

\[
v_i = h_u^{sk_{U,u}r_u,1} \prod_{i \in V_{S_u}} \left( \prod_{k=0}^{n} a_{u,i,k}^{m_u,i_k} \right)^{a_{u,i,1}} \prod_{(i,j) \in E_{S_u}} \left( \prod_{k=0}^{n} a_{u,(i,j),k}^{m_u,(i,j)_k} \right)^{a_{u,(i,j),1}} b_u^{s_u' + (s_u - s_u') d_u}
\]

where \( s_u' = s_u - s_u' \). When the protocol ends, \( A_2 \) can compile the intermediary signature as \( \sigma_{inter_u} = (t_u, s_u = s_u' + s_u', v_u, \mathcal{G}_{U_u}, \{o_u,i, o_{u,(i,j)}\}, \mathcal{G}_{S_u}) \). As \( C \) simulates the \textit{IntermSign} oracle perfectly, we have:

\[
\Pr[S_3] \leq \Pr[S_2] + (q - 2)/p^{q-2}.
\]

where \( A_1 \) can make, at most, another \( q - 2 \) intermediary signature queries.

**Game4.** This is the same as the \textbf{Game4} in Lemma 5 except that, after resetting \( A_2, C \) simulates the signature \( \sigma_u' = (t_u, s_u, v_u) \) for \( A_u = \{sk_{U,u}, s_u', \mathcal{G}_{U_u}, \{o_u,i, o_{u,(i,j)}\}\} \) such that:

\[
v_i = h_u^{sk_{U,u}r_u,1} \prod_{i \in V_{S_u}} \left( \prod_{k=0}^{n} a_{u,i,k}^{m_u,i_k} \right)^{a_{u,i,1}a_{u,i,2}} \prod_{(i,j) \in E_{S_u}} \left( \prod_{k=0}^{n} a_{u,(i,j),k}^{m_u,(i,j)_k} \right)^{a_{u,(i,j),1}a_{u,(i,j),2}} b_u^{s_u' + (s_u - s_u') c_u}
\]

where \( s_u' = s_u - s_u' \). When the protocol ends, \( A_2 \) can compile the final signature as \( \sigma_u = (t_u, s_u = s_u' + s_u', v_u, \mathcal{G}_{U_u}, \{o_u,i, o_{u,(i,j)}\}) \). As \( C \) simulates the \textit{FinalSign} oracle perfectly, we have:

\[
\Pr[S_4] \leq \Pr[S_3] + (q - 2)/p^{q-2}.
\]
where $A_1$ can make, at most, another $q - 2$ final signature queries.

**Game5.** This is the same as the Game$_5$ in Lemma 5 and we have:

$$\Pr[S_5] = \Pr[S_4].$$  \hspace{1cm} (14)

**Game6.** This is the same as the Game$_6$ in Lemma 5 and we have:

$$\Pr[S_6] = \Pr[S_5].$$  \hspace{1cm} (15)

**Game7.** Similar to the Game$_7$ in Lemma 5, $C$ can reset $A_2$ to extract the elements $(t^*, s^*, v^*)$ of $\sigma^*$ where $v^*$ has the form:

$$v^* = \left( f(x)(t_h k u^r + \sum_{i \in V} (a_i t_{u i} \sum_{k=0}^n m_i^k x^{t k}) + \sum_{(i, j) \in E} (a_i^j t_{u (i, j)} \sum_{k=0}^n m_{i (i, j)}^k x^{t k}) + s^* t_h + t_e) \right) \cdot \left( 1/(x + t_u) \right).$$

Since $A_2$ must output $t^* = t_u \in \{t_1, \ldots, t_q\}$ but $s^* \neq s_u \in \{s_1, \ldots, s_q\}$ for an $u \in \{1, \ldots, q\}$, $C$ proceeds to compute $c(x)$ of degree $q - 2$ and $r \in \mathbb{Z}_n^*$ 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 $v^* \notin L_{(F_0, S)} \cup L_{(P, V)}$ or $C$ already found $x = t_u$ during Game$_1$. Subsequently, $C$ calculates:

$$\frac{\sum_{u' = 1, u' \neq u}^n (s_{u'} - s^*)(f_u(x)(t_h k u^r + \sum_{i \in V} (a_i t_{u i} \sum_{k=0}^n m_i^k x^{t k}) + s^* t_h + t_e) - (f_u(x) - c(x)(x + t_u))^r(s_{u'} - s^*)}{g_1(t_h k u^r + \sum_{i \in V} (a_i t_{u i} \sum_{k=0}^n m_i^k x^{t k}) + s^* t_h + t_e)} = g_1 = g_1^{1/t_u}$$

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

$$\Pr[S_5] \leq \Pr[S_4] + \sqrt{\sqrt{q} - 1 + 1/p + 1}$$

and summing up the probability from (10) to (16), we have $\varepsilon_{imp} \leq \sqrt{\sqrt{q} - 1 + 1/p + 1 + (q - 1)!/p^{q - 2}}$ as required. The time taken by $C$ is at least $2Nt_{imp}$ 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 7.** If an adversary $A_3$ ($t_{imp}, \varepsilon_{imp}$)-breaks the imp–aca–security of the proposed graph signature system, then there exists an algorithm $C$ which $(t_{sdh}, \varepsilon_{sdh})$-solves the SDH problem such that:

$$\varepsilon_{imp} \leq \frac{\y}{\sqrt{\sqrt{q} - 1 + 1/p + 1 + (q - 1)!/p^{q - 2}}} + 1,$$

$$t_{imp} \leq t_{sdh}/2N - T(q^2).$$

38
where $N$ is the total of adversary instances, $q = Q_{(O,S)} + Q_{(P,V)}$ is the number of queries made to the Obtain and Verify oracles, while $T(q^2)$ is the time parameterized by $q$ to setup the simulation environment and to extract the SDH solution.

**Proof.** Given a $q$-SDH instance $(g_1, g_1^x, g_1^{x^2}, \ldots, g_1^{x^q}, g_2, g_2^z)$ where $q = Q_{(O,S)} + Q_{(P,V)}$ is the maximum number of queries $A_3$ can make to the Obtain and Verify oracles, there exists an algorithm $C$ which can output $(g_1^{\frac{x}{q^2}}, t)$ by acting as the simulator for the graph signature scheme as follows:

**Game**0. This is the same as the **Game**0 in Lemma 5 and we have:

$$\Pr[S_0] = \varepsilon_{imp}. \quad (17)$$

**Game**1. The precomputations and checking are the same as the **Game**1 in Lemma 6 but $(e, G_1, G_2, G_T, p, \{a_{0k} = g_1^{f(x)k} - \sum_{i=1}^{n} f_i(x), X_0 = g_2^z\})_{k=0}^{n}, \{a_{ik}, X_{ik} = g_1^{f(x)k} - \sum_{i=1}^{n} f_i(x)\}, c = g_1^{f(x) + \sum_{i=1}^{q} z_i f_i(x)}, X = g_2^z, X_0 = g_2, X_1 = X_0^{z_1}, \ldots, X_n = X_0^{z_n}$ as the public key to $A_3$ where the random $z_1, \ldots, z_q \in \mathbb{Z}_p^m$ are uniformly distributed. This gives:

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

**Game**2. This is the same as the **Game**2 in Lemma 5 except that, after resetting $A_3, C$ simulates the signature $\sigma_{s}^{t} = (t_u, s'_u, v_u)$ for $A_u = \{sk_{U_u}, s'_u, \perp, \perp\}$ and $\mathcal{G}_{S_u} = \{V_{S_u} \cup E_{S_u}\}$ by letting

$$s_u = z_u - \prod_{i \in V_{S_u}} (x' + i) \prod_{w \in f_V(i)} (x' + w) - \prod_{(i,j) \in E_{S_u}} (x' + i)(x' + j) \prod_{w \in f_E(i,j)} (x' + w),$$

where:

$$v_i = g_1^{f(x)(t_{u}, sk_{U_u} + \sum_{i \in V} (t_{u} - \sum_{k=0}^{n} m_{u,i} x^{ik}) + \sum_{(i,j) \in E} (t_{u, (i,j)} - \sum_{k=0}^{n} m_{u, (i,j)k} x^{ik}) + s_{u}, x_{i,j})} \cdot g_1^{f(x) \cdot \sum_{u', u'' \neq u} (z_{u'} - z_{u''}) f_{u', u''}(x)}$$

$$= h_{u, sk_{U_u}, r_{u, 1}} \prod_{i \in V_{S_u}} \prod_{k=0}^{n} a_{u, i, k} \prod_{(i,j) \in E_{S_u}} \prod_{k=0}^{n} a_{u, (i,j), k} b_{u, (i,j), k} b_{u, s_u^{u''} + (s_u - s'_u), a_{u, (i,j), k}}$$

where $s'_u = s_u - s''_u$. When the protocol ends, $A_3$ compiles the signature as $\sigma_{\text{init}_u}(s_u) = (t_u, s_u = s'_u + s''_u, v_u, \mathcal{G}_{U_u} = \mathcal{G}_{S_u})$. As $C$ simulates the InitSign oracle perfectly, we have:

$$\Pr[S_2] \leq \Pr[S_1] + (q - 1)!/p^{q-1}. \quad (19)$$

where $A_1$ can make, at most, another $q - 1$ initial signature queries.
Game3. This is the same as the Game3 in Lemma 5 except that, after resetting 
\(A_3\), \(C\) simulates the signature \(\sigma'_u = (t_u, s''_u, v_u)\) for \(A_u = \{sk_u, s'_u, \mathcal{G}_u, \{o_{u,i}, o_{u,(i,j)}\}\}\) and \(\mathcal{G}_{S_u}\) by letting 
\[
    s_u = z_u - \prod_{i \in V_{0_u}} a_{u,i_1}(x' + i) \prod_{w \in f_{V}(i)} (x' + w) - \prod_{(i,j) \in E_{0_u}} a_{u,(i,j)}(x'+i)(x'+j) \prod_{w \in f_{V}(i)} (x' + w) - \prod_{i \in V_{5_u}} (x' + i) \prod_{w \in f_{V}(i)} (x' + w) - \prod_{(i,j) \in E_{5_u}} (x'+i)(x'+j) \prod_{w \in f_{V}(i)} (x' + w)
\]
where:
\[
    v_i = h_u^{sk_u r_u, 2} \prod_{i \in V_{0_u}} \left( \prod_{k=0}^{n} a_{u,i_k} \right)^{o_{u,i_1}} \prod_{(i,j) \in E_{0_u}} \left( \prod_{k=0}^{n} a_{u,(i,j)_k} \right)^{o_{u,(i,j)_1}} \prod_{i \in V_{5_u}} \left( \prod_{k=0}^{n} a_{u,i_k} \right) b_u^{s'_u + (s_u - s'_u)} d_u
\]
and \(s''_u = s_u - s'_u\). When the protocol ends, \(A_3\) compiles the signature as 
\(\sigma'_{\text{Interm}u} = (t_u, s'_u + s''_u, v_u, \mathcal{G}_u, \{o_{u,i}, o_{u,(i,j)}\}, \mathcal{G}_{S_u})\). As \(C\) simulates the IntermSign oracle perfectly, we have:
\[
    \Pr[S_3] \leq \Pr[S_2] + (q-2)!/p^{q-2}.
\]
where \(A_1\) can make, at most, another \(q-2\) intermediary signature queries.

Game4. This is the same as the Game4 in Lemma 5 except that, after resetting 
\(A_3\), \(C\) simulates the signature \(\sigma'_u = (t_u, s''_u, v_u)\) for \(\{sk_u, s'_u, \mathcal{G}_u, \{o_{u,i}, o_{u,(i,j)}\}\}\) by letting 
\[
    s_u = z_u - \prod_{i \in V_{0_u}} o_{u,i_1} o_{u,i_2}(x' + i) \prod_{w \in f_{V}(i)} (x' + w) - \prod_{(i,j) \in E_{0_u}} o_{u,(i,j)_1} o_{u,(i,j)_2}(x'+i)(x'+j) \prod_{w \in f_{V}(i)} (x' + w) - \prod_{i \in V_{5_u}} o_{u,i_1} (x' + i) \prod_{w \in f_{V}(i)} (x' + w) - \prod_{(i,j) \in E_{5_u}} o_{u,(i,j)_1} (x'+i)(x'+j) \prod_{w \in f_{V}(i)} (x' + w)
\]
where:
\[
    v_i = h_u^{sk_u r_u, 2} \prod_{i \in V_{0_u}} \left( \prod_{k=0}^{n} a_{u,i_k} \right)^{o_{u,i_1} o_{u,i_2}} \prod_{(i,j) \in E_{0_u}} \left( \prod_{k=0}^{n} a_{u,(i,j)_k} \right)^{o_{u,(i,j)_1} o_{u,(i,j)_2}} \prod_{i \in V_{5_u}} \left( \prod_{k=0}^{n} a_{u,i_k} \right) b_u^{s'_u + (s_u - s'_u)} d_u
\]
and \( s''_u = s_u - s'\_u \). When the protocol ends, \( \mathcal{A}_3 \) compiles the signature as \( \sigma'_u = (t_u, s_u = s'_u + s''_u, v_u, G_u, \{ o_{u,i}, o_{u,(i,j)} \} ) \). As \( \mathcal{C} \) simulates the FinalSign oracle perfectly, we have:

\[
\Pr[S_4] \leq \Pr[S_3] + (q - 2)!/p^{q - 2}.
\] (21)

where \( \mathcal{A}_1 \) can make, at most, another \( q - 2 \) final signature queries.

**Game\(_5\).** This is the same as the Game\(_5\) in Lemma 5 and we have:

\[
\Pr[S_4] = \Pr[S_4].
\] (22)

**Game\(_6\).** This is the same as the Game\(_6\) in Lemma 5 and we have:

\[
\Pr[S_4] = \Pr[S_4].
\] (23)

**Game\(_7\).** 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 \( v^* \notin L_{(F,O,FS)} \cup L_{P,V} \) or \( x = t_u \) has been found during **Game\(_1\)**. In the unlikely case of Type 16 forgery \( A^* \in L_{P,V} \) which happens with probability \( 1/p \), \( \mathcal{C} \) aborts. Similar to the **Game\(_7\)** in Lemma 5, \( \mathcal{C} \) can reset \( \mathcal{A}_3 \) to extract the signature elements \( (t^*, s^*, v^*) \). \( \mathcal{C} \) 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:

\[
\begin{align*}
\left( v^*/g_1 \right) &\cdot f_u(x)(t_u s_u v^* + \sum_{k=0}^{n} \sum_{l=0}^{n} m^*_{u,k} z^k) + \sum_{(i,j) \in B}(t_{u(i,j)} \sum_{k=0}^{n} m^*_{i,j,k} x^k) z^k t_u + t_u, \\
\sum_{j \in V}(t_{u(j)} \sum_{k=0}^{n} m^*_{i,j,k} x^k) z^k t_u + t_u, \\
\frac{f_u(x)(x + t_u) + z^*}{(x + t_u)(x^* + z^*)} &\cdot \frac{1}{g_1} \\
&= g_1^{-1} t_u,
\end{align*}
\]

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

\[
\Pr[S_7] \leq \Pr[S_6] + \frac{\sqrt{\varepsilon_{sdh}}}{1 + 1}
\] (24)

and summing up the probability from (17) to (24), we have \( \varepsilon_{imp} \leq \frac{\sqrt{\varepsilon_{sdh}}}{1 + 1} + (q - 1)!/p^{q - 1} \) as required. The time taken by \( \mathcal{C} \) is at least \( 2Nt_{imp} \) 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 2, Lemmas 5, 6, and 7 gives Theorem 5 as required.
6.4.2 Unlinkability. Next, we prove the unlinkability of the proposed graph signature scheme. It is sufficient to show that the witnesses, the committed graph 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 \( A \) is given access to the \( \text{Obtain}, \text{Prove}, \text{Verify} \) and \( \text{Corrupt} \) oracles, it does not has advantage in guessing the challenged graphs.

**Lemma 8.** The initialization of the signing protocol in the graph signature scheme has random self-reducibility.

**Proof.** We use initial signing protocol as an example. Let \( \text{Gen} = \text{KeyGen}, P = \text{user}, V = \text{signer}, pk = C \) and \( sk = (sk_u, s') \), we define the algorithms \( \text{Rerand}, \text{Derand} \) and \( \text{Tran} \) as follows:

- \( \text{Rerand}(C) \) randomly selects \( \rho \in \mathbb{Z}_p^* \) and outputs \( C' = C^\rho \) where \( C = h^{sk_u}b^{s'} \) is the commitment generated by \( \text{user} \). For all \( (C, sk_u, s'), (C', sk'_u, s'') \) has the same uniform distribution as another \( (C'', sk''_u, s''') \) which would have been generated by \( \text{user} \).

- \( \text{Derand}(C, C', sk'_u, s'', \rho) \) outputs \( (sk_u, s') = \{sk'_u / \rho\}, s'' / \rho \) for all \( (C', \rho) \in \text{Rerand}(C) \).

- \( \text{Tran}(C, C', \rho, \tilde{C}', e', \{\hat{sk}_u\}', \tilde{s}'') \) outputs \( \tilde{C} = \tilde{C}' / \rho, \{\hat{sk}_u\}' = \{sk'_u / \rho\}, \tilde{s}' = \tilde{s}'' / \rho \) for all \( (C', \rho) \in \text{Rerand}(C) \). The transcript \( (\tilde{C}, e', \{\hat{sk}_u\}', \tilde{s}') \) is valid wrt. \( C \) if \( (\tilde{C}', e', \{\hat{sk}_u\}', \tilde{s}'') \) is valid wrt. \( C' \).

\( \square \)

**Lemma 9.** The blinded graph signature is perfectly hiding.

**Proof.** This has been proven in the previous work [29].

\( \square \)

**Lemma 10.** The showing protocol of the graph signature scheme offers random self-reducibility.

**Proof.** The proof is similar to that of Lemma 8 as well as to the previous work [29].

\( \square \)

**Theorem 6.** If the signing protocols are performed in secure channels and the showing protocols have random self-reducibility, and their witnesses, committed graph as well as the blinded graph signature are perfectly hiding, the graph signature scheme is gunl-aca-secure.

**Proof.** We show that an adversary \( A \) can win the gunl-aca-security game only with a negligible advantage \( \varepsilon_{\text{gunl}} \), with respect to the graph signature scheme simulator \( C \).

\( \textbf{Game}_0 \). This is an attack on the original graph signature scheme. Let \( S_0 \) denotes a successful distinguishing attempt, by definition we have:

\[
\Pr[S_0] \leq \varepsilon_{\text{gunl}} + \frac{1}{2}
\]  

(25)
Game1. \( C \) generates \((pk, sk)\) as in the original algorithm and forwards to \( A \) so that the latter can play the role of user and signer. In addition, \( C \) maintains four lists \( L_{(iO,iS)}, L_{(IO,IS)} \), \( L_{(FO,FS)} \), \( L_{(PV)} \) for corrupted signing protocols and showing protocols, respectively. Since \( C \) does not alter the key generation algorithm, this gives:

\[
Pr[S_1] = Pr[S_0].
\] (26)

Game2. When \( A \) act as the user to concurrently interact with signers, \( C \) simulates \texttt{Obtain} oracle to produce the corresponding signature \( \sigma_i \) for the user in the \( i \)-th session, as well as adding the related information into the corresponding lists \( L_{(iO,iS)}, L_{(IO,IS)} \). Without lost of generality, we assume every user uses different data set \( A_i \). Since our graph signature works in the secure channel, it is clear that every transcript is well hidden. If it is the hybrid ElGamal encryption that is hiding the transcripts, its ciphertext indistinguishability security also hides the transcripts from the adversary. Moreover, every protocol session is uniformly distributed by Lemma 8. This gives:

\[
Pr[S_2] = Pr[S_1].
\] (27)

Game3. Comparing to the previous games, \( A \) additionally queries the signing protocol transcript of the \( i \)-th session to the Corrupt oracle. \( C \) searches in \( L_{(iO,iS)}, L_{(IO,IS)}, L_{(FO,FS)} \) to return the internal state and the random exponents used in completing the protocol. By Lemma 8, for any two signing transcripts returned by Corrupt, the distribution of their transcripts are identical to each other from the view of \( A \). Following Lemma 11 to Lemma 16, this is true even for the non-uniformly distributed graphs \( G_i \) which have been hidden by \( o_i \) and \( o_{(i,j)} \). Since \( A \) does not gain any advantage, we have:

\[
Pr[S_3] = Pr[S_2].
\] (28)

Game4. Now \( A \) also acts as the verifier to concurrently interact with \( C \) as the provers for multiple signatures. \( C \) runs the \( i \)-th session of a showing protocol for \( \sigma_i = (t_i, s_i, v_i, G_i, \{o_i, o_{(i,j)}\}) \). Without loss of generality, we assume \( A \) always requests for successful show proofs where \( \phi_{\text{stmt}}(A_i) = 1 \).

The interaction is the same as in the original show proof from the view of \( A \). Moreover, Lemma 9 shows that the blinded signature is perfectly hiding and Lemma 10 indicates every protocol session is uniformly distributed. This gives:

\[
Pr[S_4] = Pr[S_3].
\] (29)

Game5. In contrast to the previous games, \( A \) also queries the showing protocol transcript of the \( i \)-th session to the Corrupt oracle. \( C \) searches in \( L_{(PV)} \) to return the internal state and the random exponents used in completing the protocol. By Lemma 10, for any two witness sets in a showing protocol returned by Corrupt, the distribution of their transcripts are identical from the view of \( A \). Following Lemma 9, this is true even if \( A \) knows some final signatures \( \sigma \) from the signing
protocols, which now have been perfectly hidden by \( (r_i, y_i) \). \( A \) can also act as a prover in which it does not gain useful information. We have:

\[
Pr[S_5] = Pr[S_4]
\]  

(30)

where \( A \) does not gain any advantage from the query.

**Game_6.** When \( A \) decides two data sets \( A_0 \) and \( A_1 \) as well as the predicate \( \phi_\text{stmt} \) which he wishes to challenge such that \( \phi_\text{stmt}(A_0) = \phi_\text{stmt}(A_1) = 1 \), \( C \) randomly decides a bit \( b \in \{0,1\} \) and play the user role to run the challenge signing protocols with \( A \) for \( A_b \) and \( A_{1-b} \), respectively. When the final signing protocols are completed, \( C \) obtains two signatures \( \sigma_b \) and \( \sigma_{1-b} \). In the same order, \( C \) uses \( \sigma_b \) and \( \sigma_{1-b} \) to complete the challenge show proof with \( A \) as the verifier. \( A \) can request polynomially many times of show proofs. From time to time, \( A \) still can query the oracles as before with the restriction of querying the challenge transcripts to \text{Corrupt}. Finally, if \( A \) makes a correct guess \( b' = b \), it breaks the graph unlinkability with the probability:

\[
Pr[S_6] = Pr[S_5] = Pr[b' = b] = \frac{1}{2} + \varepsilon_{\text{gunl}}.
\]  

(31)

Combining the probability from equation (25) to (31), we have a negligible \( \varepsilon_{\text{gunl}} \) as required and \( A \) runs in time \( t_{\text{gunl}} \).

Using the similar approach, we show that the security of protocol unlinkability also holds for the proposed graph signature scheme.

**Theorem 7.** If the showing protocols have random self-reducibility, and their witnesses, committed graph as well as the blinded graph signature are perfectly hiding, the graph signature scheme is punl-aca-secure.

**Proof.** The proof is the same as that of Theorem 6 except **Game_6**:

**Game_6.** When \( A \) decides two graphs \( A_0 \) and \( A_1 \) as well as the access policy \( \phi_\text{stmt} \) which he wishes to challenge such that \( \phi_\text{stmt}(A_0) = \phi_\text{stmt}(A_1) = 1 \), \( C \) randomly decides a bit \( b_1 \in \{0,1\} \) and play the user role to run the challenge signing protocols with \( A \) for \( A_{b_1} \) and \( A_{1-b_1} \), respectively. When the final signing protocols are completed, \( C \) obtains two signatures \( \sigma_{b_1} \) and \( \sigma_{1-b_1} \). \( C \) randomly decides another bit \( b_2 \in \{0,1\} \) and uses \( \sigma_{b_2} \) and \( \sigma_{1-b_2} \) to complete the challenge showing protocol with \( A \) as the verifier. \( A \) can request polynomially many times of show proofs. From time to time, \( A \) still can query the oracles as before with the restriction of querying the challenge transcripts to \text{Corrupt}. Finally, if \( A \) makes a correct guess \( (\pi_{(O,I)}, \pi_{(P,V)}) \) such that \( \sigma_{\pi_{(O,I)}} = \sigma_{\pi_{(P,V)}} \), it breaks the
full protocol unlinkability of the graph signature scheme with the probability:
\[
\Pr[S_6] = \Pr[S_5] \\
= \Pr[\sigma_{\pi(O,I)} = \sigma_{\pi(P,V)}] \\
= \frac{1}{2} + \varepsilon_{\text{punl}}.
\] (32)

Therefore, we have a negligible \(\varepsilon_{\text{punl}}\) as required and \(\mathcal{A}\) runs in time \(t_{\text{punl}}\). □

7 An Efficient Proof System on Graph Predicates

7.1 Proof of Knowledge Predicates

Our proposed new graph signature scheme can support all predicate statements offered by Groß' graph signature [18, 19]. For instance, Groß' partition statement is implicitly realized by our vertices statement. The disjoint statement in Groß' scheme proves the pair-wise differences of vertices and edges in a graph but this has been indirectly done by our vertices and edges through their proof of cumulative products. Our disjoint statement is based on the NAND proof of the MoniPoly ABC [28, 29] which proves that a queried graph \(G'\) does not appear in the graph signature. Table 2 displays the supported statements in a predicate. The following sub-sections explain the corresponding constructions.

Table 2. Proof of knowledge predicate for graph signature.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>set((G'))</td>
<td>Representation of a set (G' \subseteq G), based on AND((G')).</td>
</tr>
<tr>
<td>disjoint((G'))</td>
<td>Graph disjointness ((G' \cap G) = \emptyset), based on NAND((G')).</td>
</tr>
<tr>
<td>cover((\ell, G'))</td>
<td>Graph coverage (</td>
</tr>
<tr>
<td>vertices</td>
<td>Proof of composition of graph vertices, based on set.</td>
</tr>
<tr>
<td>edges</td>
<td>Proof of composition of graph edges, based on set.</td>
</tr>
<tr>
<td>possession</td>
<td>Proof of possession of a graph signature (\sigma) on the graph (G = (V \cup E)).</td>
</tr>
<tr>
<td>edge((i,j))</td>
<td>Adjacency of ((i,j)) is a set statement on an edge identifier.</td>
</tr>
<tr>
<td>connected((i,j,\ell))</td>
<td>Existence of the (\ell)-path between vertices (i) and (j), based on edges and set statements.</td>
</tr>
<tr>
<td>isolated((i,j))</td>
<td>Isolation of vertices (i) and (j), based on set and disjoint.</td>
</tr>
</tbody>
</table>

*Note: AND, NAND, ANY are show proofs adapted from the MoniPoly ABC system [28, 29].*

7.2 Proof of Possession

The proof of possession is specified by the \(\phi_{\text{possession}}\) predicate. It is a compound proof of knowledge on the validity of prover’s blinded final graph signature \(\sigma'\).
and its encoded graph $G = (V \cup E)$. Prover can interact with the verifier to construct the proof of possession as follows:

$$PK \left\{ \forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}, \forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \zeta, \rho, \omega, \tau, \gamma \right\} :$$

$$\prod_{i \in V} e \left( W'_i, X_{0_i}^{\varepsilon_{i_0}}, X_{0_i}^{\varepsilon_{i_1}} \right) \prod_{(i,j) \in E} e \left( W'_{(i,j)}, X_{0_i}^{\varepsilon_{(i,j)_0}}, X_{0_j}^{\varepsilon_{(i,j)_1}} \right).$$

$$e \left( h^{\zeta} b^\rho c^\omega v'^{\tau}, X_{0_b} \right) = e \left( v'^{\gamma}, X \right)$$

where $v', \{W'_i, W'_{(i,j)}\}$ are the public inputs. Note that the prover can set $\{\varepsilon_j, \varepsilon_{(i,j)}\}$ as an attribute randomly chosen from the vertex and edge attributes, respectively.

**Remark 4.** Proving the knowledge of $\{\varepsilon_i, \varepsilon_{(i,j)}\}$ is a must in the possession statement. It should not be replaced by a witness $W$ such that the equation

$$e \left( W h^{\zeta} b^\rho c^\omega v'^{\tau}, X_{0_b} \right) = e \left( v'^{\gamma}, X \right)$$

holds. This is because an adversary knowing the quadruple $(v', W, h^{\tau_k} b^s c^t v'^r, v'^r)$ from the transcript can easily create another valid quadruple:

$$(v'' = v'^y, W' = \left( Wh^{\tau_k} b^s c^t v'^r \right)^z, h^{z\tau_k} b^sz^t c^z v'^ny, v'^ny)$$

such that:

$$e \left( W'^{h^{\tau_k} b^s c^t v'^r}, X_{0_b} \right) = e \left( v'^{ny}, X \right).$$

We highlight that this feature is useful for a group signature scheme. Concisely, viewing the graph signing protocol as the group joining protocol, $(t, s, v)$ is the group user secret key. Let $W$ combine all elements at the left hand side, the validity of the group user secret key can be proven by the equation $e (W, X_{0_b}) = e (v'^r, X)$. Subsequently, the signing and verification can be constructed by using the Fiat-Shamir paradigm.

### 7.3 set Proof

When the verifier requests a show proof for the predicate $\phi_{\text{set}(G')}$, i.e., showing a sub-graph $G' = (V' \cup E')$ is inside the graph $G$ such that $G' \subseteq G$, prover can
interact with the verifier to construct the set proof as follows:

\[
PK\left\{\left(\forall i \in V \setminus V' : \varepsilon_{i_0}, \varepsilon_{i_1}, (\forall (i, j) \in E \setminus E' : \varepsilon_{(i, j)_0}, \varepsilon_{(i, j)_1}), \zeta, \rho, \omega, \tau, \gamma \right) : \right.
\]
\[
\prod_{i \in V \setminus V'} e\left(W'_i, X'_{0_{i_0}} X'_{0_{i_1}}\right) \prod_{(i, j) \in E \setminus E'} e\left(W'_{(i, j)}, X'_{0_{i_0}} X'_{0_{i_1}}\right).
\]
\[
\prod_{i \in V'} e\left(W'_i, \prod_{k=0}^{E'_i} X'^{m_{i_k}}_{0_k}\right) \prod_{(i, j) \in E'} e\left(W'_{(i, j)}, \prod_{k=0}^{E'_{(i, j)}} X'^{m_{(i, j)_k}}_{0_k}\right)
\]
\[
e\left(h^s b^p c^\omega u'^{\tau}, X'_{0}\right) = e(u', X).
\]

The correctness for the witnesses \(\{W_i, W_{(i, j)}\}\) for the queried graph \(G'\) can be verified from the following:

\[
e\left(\prod_{i \in V'} C_i \prod_{(i, j) \in E'} C_{(i, j)} X_0\right)
\]
\[
= \prod_{i \in V'} e\left(a_{i_0}, x_{i_0}^{r_{0(i, j)}(x'+i)} \prod_{w \in f_{V'}(x'+i+w)} X_0\right) \prod_{(i, j) \in E'} e\left(a_{(i, j)_0}, x_{(i, j)_0}^{r_{0(i, j)}(x'+i)(x'+j)} \prod_{w \in f_{E'}(x'+i+w)} X_0\right)
\]
\[
= \prod_{i \in V'} e\left(W'_i, \prod_{k=0}^{E'_i} X'^{m_{i_k}}_{0_k}\right) \prod_{(i, j) \in E'} e\left(W'_{(i, j)}, \prod_{k=0}^{E'_{(i, j)}} X'^{m_{(i, j)_k}}_{0_k}\right)
\]

The set proof above gives rise to some diversification and we describe a few which we think are important for a graph signature scheme:

1. \(\phi_{set(V')}\): remove \(\prod_{(i, j) \in E'} e\left(W'_{(i, j)}, \prod_{k=0}^{E'_{(i, j)}} X'^{m_{(i, j)_k}}_{0_k}\right)^{\varepsilon_{(i, j)}}\).
2. \(\phi_{set(E')}\): remove \(\prod_{i \in V'} e\left(W'_i, \prod_{k=0}^{E'_i} X'^{m_{i_k}}_{0_k}\right)^{\varepsilon_i}\).
3. \(\phi_{set(A_i)}\): given labels \(A_i \subset V'_i\), replace \(\prod_{i \in V'} e\left(W'_i, \prod_{k=0}^{E'_i} X'^{m_{i_k}}_{0_k}\right)^{\varepsilon_i}\)
   \[e\left(W'_i, \prod_{k=0}^{A_{i_k}} X'^{m_{i_k}}_{0_k}\right)^{\varepsilon_i}\] with
   \[e\left(W'_i, \prod_{k=0}^{A_{i_k}} X'^{m_{i_k}}_{0_k}\right)^{\varepsilon_i}\].
4. \(\phi_{set(A_{(i, j)})}\): given labels \(A_{(i, j)} \subset E'_{(i, j)}\), replace \(\prod_{(i, j) \in E'} e\left(W'_{(i, j)}, \prod_{k=0}^{E'_{(i, j)}} X'^{m_{(i, j)_k}}_{0_k}\right)^{\varepsilon_{(i, j)}}\)
   \[e\left(W'_{(i, j)}, \prod_{k=0}^{A_{(i, j)_k}} X'^{m_{(i, j)_k}}_{0_k}\right)^{\varepsilon_{(i, j)}}\] with
   \[e\left(W'_{(i, j)}, \prod_{k=0}^{A_{(i, j)_k}} X'^{m_{(i, j)_k}}_{0_k}\right)^{\varepsilon_{(i, j)}}\].
5. Complex cases: given \(\phi_{set(i, \ldots, i_t)}\) and additionally asked to prove the connection among the vertex identifiers.

**Lemma 11.** The randomization of \(G'\) in the set predicate is perfectly hiding.
Proof. The blinding factors \(a_i, a_{(i,j)}\) act as the opening values and turns the \(G_2\) elements into MoniPoly commitments:

\[
\prod_{i \in V'} e\left( \prod_{k=0}^{\ell'} X_{0_k}^{m_{i,k}} \right) \prod_{(i,j) \in E'} e\left( \prod_{k=0}^{\ell'} X_{0_k}^{m_{(i,j),k}} \right)
\]

which are perfectly hiding. Moreover, the \(G_2\) elements also form a Pedersen set commitment:

\[
\prod_{i \in V'} e\left( \prod_{k=0}^{\ell'} X_{0_k}^{m_{i,k}} \right) \prod_{(i,j) \in E'} e\left( \prod_{k=0}^{\ell'} X_{0_k}^{m_{(i,j),k}} \right)
\]

that is perfectly hiding according to Lemma 14. \(\square\)

7.4 cover Proof

If a prover is requested to prove the knowledge of \(\ell\) out of all vertices and edges in a set proof, it becomes a coverage proof cover. To be precise, when the verifier requests a show proof for the predicate \(\phi_{\text{cover}(\ell, G')}\), the prover proves that part of \(G'\) is in \(G\) such that \(|G' \cap G| \geq \ell\), without verifier knowing what \((G' \cap G)\) is. The protocol can be executed as follows:

\[
PK\left\{ (\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}), (\forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), (\forall i \in V' - V : \varepsilon_{i_0}, \varepsilon_{i_1}),
(\forall (i,j) \in E' - E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), \zeta, \rho, \tau, \gamma) : \right.
\]

\[
\prod_{i \in (V' \cap V)} e\left( W_{i}, X_{0_i}^{\varepsilon_{i_0}} \right) \prod_{(i,j) \in (E' \cap E)} e\left( W_{(i,j)}, X_{0_i}^{\varepsilon_{(i,j)_0}} \right),
\]

\[
\prod_{i \in (V - V')} e\left( W_{i}, X_{0_i}^{\varepsilon_{i_0}} \right) \prod_{(i,j) \in (E - E')} e\left( W_{(i,j)}, X_{0_i}^{\varepsilon_{(i,j)_0}} \right),
\]

\[
e\left( h^{\ell'} t^{\rho' e^{\omega' - \tau}}, X_{0_0} \right) = e(u^{\gamma'}, X) \land
\]

\[
\prod_{i \in V'} e\left( W_{i}', X_{0_i}^{\varepsilon_{i_1}} \right) \prod_{(i,j) \in E'} e\left( W_{(i,j)'}, X_{0_i}^{\varepsilon_{(i,j)_0}} \right) =
\]

\[
\prod_{i \in V'} e\left( W_{i}', \prod_{k=0}^{\ell'} X_{0_k}^{m_{i,k}} \right) \prod_{(i,j) \in E'} e\left( W_{(i,j)'}, \prod_{k=0}^{\ell'} X_{0_k}^{m_{(i,j),k}} \right)
\]

where \(u', \{W_{i}', W_{(i,j)'}\}\) are the public inputs. The details for the exponents are \(\{m_{i}\} = \text{MP Encode}(V'_i), \{m_{(i,j)}\} = \text{MP Encode}(E'_i), \zeta = r \times s k_0, \rho = s', \omega = r, \varepsilon_{i_1} = r_i, \varepsilon_{(i,j)_1} = r_{(i,j)}, \tau = t', \gamma = y\) for the randomly chosen blinding factors \(r, y, r_i, r_{(i,j)} \in \mathbb{Z}_p^*\). The same diversification for set proof is applicable to the cover proof.
Lemma 12. The randomization of $G'$ in the cover predicate is perfectly hiding if $G' \in G_u$.

Proof. The proof is similar to that of Lemma 11.

\[\Box\]

7.5 disjoint Proof

When the verifier requests a show proof for the predicate $\phi_{\text{disjoint}}(G')$, i.e., showing a sub-graph $G' = (V' \cup E')$ is not inside the graph $G$, it is sufficient to show that the vertices in both graphs are disjoint such that $(V' \cap G) = \emptyset$. In order to achieve this, the prover needs to separate the vertex sets from the edges set. This can be done by making use of the vertices predicate as well as the set membership proof for edge identifiers from the edges predicate. The latter is because proving a commitment has two vertex identifiers implicitly proves that it is an edge. The disjoint predicate is as follows:

\[
PK\left\{\left(\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}\right),\left(\forall (i,j) \in E : \varepsilon_{(i,j)_1}, \varepsilon_{(i,j)_2}\right), \zeta, \rho, \omega, \tau, \gamma, \{\alpha_k\}_{k=0}^{\left|V'\right|-1}\right) : \\
\prod_{i \in V} e\left(W'_i, \prod_{k=0}^{1} X_{\varepsilon_{i_0}}^k_i\right) \prod_{(i,j) \in E} e\left(W'_{(i,j)}, \prod_{k=0}^{2} X_{\varepsilon_{(i,j)_0}}^k_j\right) \cdot \\
e^{h^{\zeta} b^\omega c^{\tau} d^{\rho} e^{\gamma}, X_0} = e^{v^{\gamma}, X}\right\}
\]

vertices$(V) \land \varepsilon_{(i,j)} [C_{(i,j)}]_e \in \Xi_V$ \land 
\[e\left(a_{0_0}, W_{\gamma_i}\right) = e\left(W_V, \prod_{k=0}^{\left|V'\right|} X_{m_0}^k\right) e\left(\prod_{k=0}^{\left|V'\right|-1} a_{0_0}^k, X_{0_0}\right) \land 1_G \neq \prod_{k=0}^{\left|V'\right|-1} a_{0_0}^k\]

where $\{m_k\} = \text{MP Encode}(\forall i \in V')$ and $W_{\gamma_i}$ is provided by vertices.

Lemma 13. The disjoint predicate is perfectly hiding.

Proof. The proof is similar to that of Lemma 1 and 11.

\[\Box\]

In Groß' graph signature scheme [18], the pool of bases are deliberately separated for vertices and edges. If we also apply the same setting, we get a more
efficient disjoint predicate as follows:

\[
P K \left\{ (\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}), (\forall (i, j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), \zeta, \rho, \omega, \tau, \gamma, \{ \alpha_k \}_{k=0}^{|V'|-1} : \right. \\
\prod_{i \in V} e \left( W'_{i}, \prod_{k=0}^{1} X_{i k}^{\varepsilon_{i k}} \right) \prod_{(i,j) \in E} e \left( W_{(i,j)}, \prod_{k=0}^{1} X_{(i,j) k}^{\varepsilon_{(i,j) k}} \right).
\]

\[
e \left( h^{\zeta} b^\rho e^\omega u'^{-\tau}, X_0 \right) = e(u'^\gamma, X \wedge \\
\left( \prod_{i \in V} W'_{i}, X_{0_0} \right) = \prod_{i \in V} e \left( W_{i}, \prod_{k=0}^{1} X_{i k}^{m_k} \right) e \left( \prod_{k=0}^{1} \alpha_k^0, X_{0_0} \right) \wedge 1_\mathbb{G} \neq \prod_{k=0}^{1} a_{0_0}^{\alpha_k} \right \}
\]

where \( \{ m_k \} = \text{MP Encode} (\bar{\gamma} \in V') \).

There is an alternate proof, namely, disjoint* whose complexity is independent of total system vertices \(|\Xi_V|\). However, disjoint* has to prove \(|V'| \times |V|\) secret exponents in \( G_1 \) instead of \(|V'|\) as in disjoint. In order to simplify the notation, we assume the vertex identifier \( \bar{i} \), instead of the vertex label, for every \( V'_i \in V' \) does not appear in \( G \). Prover can interact with the verifier to construct the disjoint* proof as follows:

\[
P K \left\{ (\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}), (\forall (i, j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), (\forall \bar{i} \in V' : \{ \alpha_i \}_{i=1}^{V|}, \right. \\
\zeta, \rho, \omega, \tau, \gamma : \left. \right. \\
\prod_{i \in V} e \left( W_{i}, \prod_{k=0}^{1} X_{i k}^{\varepsilon_{i k}} \right) \prod_{(i,j) \in E} e \left( W_{(i,j)}, \prod_{k=0}^{1} X_{(i,j) k}^{\varepsilon_{(i,j) k}} \right).
\]

\[
\forall \bar{i} \in V' : \\
\prod_{i \in V} e \left( W_{i}, \prod_{k=0}^{1} X_{i k}^{\varepsilon_{i k}} \right) = \\
e \left( \prod_{i \in V} W_{i}, X_{0_0} \bar{X}_{0_0} \right) e \left( \prod_{i \in V} a_{i_0}^{\alpha_i}, X_{0_0} \right) \wedge 1_\mathbb{G} \neq a_{i_0}^{\alpha_i} \right \}
\]

where \( u', \{ W_{i}, W_{(i,j)} \} \) are the public inputs and \( \zeta = r \times s k_0, \rho = s', \omega = r, \varepsilon_{i_1} = r_i, \varepsilon_{(i,j)_1} = r_{(i,j)_1}, \tau = t' \) and \( \gamma = y \) for the randomly chosen blinding factors.
Let $r, y, r_i, r_{(i,j)} \in \mathbb{Z}_p$. The correctness for the second statement is as follows:

\[
e \left( \prod_{i \in V} C_i, X_{00} \right) = e \left( \prod_{i \in V} a_{i0}^{\alpha_i} \prod_{k \in V_i} (x' + k), X_{00} \right)
\]

\[= e \left( \prod_{i \in V} a_{i0}^{q_i(x')(x' + i) + \bar{r}_i}, X_{00} \right) e \left( \prod_{i \in V} a_{i0}^{\bar{r}_i}, X_{00} \right)
\]

\[= e \left( \prod_{i \in V} a_{i0}^{\alpha_i(x')}, X_{00} \right) e \left( \prod_{i \in V} a_{i0}^{\alpha_i}, X_{00} \right)
\]

where $q_i(x')$ is the quotient polynomial and $\alpha_i = \bar{r}_i$ is the remainder. Note that the second last statement should not be stacked together or it conflicts with the last statement. When the verifier queries $\phi_{\text{disjoint}}(E')$ only, the prover can modify the last statement accordingly to prove $(E' \cap G) = \emptyset$. Note that $\prod_{i \in V} a_{i0}^{\alpha_i}$ in the second statement is a Pedersen set commitment and Lemma 13 still holds. For the completeness of this argument, we prove the perfectly hiding and binding properties as below.

**Lemma 14.** The Pedersen set commitment of the disjoint* statement is perfectly hiding.

**Proof.** Given a Pedersen set commitment $C = a_{i0}^{\alpha_1} a_{i0}^{\alpha_2} \cdots a_{i0}^{\alpha_L}$, there are $|\mathbb{Z}_p| - 1$ possible pairs of $(\alpha_1, (\alpha_2, \ldots, \alpha_L)) \neq (\alpha'_1, (\alpha'_2, \ldots, \alpha'_L))$ which can result in the same $C$. Furthermore, for each committed set $(\alpha_2, \ldots, \alpha_L)$, there is a unique $\alpha_1$ such that:

\[
d\log_{a_{i0}}(C) = \alpha_1 \prod_{j=2}^{L} d\log_{a_{i0}}(a_{j0}) \alpha_j \mod p
\]

\[
\alpha_1 = \frac{d\log_{a_{i0}}(C)}{\prod_{j=2}^{L} d\log_{a_{i0}}(a_{j0}) \alpha_j} \mod p
\]

Since $\alpha_1$ is calculated independently of the committed set $(\alpha_2, \ldots, \alpha_L)$, the latter is perfectly hidden. \hfill $\square$

**Lemma 15.** The Pedersen set commitment of the disjoint* statement is binding if the DLOG problem is hard.

**Proof.** Given a DLOG instance $(g, h = g^x)$, we construct a challenger $C$ that runs the adversary $A$ of a Pedersen set commitment scheme as the sub-routine to find
the solution \( x \). \( C \) sets \( a_{i0} = g \) and \( \{a_{i0} = h^{b_i}\}_{i=1}^L \) for randomly chosen \( b_i \in \mathbb{Z}_p^* \). \( C \) publishes \( \{a_{i0}\}_{i=0}^L \) as the public parameters for the Pedersen set commitment scheme.

If an adversary can output a Pedersen set commitment \( C \) for two different sets \( \{a_i\}_{i=0}^L, A^* = \{a_i^*\}_{i=0}^L \) such that \( |A \cap A^*| \geq 2 \) and:

\[
C = a_{i0}^{a_{i1}} \cdot a_{20}^{a_{i2}} \cdots a_{L0}^{a_{iL}} = a_{i0}^{a_{i1}^*} \cdot a_{20}^{a_{i2}^*} \cdots a_{L0}^{a_{iL}^*}
\]

\[
\Leftrightarrow g^{\sum b_i \cdot a_i} = g^{\sum b_i \cdot a_i^*}
\]

\( C \) can compute:

\[
x = \frac{\alpha_1 - \alpha_1^*}{\sum_{i=1}^L b_i (\alpha_i^* - \alpha_i)} \mod p
\]

to solve the DLOG problem.

From Table 3, we observe that the complexity of the disjoint* predicate is faster than that of the disjoint predicate only when \( n' \) is small. Let the complexity ratio \([29]\) at 128-bit security for a scalar multiplication in \( G_1 (M_1) \) to a scalar multiplication in \( G_2 \) and exponentiation in \( \mathbb{G}_T \) and pairing be 2, 6 and 9, respectively. Assuming \( l = 1 \), the disjoint predicate has a total of \((29m + n^2/2 + (127n)/2 + 2|\Xi| + 5n' + 29n + 27)M_1 \) while disjoint* has a total of \((30m + (5n + 32)n' + 29n + 27)M_1 \).

Setting \( n = 100, m = 1000 \) and \( |\Xi| = 1000 \), we get \((5n' + 42439)M_1 \) for disjoint and \((532n' + 8927)M_1 \) for disjoint*. As a result, as long as \( n' = |V'| \geq 18 \), disjoint* is slower than disjoint. On the other hand, at \( n = 1000, m = 100 \), we get \((5n' + 589389)M_1 \) for disjoint and \((5032n' + 35027)M_1 \) for disjoint* which pushes the threshold to \( n' = |V'| \geq 106 \). This shows that disjoint is suitable for graphs with little vertices but large edges while disjoint* is the other way round.

### 7.6 edge(i, j)

The edge(i, j) statement is a special case of set proof such that the query is a single edge identifier \( E' = (i^*, j^*) \) as below:

\[
PK \left\{ (\forall i \in V : \varepsilon_{i0}, \varepsilon_{i1}), (\forall (i, j) \in E \setminus (i^*, j^*) : \varepsilon_{(i, j)0}, \varepsilon_{(i, j)1}), \zeta, \rho, \omega, \tau, \gamma : \right. \\
\prod_{i \in V} e(W'_i, X_{0i}^{\varepsilon_{i0}}, X_{0i}^{\varepsilon_{i1}}) \prod_{(i, j) \in E \setminus E'} e(W'_{(i, j)}, X_{0i}^{\varepsilon_{(i, j)0}}, X_{0j}^{\varepsilon_{(i, j)1}}), \\
e(W'_{(i^*, j^*)}, X_{0i} \gamma, X_{0j} \rho) e(h^\zeta b^\rho \omega^\gamma \tau, X_{0b}) = e(v^\gamma, X) \left\}
\]

where \( \{W'_i, W'_{(i, j)}\} \) and \( W'_{(i^*, j^*)} = \prod_{k=0}^{[E'_{(i^*, j^*)}]} a_{(i, j)k}^r \prod_{w \in \mathbb{F}_{(i^*, j^*)}} (x' + w) \) are public inputs. If \( E'_{(i^*, j^*)} \in G_0 \), it is \( W'_{(i^*, j^*)} = \prod_{k=0}^{[E'_{(i^*, j^*)}]} a_{(i, j)k}^r \prod_{w \in \mathbb{F}_{(i^*, j^*)}} (x' + w) \) instead.
Lemma 16. The randomization of the isolated statement shows that two vertices $i$ and $j$ are connected by $\ell$ edges. This is an extension of edge($i, j$) statement, which is a simplified $\ell$ times edges within a set statement. The connected statement for $\ell$ edges $E'$ connecting $i$ and $j$ is as below:

$$PK\Big\{ (\forall i \in V : \varepsilon_{i0}, \varepsilon_{i1}),(\forall (i,j) \in E \setminus E') : \varepsilon_{(i,j)0}, \varepsilon_{(i,j)1} ,\{\varepsilon_{l,0}\}_{l=2}^{\ell-1}, \{\varepsilon_{l,1}\}_{l=2}^{\ell-1},\Big\}$$

$$\prod_{i \in V} e\left(W'_{i}, X_{01}^{\varepsilon_{i1}} X_{00}^{\varepsilon_{i0}}\right) \prod_{(i,j) \in E'} e\left(W'_{(i,j)}, X_{01}^{\varepsilon_{(i,j)1}} X_{00}^{\varepsilon_{(i,j)0}}\right) \cdot e\left(W'_{1}, X_{02}^{\varepsilon_{1,2}} (X_{00}^{\varepsilon_{1,1}} (X_{01}^{\varepsilon_{1,1}})^{2})\right) \cdot \cdots \cdot e\left(W'_{\ell-1}, X_{02}^{\varepsilon_{\ell-1,2}} (X_{01}^{\varepsilon_{\ell-1,1}})^{2}\right) \cdot e\left(W'_{\ell}, X_{02}^{\varepsilon_{\ell,2}} (X_{01}^{\varepsilon_{\ell,1}})^{2}\right) \cdot e\left(W'_{\ell}, X_{02}^{\varepsilon_{\ell,2}} (X_{01}^{\varepsilon_{\ell,1}})^{2}\right)$$

where $\{W'_{i}, W'_{(i,j)}\}$ and $W'_{1}, \ldots, W'_{\ell}$ are public inputs. The details for the secret exponents are listed below:

- $\varepsilon_{1,2} = \varepsilon_{2,2} = \cdots = \varepsilon_{\ell,2} = r(i^*, j^*)$,
- $\varepsilon_{1,1} = r(i^*, j^*) \cdot j_{1} = r(i^*, j^*) \cdot i_{2}$,
- $\varepsilon_{2,1} = r(i^*, j^*) \cdot j_{2}$,
- $\varepsilon_{2,0} = r(i^*, j^*) \cdot i_{2} j_{2}$,
- $\cdots$,
- $\varepsilon_{\ell-1,1} = r(i^*, j^*) \cdot j_{\ell-1} = r(i^*, j^*) \cdot i_{\ell}$

such that all $\ell$ edges share the same randomness $r(i^*, j^*) \in Z_{p}^*$.  

**Lemma 16.** The randomization of the $\ell$-path in the connected predicate is perfectly hiding.

**Proof.** The proof is similar to that of Lemma 1. $\square$

7.8 isolated($i, j$)

The isolated statement allows a prover to prove the possession of $i$ and $j$ as well as their disjointness. In brief, a prover shows that his graph consists of a vertex set and some sub-graphs that are represented by edge sets, with $i$ and $j$ fall in different sub-graphs. Specifically, a prover proves the vertex composition to separate the vertex set from the edge sets. Subsequently, a bi-partition variant of edges is executed to show that $i$ and $j$ are disjoint. Let $E_{i^*}$ and $E_{j^*} = E \setminus E_{i^*}$
represent the disjoint edge sets which consist of the queried $i^*$ and $j^*$ respectively. We describe the isolated statement as the following protocol:

\[
PK\left\{\left(\forall i \in V : \varepsilon_{i_{0}}, \varepsilon_{i_{1}}\right), \left(\forall (i, j) \in E : \varepsilon_{(i,j)_{0}}, \varepsilon_{(i,j)_{1}}, \varepsilon_{(i,j)_{2}}\right), \zeta, \rho, \omega, \tau, \gamma, \right\} \min\left(\left|\varepsilon_{Ei^*}\right|, \left|\varepsilon_{Ej^*}\right|\right) - 1:
\prod_{i \in V} e \left(W'_{i, 1} X_{\varepsilon_{i_{k}}}^{\varepsilon_{i_{0}}} X_{\varepsilon_{i_{1}}}^{\varepsilon_{i_{1}}}, \right) \prod_{(i,j) \in E} e \left(W'_{(i,j), 2} X_{\varepsilon_{(i,j)_{k}}}^{\varepsilon_{(i,j)_{0}}} X_{\varepsilon_{(i,j)_{1}}}^{\varepsilon_{(i,j)_{1}}}, \right)
\cdot e \left(h^{b_{0}} e^{\omega^{v'-\tau}}, X_{0}\right) = e \left(v'^{\gamma}, X\right) \wedge
\right\}
\]

8 Application Scenarios

While Groß proposed theoretical applications for graph signatures [18], e.g., in encoding statements from arbitrary NP languages, and cloud topology security applications [19] based on a graph grammar developed for virtualization security assurance [3], we briefly introduce an application scenario to illustrate how to employ the graph signature scheme and its proof system in practice.
Fig. 1. Example of a scale-free network graph with 200 vertices and 289 edges

First, we take for granted that there is infrastructure auditor engaging in topology certification [17], based on a faithful synched graph representation of a system of systems. In a virtualized infrastructure context, such graph representations can be obtained from host [27] and topology [5, 4] security assurance systems. How a specific assurance system obtains a faithful graph representation of the infrastructure is outside the scope of this paper.

Given such a setup, (1) vertices represent physical machines and devices, hypervisors, virtual machines, storage and network devices. (2) edges encode hierarchical, VLAN and inter-VM RPC connections, (3) labels can encode a wide range of properties incl. the types of vertices and edges, geo-location of physical devices, OS image versions, owner, role, etc. Figure 1 represents a typical structure of such a scale-free network. One could consider the leaves as virtual workloads, the adjacent vertices as the respective hypervisors.

Initially, the infrastructure auditor signs the input graph representation of the system, hence outputting a graph signature $\sigma$ to the infrastructure provider as signature recipient. Subsequently, the infrastructure provider acts as prover towards tenants in verifier roles. This enables the infrastructure provider to show, for example,

(1) that the provider is in possession of a certified graph signature without disclosing further information with the possession predicate,
(2) that all virtual machines of a tenant and co-located virtual machines of unknown other tenants are based on OS image versions deemed secure, by set membership proofs on labels, while keeping other tenants’.

(3) that the geo-location of different systems of a tenant is in different countries (encoded as labels) with the disjoint predicate (cf. [17]), thereby convincing the tenant of backup redundancy,

(4) that a tenant’s workloads are segregated from other tenant’s workloads or that a tenant’s mission-critical workloads are segregated from test systems by use of the isolated predicate.

Hence, the graph signature scheme and its proof system enables the infrastructure provider to convince different verifiers that security properties relevant for them are fulfilled, while keeping sensitive information confidential.

9 Performance

9.1 Computational Complexity

We list the show proofs complexity in Table 3 and that of the optimized version from Appendix A in Table 4. In order to ease the presentation, we assume every vertex \(V_i \in V\) and edge \(E_i \in E\) has the same label length \(l\).

<table>
<thead>
<tr>
<th>Statement</th>
<th>Point Add. ((G_1))</th>
<th>Point Mult. ((G_1))</th>
<th>Point Add. ((G_2))</th>
<th>Point Mult. ((G_2))</th>
<th>Mult. ((G_2))</th>
<th>Pairing</th>
</tr>
</thead>
<tbody>
<tr>
<td>vertices</td>
<td>((l + 2)m + 2(2</td>
<td>E_v</td>
<td>+</td>
<td>E_c</td>
<td>) + \sum_{k=1}^{n} (k + 1) + \sum_{i=1}^{2} (k + 1)</td>
<td>E_v</td>
</tr>
<tr>
<td>edges</td>
<td>((l + 3)m + 2</td>
<td>E_c</td>
<td>+ \sum_{k=1}^{n} (</td>
<td>E_v</td>
<td>+ 4n + ml + 2</td>
<td>E_c</td>
</tr>
<tr>
<td>possession</td>
<td>(I(n + m) + 9)</td>
<td>((l + 1)n + (l + 2)m + 15)</td>
<td>(3(n + m))</td>
<td>(6(n + m))</td>
<td>(n + m)</td>
<td>(n + m + 2)</td>
</tr>
<tr>
<td>set</td>
<td>((l - 1)n + (l + 1)m + (1 - l)n^* + 5(4l) + (l + 1) - l)n^* + 2(l + 2)m - 1m^* + 15)</td>
<td>(3n + m + (l - 1)n^* + 15)</td>
<td>(6n + m + (l - 5)n^* + (l - 4)m^*)</td>
<td>(n + m)</td>
<td>(n + m + 2)</td>
<td></td>
</tr>
<tr>
<td>disjoint</td>
<td>((2l + 2)m + 3n^* + 2</td>
<td>E_v</td>
<td>+ \sum_{k=1}^{n} 2</td>
<td>E_v</td>
<td>+ \sum_{k=1}^{n} (k + 1) + 16)</td>
<td>(4n + 3m + n^*)</td>
</tr>
<tr>
<td>disjoint*</td>
<td>(n^* + 2m + n + 2n^* (n + 3n - 3) + 9)</td>
<td>(n^* + 2m + n + 2n^* (n + 1) + n + m + n^*)</td>
<td>(6n + m + n^*)</td>
<td>(n + m + 2n^* + 1)</td>
<td>(n + m + 2n^* + 2)</td>
<td></td>
</tr>
<tr>
<td>cover</td>
<td>((l + 2)n + (l + 2)m + 15)</td>
<td>((l + 1)n + (l + 2)m + 14)</td>
<td>(3n + m - 1)</td>
<td>(6n + m - 4)</td>
<td>(n + m)</td>
<td>(n + m + 2)</td>
</tr>
<tr>
<td>edge</td>
<td>((l + 1)m + 8)</td>
<td>((l + 1)m + (l + 2)m - 15)</td>
<td>(3n + m + 6)</td>
<td>(6n + m + 3)</td>
<td>(n + m)</td>
<td>(n + m + 2)</td>
</tr>
<tr>
<td>connected</td>
<td>(n + m + m + \sum_{k=1}^{n} (2</td>
<td>E_v</td>
<td>) + \sum_{m=1}^{k} (</td>
<td>E_v</td>
<td>) + (l + 1))</td>
<td>(n^* + n + m + 2m + \sum_{t=1}^{n}</td>
</tr>
</tbody>
</table>

Note: \(l\): label length, \(n, |V|, m, |E|, 0 = |E_0|, m_1 = |E_1|, n_1 = |V_1|, m_1 = |E_1|, n = |V - V_1|, m = |E - E_1|, \(\epsilon\): threshold, * the slower alternative.
Table 4. Complexity for our optimized proofs of knowledge statements.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Point Add. ($G_1$)</th>
<th>Point Mult. ($G_1$)</th>
<th>Point Add. ($G_2$)</th>
<th>Point Mult. ($G_2$)</th>
<th>Mult ($G_T$)</th>
<th>Pairing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>vertices</strong></td>
<td>$(l + 13)n + 2</td>
<td>E</td>
<td>V</td>
<td>+ 2</td>
<td>E</td>
<td>E</td>
</tr>
<tr>
<td><strong>edges</strong></td>
<td>$</td>
<td>E</td>
<td>- 2</td>
<td>+ \sum_{i=1}^{l} (</td>
<td>E</td>
<td>- 1) - 7</td>
</tr>
<tr>
<td><strong>possion</strong></td>
<td>$(2l + 6)n + (2l + 8)m - 1</td>
<td>$</td>
<td>$(2l + 8)n + (2l + 10)m + 15</td>
<td>$</td>
<td>$0</td>
<td>6</td>
</tr>
<tr>
<td><strong>set</strong></td>
<td>$(2l + 4)m - (2l - 6)</td>
<td>+ (2l + 6)m - (2l - 5)</td>
<td></td>
<td>\sum_{i=1}^{l} (</td>
<td>E</td>
<td>- 1)</td>
</tr>
<tr>
<td><strong>set</strong></td>
<td>$(2l + 4)(n - n') + (2l + 6)(m - m')</td>
<td>+ n_{max}(2</td>
<td>n' + 2m' - 1</td>
<td>+ 5</td>
<td>3</td>
<td><em>{n</em>{max}}(</td>
</tr>
<tr>
<td><strong>disjoint</strong></td>
<td>$(2l + 4)(n + n' + 3</td>
<td>n' + 2</td>
<td>E</td>
<td></td>
<td>+ \sum_{i=1}^{l} (k + 1)</td>
<td>+ 9</td>
</tr>
<tr>
<td><strong>disjoint</strong></td>
<td>$(2l + 4)m + (2l + 8)(m + m') + (2l + 4)(n + n') + (2l + 6)(n + n')</td>
<td>+ (2l + 8)(m + m')</td>
<td>+ n_{max}(2</td>
<td>n' + 2m' - 1</td>
<td>+ 6</td>
<td>15</td>
</tr>
<tr>
<td><strong>cover</strong></td>
<td>$(2l + 4)m + (2l + 6)m - l</td>
<td>- 3</td>
<td>(2l + 8)n + (2l + 10)m - l + 6</td>
<td>2</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td><strong>isolated</strong></td>
<td>$(2l + 6)m + (2l + 8)m - (l + 2)</td>
<td>- 3</td>
<td>0</td>
<td>9</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td><strong>point</strong></td>
<td>$</td>
<td>E</td>
<td>- 2</td>
<td>+ \sum_{i=1}^{l} (</td>
<td>E</td>
<td>)</td>
</tr>
</tbody>
</table>

Note: $l$: label length, $n$: $|V|$, $m$: $|E|$, $n_0$: $|E|_0$, $m_0$: $|E|_0$, $n'$: $|V'|$, $m'$: $|E'|$, $n$: $|V'|$, $m$: $|E'|$, $|E| - k$, $\ell$: threshold, $*$: the slower alternative, $\#$: the $n_{max}$ alternative.

We compare the efficiency of show proofs of our proposed graph signature to that of Grob's graph signature [18, 19] in Table 5. We first recall the notations from Grob's graph signature scheme. Let $n$ be the total vertices and $m \leq n(n-1)$ be the total edges in a graph. Also, let $k$ be the number of sets considered in a statement where $k \leq l \leq n$ and $O(kl) = O(n)$. We follow the approach in Grob's work in computing the asymptotic complexity where we consider only the complexity of the core computations to realize a statement. For instance, the $O(1)$ for edge in Grob's signature does not include the complexity of possession, and neither does ours.

Our graph signature outperforms Grob's graph signature in some statements because the set membership proofs in vertices and edges implicitly prove pair-wise disjointness. This allows our graph signature to enjoy a more efficient graph well-founded proof as well as realizing the disjoint statement using the set statement instead of the more complicated partition statement [18, 19]. Our proofs can be further optimized to reduce its complexity as shown in Appendix A. The optimization transfers the computation cost from pairing to the point multiplication in $G_1$ for every statement and achieves better efficiency. For instance, the optimization on possession reduces $n + m + 2$ pairings to only 4 while added $n(|f_{V'}(i) + 7| + m(|f_{E'}(i) + 8|)$ point multiplications in $G_1$. Moreover, Table 5 shows that the optimization on set reduces the complexity from $O(|V'|)$ to $O(n_{max})$ where $n_{max} = max(|V'_1|, \ldots, |V'_r|)$, a number significantly smaller than $|V'|$.
Table 5. Efficiency comparison for proofs of knowledge statements.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Groß [18, 19]</th>
<th>Ours Non-optimized</th>
<th>Ours Optimized</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Basis</td>
<td>$O$</td>
<td>$O_{G_T}$</td>
</tr>
<tr>
<td>basis</td>
<td>-</td>
<td>$O(n + m)$</td>
<td>$O_{G_T}(n + m)$</td>
</tr>
<tr>
<td>vertices possession</td>
<td>$O(n)$</td>
<td>set $O_{G_T}(n)$</td>
<td>set $O_{G_1}(n)$</td>
</tr>
<tr>
<td>edges possession</td>
<td>$O(m)$</td>
<td>set $O_{G_T}(m)$</td>
<td>set $O_{G_1}(m)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$O(1)$</td>
<td>set $O_{G_T}(1)$</td>
</tr>
<tr>
<td>isolated edges</td>
<td>$O(m)$</td>
<td>edges $O_{G_T}(m)$</td>
<td>edges $O_{G_1}(m)$</td>
</tr>
</tbody>
</table>

Note: $n$: $|V|$, $m \leq \frac{n(n-1)}{2}$, $k \leq t \leq n$, $n'$: $|V'|$, $n'': |V' - V|$, $t$: threshold.

9.2 Implementation Benchmarks

We have implemented the MoniPoly 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 and uses a 128-bit security parameter. The implementation realizes the key generation, issuing and different predicate proof protocols in 16,767 lines of code.

For the presented performance experiments, we have generated scale-free Internet autonomous-system (AS) alike graphs with the algorithm developed by Elmokashfi et al. [15] with the Python library networkx, v. 2.5. Their graph size was parametrized by the number of vertices, the number of edges being a function thereof. These graphs capture characteristics of a range of natural graph configurations of Internet systems. The vertex-to-edge ratio was typically approximately 1:1.5. Hence, for instance, for the case of 400 vertices the graph contained 551 edges, whereby the MoniPoly graph signature scheme encoded 951 graph elements (vertices+edges). The number of labels was fixed to 5 per vertex and edge. We depicted such a graph of intermediate size in Figure 1 of Section 8. For each graph size specified for the experiment, we generated and stored as reference sample a set of 10 graphs. For the isolation proofs, we generated an additional random 3-vertex user graph as second input for the
proof system, constrained to be disjoined from the experiment graphs, that is, fulfilling isolation by design.

Subsequently, we computed 10 warm-up rounds and then 100 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.

Figure 2 summarizes the results for all performance experiments, where each datapoint constitutes the mean of the evaluated execution times. We observe that the performance of all issuing protocols and the proof-system protocols apart from isolation is a flat linear function of the graph size. Let us consider issuing, non-isolation proofs and isolation proofs in turn.

**Issuing.** For a graph size of 10 vertices, the issuing took approximately 1.8 seconds. For a graph size of 400 vertices, the initial issuing took 15 seconds, further issuing protocols were completed in less than 60 seconds.

**Non-Isolation Proofs.** For graphs with 10 vertices, all non-isolation proof protocols were executed less than one second. For 400-vertex graphs, the non-
isolation proof protocols all ran for approximately 20 seconds. For computing zero-knowledge proofs of knowledge on a graph signature with nearly 1,000 elements with a realistic key size these performance results are encouraging.

Isolation Proofs. Considering the performance of the isolation proof protocol, we observe that the execution time grows super-linearly. For a graph with 10 vertices, this proof executed in approximately 30 seconds. For a graph with 400 vertices, the isolation proof typically took less than 750 seconds, that is, less than 12.5 minutes. While the execution time is impacted by a number of pairings as well, the true culprit for this is generation of and proof of representation of cumulative products of edge sets of graph partitions to be isolated. Creating those has a complexity of $O(n^2)$, where $n$ is the number of edges in the corresponding partition. Notably, use cases in which an infrastructure provider proves isolation of the resources of a tenant from other tenants, such as proposed by Groß [19], allow the provider to compute the signature proof of knowledge of the isolation statement offline, whereby the execution time of the isolation proof is less of an issue.

10 Conclusion

In this paper, we proposed a SDH-based graph signature scheme based on MoniPoly encoding. We rigorously proved the security of our graph signature scheme in the standard model with tight reduction. Our graph signature scheme provides more efficient show proofs compared to that of the RSA-based graph signature scheme.

Acknowledgment

This work was supported by the ERC Starting Grant CASCAde (GA n°716980).
Bibliography


A Optimized Proofs

We present the optimized proofs for the proposed graph signature scheme. The exponents which represent the encoded vertices and edges at $G_2$ bases ($X_i^{k}, X_{(i,j)}^{k}$) in a pairing are moved to the $G_1$ bases ($W_i, W_{(i,j)}$). Therefore, every pairing can now be compressed together by sharing the same $X_i^{k}$, i.e., reduces the number of pairings from linear to $W_i$, to linear to $X_i^{k}$. For instance, the vertex decomposition in the graph statement can be rewritten as:

$$
\prod_{i \in V} e\left(\prod_{w \in f_{V(V)}(x'+w)}^{a_{0}} X_{0}^{(x'+i)}\right) = \prod_{i \in V} e\left(W_{i}^{r_{i}}, X_{0}^{r_{i}},X_{0}^{r_{i}}\right)
$$

where $|V|$ pairings are compressed to only two pairings.

**Lemma 17.** The randomization in the optimization gives perfectly hiding property.

**Proof.** The random exponents $u_0, u_1$ turns the $G_T$ elements

$$
e\left(\prod_{i \in V} W_{i}^{r_{i}r_{i}^{-1}}, X_{0}^{u_{1}}\right)^{u_{1}} e\left(\prod_{i \in V} W_{i}^{r_{i}r_{i}^{-1}}, X_{0}^{u_{0}}\right)^{u_{0}}$$

into a Pedersen set commitment (Lemma 14) which is perfectly hiding. \hfill \Box

---


A.1 bootstrap Proof

The bootstrap statement can be optimized as follows:

\[ PK \left\{ \left\{ \bar{v}_i : \{ \varepsilon_{i,1} \}_{i=1}^{\ell}, \bar{v}(i,j) : \{ \varepsilon_{(i,j),1} \}_{i,j}^{2} \right\}, \mu_0, \mu_1, \mu_2 \right\} : \]

\[ e \left( \prod_{i \in V} C_i \prod_{(i,j) \in E} C^{(i,j)}, X_0 \right) = \prod_{k=0}^{\ell} e \left( \prod_{i \in V} \left( W_i^l W_i \right)^{\varepsilon_{i,k}}, X_0 \right) \land \]

\[ e \left( \prod_{(i,j) \in E} W_{(i,j)}^l W_{(i,j)} \right) e \left( W_{(i,j)}^l, X_{(i,j)} \right) \}

where the number of pairings are reduced from \( 2 + 2(|V| + |E|) \) to \( 7 + |V| + |E| \).

We can further compress the proof to save another 3 pairings but we present it in the current form for clarity purposes.

A.2 vertices Proof

Let \( V = \bigcup \{ V_1, \ldots, V_\ell \} \) be the vertex set in a graph \( G \), the prover runs the following protocol:

\[ PK \left\{ \left\{ \varepsilon_{i,1} \right\}_{i=1}^{\ell}, \mu_0, \mu_1 \right\} : e \left( \prod_{i \in V} C_i, X_0 \right) = e \left( \prod_{i=1}^{\ell} W_i^{\varepsilon_{i,1}}, X_0 \right) \land \]

\[ e \left( W_{1,1} X_0 \right) = e \left( a_{0,0}^{\varepsilon_{1,1}}, X_0 \right) \land e \left( a_{0,0}^{\varepsilon_{1,0}}, X_0 \right) \land e \left( a_{0,0}, W_{1,1} \right) = e \left( W_{1,1}, X_0 \right) \land \]

\[ e \left( W_{2,1} X_0 \right) = e \left( W_{2,1}, X_0 \right) \land \]

\[ e \left( W_{3,1} X_0 \right) = e \left( W_{3,1}, X_0 \right) \land \cdots \land e \left( a_{0,0}, W_{1,1} \right) = e \left( W_{1,1}, X_0 \right) \land \]

\[ e \left( \prod_{k=0}^{l} a_{0,k}^{m_{k}}, X_0 \right) = e \left( W_{2,1}, X_0 \right) \land \]

\[ e \left( a_{0,0}, W_{1,2} \right) = e \left( W_{2,1}, X_0 \right) \land e \left( a_{0,0}, W_{1,3} \right) = e \left( W_{3,1}, X_0 \right) \land \]

\[ \cdots \land e \left( a_{0,0}, W_{1,\ell} \right) = e \left( W_{\ell,1}, X_0 \right) \land e \left( \prod_{k=0}^{l} a_{0,k}^{m_{k}}, X_0 \right) = e \left( W_{\ell,1}, X_0 \right) \right\} \]
and the correctness can be observed from the following:

\[ e(C_{i_1}, X_{0_0}) = e\left( a_{0_0}^{\alpha_1(x'+i_1)} \prod_{w \in f\nu_1} (x'+w), X_{0_0} \right) \]
\[ = e\left( a_{0_0}^{\alpha_1 \cdot r_i^{-1} \prod_{w \in f\nu_1} (x'+w)} X_{0_0}^{r_i (x'+i_1)} \right) \]
\[ = e\left( W_{i_1}, X_{0_1}^{r_i} X_{0_0}^{r_i} \right) \]
\[ = e\left( W_{i_1}^{r_i u_1^{-1}}, X_{0_1}^{u_1} \right) e\left( W_{i_1}^{r_i u_1 u_0^{-1}}, X_{0_0}^{u_0} \right) \]
\[ = e\left( W_{i_1}^{r_i u_1}, X_{0_1}^{u_1} \right) e\left( W_{i_1}^{r_i}, X_{0_0}^{r_i} \right) \]

and,

\[ e\left( W_{\nu_1}, X_{0_0} \right) = e\left( a_{0_0}^{r_i (x'+i_1)} \prod_{k=1}^{r_i-1} r_k (x'+i_k), X_{0_0} \right) \]
\[ = e\left( a_{0_0}^{\prod_{k=1}^{r_i-1} r_k (x'+i_k)}, X_{0_0}^{r_i (x'+i_1)} \right) \]
\[ = e\left( W_{\nu_1-1}, X_{0_1}^{r_i} X_{0_0}^{r_i} \right) \]
\[ = e\left( W_{\nu_1-1}^{r_i u_1^{-1}}, X_{0_1}^{u_1} \right) e\left( W_{\nu_1-1}^{r_i u_1 u_0^{-1}}, X_{0_0}^{u_0} \right) \]
\[ = e\left( W_{\nu_1-1}^{r_i u_1}, X_{0_1}^{u_1} \right) e\left( W_{\nu_1-1}^{r_i}, X_{0_0}^{r_i} \right) \]

and,

\[ e\left( a_{0_0}, W_{\nu_{i_1}} \right) = e\left( a_{0_0} X_{0_0}^{\alpha_1 \cdot r_i^{-1} \prod_{w \in f\nu_1} (x'+w)} \prod_{k=1}^{r_i-1} a_{0_k}^{r_k^{-1} \prod_{w \in f\nu_1} (x'+w)} \right) \]
\[ = e\left( a_{0_0}^{\alpha_1 \cdot r_i^{-1} \prod_{w \in f\nu_1} (x'+w)}, X_{0_0}^{\prod_{k=1}^{r_i-1} a_{0_k}^{r_k^{-1} \prod_{w \in f\nu_1} (x'+w)}} \right) \]
\[ = e\left( W_{i_1}, W_{\nu_{i_1-1}} \right) \]

where \( \{ m_{\nu_k} \} = \text{MP Encode}(\Xi_{\nu}), \{ m_{\nu_{i_1}} \} = \text{MP Encode}(\Xi_{\nu_{i_{1}}}), W_{i_1} = \prod_{k=0}^{f\nu_1} a_{0_k}^{m_{i_k}}, \)
\( \varepsilon_{i_0} = r_i u_0^{-1}, \varepsilon_{i_1} = r_i u_1^{-1} \) and \( \{ \varepsilon_{i_k} \} = \alpha_1 \cdot r_i^{-1} \times \text{MP Encode}(f\nu_1) \) for randomly selected \( \alpha_1, r_i, u_1 \in \mathbb{Z}_p \). The public inputs \( (W_{\nu_1}, \ldots, W_{\nu_{i_1}}, W_{\nu_{i_{1}}}, W_{\nu_{i_{1}}}) \) are witnesses for the cumulative product of vertex identifiers while \( (W_{\nu_{i_1}}, \ldots, W_{\nu_{i_{1}}}, W_{\nu_{i_{1}}}) \) are witnesses for the cumulative product of vertex labels. Subsequently, simplifying the pairing notations
in the proof above gives:

\[
PK\left\{\{\varepsilon_{00}, \varepsilon_{11}\}_{i=1}^\ell, \mu_0, \mu_1\} : e\left(\prod_{i \in V} C_i, X_{00}\right) = \prod_{k=0}^1 e\left(\prod_{l=1}^\ell W_{v_{l+1}}, X_{0_k}\right) \land \right.

\[
e\left(\prod_{k=0}^{\varepsilon_k} a_{0_k}^{m_{\varepsilon_k}}, X_{00}\right) = \prod_{k=0}^1 e\left(\prod_{l=1}^\ell W_{v_{l+1}}, X_{0_k}\right) e\left(a_{00}^{-1} W_{\varepsilon\setminus v}, W_{v_2}\right) \land \]

\[
e\left(\prod_{k=0}^{\varepsilon_k} a_{0_k}^{m_{\varepsilon_k}}, X_{00}\right) e\left(a_{00}, \prod_{l=1}^\ell W_{\varepsilon_k}\right) = \prod_{l=1}^\ell e\left(W_l, W_{\varepsilon_{l-1}}\right) e\left(W_{\varepsilon\setminus f(V)}, W_{\varepsilon_l}\right) \right\}
\]

as the vertices\( (V) \) statement where \( W_{v_0} = a_{00}. \) Notice that this optimization decreases the number of pairing in vertices\( (V) \) from \( 5 + 3n \) into \( 10 + n \) only.

### A.3 edges Proof

Let \( E = \bigcup\{E_1, \ldots, E_\ell\} \) be the edge set in a graph \( G, \) the protocol below establishes the optimized statement edges\( (E): \)

\[
PK\left\{\forall (i, j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \varepsilon_{(i,j)_2}, \mu_0, \mu_1, \mu_2\} : \\
e\left(\prod_{(i,j) \in E} C_{(i,j)}, X_{00}\right) = \prod_{k=0}^2 e\left(\prod_{(i,j) \in E} W_{\varepsilon_{(i,j)_k}}, X_{0_k}\right) \land \\
e\left(\prod_{l=1}^{E_{|v|}} a_{0_k}^{m_{\varepsilon_l}}, \prod_{w_l=1}^{E_{|v|}-1} W_{\varepsilon_{w_l}}, X_{0_0}\right) = \prod_{l=1}^2 \prod_{k=0} e\left(\prod_{w_l=1}^{E_{|v|}} W_{\varepsilon_{w_l-1}}, X_{0_k}\right) e\left(a_{00}^{-1} W_{\varepsilon\setminus E_{|v|}}, W_{E_{|v|}}\right) \land \]

\[
e\left(\prod_{k=0}^{\varepsilon_k} a_{0_k}^{m_{\varepsilon_k}}, X_{00}\right) e\left(a_{00}, \prod_{l=2,3} W_{\varepsilon_l}\right) = \prod_{l=4,5}^\ell e\left(W_l, W_{\varepsilon_{l-1}}\right) e\left(W_{\varepsilon\setminus f(\ell(E_{|v|})}, W_{\varepsilon_{l-2,\ell-1}}\right) \right\}
\]

and can be rewritten as:
for undirected acyclic graph. On the other hand, the proof for a directed acyclic
graph is as follows:

\[
PK\left\{ \left( \forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \varepsilon_{(i,j)_2}, \mu_0, \mu_1, \mu_2 \right) : \\
\right.
\]

\[
e \left( \prod_{(i,j) \in E} C_{(i,j)}, X_{0_k} \right) = \prod_{k=0}^{2} e \left( \prod_{(i,j) \in E} W^{\varepsilon_{(i,j)_k}}_{(i,j)}, X_{0_k} \right) \land \\
\varepsilon_{(i,j)}[C_{(i,j)}] \in \Xi_V \land \varepsilon_{(i,j)}[C_{(i,j)}] \in \Xi_V \land \\
W_{(i,j)}[C_{(i,j)}] \in \Xi_L
\}
\]

where we also prove the extra vertex identifiers in every edge identifier of a
directed acyclic graph. Similar to the original version, the optimized proof above
is applicable to the directed cyclic graph.

A.4 Intermediary Signing

The optimized intermediary signing protocol uses the optimized proof of representation during the first step:

1. User randomly selects \( r_1, y, o_{i,j}, o_{(i,j)_1}, r_{(i,j)_2}, \mu_0, \mu_1 \in \mathbb{Z}_p^* \) and interacts with
   the signer to prove the possession of \( \sigma_{\text{init}} \) and the representation of his hidden
The same changes also appear in the final signing protocol.

A.5 Proof of Possession

The optimized proof for the possession statement is as follows:

\[
PK \left\{ \left( \forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}, \varepsilon_i \right), \left( \forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \varepsilon_{(i,j)} \right), \right.
\]

\[
\left. \zeta, \rho, \omega, \tau, \gamma, \mu_0, \mu_1 \right) : \\
\left( e \left( C_1 C_2 e_0^{\omega^t - \tau}, X_{0_0} \right) = e(\nu^\gamma, X) \right) \land \\
C_1 C_2 = h^\zeta b^\rho \prod_{i \in V} C_{i}^\varepsilon \prod_{(i,j) \in E} C_{(i,j)}^{\varepsilon_{(i,j)}} \land \\
e \left( \prod_{i \in V} C_i \prod_{(i,j) \in E} C_{(i,j)}, X_{0_0} \right) = \\
\prod_{k=0}^{1} \prod_{i \in V} \prod_{j \in E} W_i^{\varepsilon_{(i,j)_k}} W_{(i,j)}^{\varepsilon_{(i,j)_k}} X_{0_k}^{\mu_k}
\]

where \( \varepsilon_{i_1} = r_i \mu_1^{-1}, \varepsilon_{i_0} = r_i \mu_0^{-1}, \varepsilon_{(i,j)_1} = r_{(i,j)} \mu_1^{-1}, \varepsilon_{(i,j)_0} = r_{(i,j)} \mu_0^{-1} \) while the witnesses are

\[
W_i' = a_{i_0}^{\alpha_{i_0} r_i^{-1}} \prod_{w \in f_{V(i)}(v')} \omega_{i_0}, \xi_{i_0}
\]

and

\[
W_{(i,j)}' = a_{(i,j)_0}^{\alpha_{(i,j)_0} r_{(i,j)}^{-1}} \prod_{w \in f_{E(i,j)}(v')} W_{(i,j)_0}^{\mu_{(i,j)_0}}
\]

The same changes also appear in the final signing protocol.
A.6 set Proof

The optimized set proof is as follows:

\[
PK\left\{((\forall i \in V \setminus V' : \varepsilon_i, \varepsilon_{i_0}), (\forall (i,j) \in E \setminus E' : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), \zeta, \rho, \omega, \tau, \gamma, \mu_0, \mu_1) : \right.
\]

\[
\prod_{k=0}^{1} e\left(\prod_{i \in V \setminus V'} W_i^{\varepsilon_i_{i_0}} \prod_{(i,j) \in E \setminus E'} W_{(i,j)}^{\varepsilon_{(i,j)_0}}, X_{0_k}\right) \cdot \\
\prod_{i \in V'} e\left(W_i^{\varepsilon_{i_0}} \prod_{k=0}^{n_{\text{max}}} X_{0_k}^{m_i} \prod_{(i,j) \in E'} W_{(i,j)}^{\varepsilon_{(i,j)_0}}, X_{0_k}\right) \cdot \\
\prod_{k=0}^{n_{\text{max}}} e \left(h^r \rho^s \tau^{\gamma}, X_0\right) = e(v^{\gamma}, X) \}
\]

where \(\{m_i\} = \text{MPEncode}(V'_1), \{m_{(i,j)}\} = \text{MPEncode}(E'_1), \zeta = r \times sk_U, \rho = s', \omega = r, \varepsilon_{i_0} = r_k, \varepsilon_{(i,j)_1} = r_{(i,j)}, \tau = t', \gamma = y\) for the randomly chosen blinding factors \(r, y, r_k, r_{(i,j)} \in \mathbb{Z}_p^*\).

Noticed that the second line in the set proof above is the same as that in the original proof. If we apply the optimization mechanism on it, we can compress the \(|V'| + |E'|\) pairings into a possibly smaller number of pairings which is linear to the size of the largest encoded vertex or edge in \(G'\). Let \(n_{\text{max}} = \max(\max(|V'_1|, \ldots, |V'_{\text{max}}|), \max(|E'_1|, \ldots, |E'_{\text{max}}|))\) if \(n_{\text{max}} < |G'|\), the verifier can choose to verify the set proof as below:

\[
PK\left\{((\forall i \in V \setminus V' : \varepsilon_{i_0}, \varepsilon_i), (\forall (i,j) \in E \setminus E' : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), \zeta, \rho, \omega, \tau, \gamma, \mu_0, \mu_1) : \right.
\]

\[
\prod_{k=0}^{1} e\left(\prod_{i \in V \setminus V'} W_i^{\varepsilon_i_{i_0}} \prod_{(i,j) \in E \setminus E'} W_{(i,j)}^{\varepsilon_{(i,j)_0}}, X_{0_k}\right) \cdot \\
\prod_{i \in V'} e\left(W_i^{\varepsilon_{i_0}} \prod_{k=0}^{n_{\text{max}}} X_{0_k}^{m_i} \prod_{(i,j) \in E'} W_{(i,j)}^{\varepsilon_{(i,j)_0}}, X_{0_k}\right) \cdot \\
\prod_{k=0}^{n_{\text{max}}} e \left(h^r \rho^s \tau^{\gamma}, X_0\right) = e(v^{\gamma}, X) \}
\]
where \(m_{ik} = 0, m_{(i,j)k} = 0\) whenever \(|V'_i| < k\) and \(|E'_{(i,j)}| < k\), respectively. The correctness for \(\mathcal{G}'\) can be verified from the following:

\[
e \left( \prod_{i \in V'} C_i \prod_{(i,j) \in E'} C_{(i,j)} \cdot X_{0_0} \right) = e \left( \prod_{i \in V'} a_{i_0}^{r(x'_i+1)} \prod_{w \in V'(i)(x'_i+w)} a_{(i,j)_0}^{r(x'_i+1)(x'_j+w)}, X_{0_0} \right) = \prod_{i \in V'} e \left( W'_{i}^{x'_i}, X_{0_0} \right) = \prod_{i \in V'} \prod_{k=0}^{N_{max}} e \left( W_{i}^{x'_i,}, X_{0_k} \right) = \prod_{i \in V'} \prod_{(i,j) \in E'} \prod_{k=0}^{N_{max}} e \left( W_{(i,j)}^{x'_{i,j}}, X_{0_k} \right),
\]

and we have \(\{W'_i, E'_{(i,j)}\} \in \mathcal{G}_0\).

**A.7 cover Proof**

The optimized protocol can be executed as follows:

\[
PK \left\{ \langle \forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}, \rangle, \langle \forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \rangle, \langle \forall i \in V' - V : \varepsilon_{i_0}, \varepsilon_{i_1}, \rangle, \langle \forall (i,j) \in E' - E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \rangle, \langle \forall i \in V' : \varepsilon_{i_0}, \varepsilon_{i_1}, \rangle, \langle \forall (i,j) \in E' : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}, \rangle, \right\}:
\]

\[
\prod_{k=0}^{1} e \left( \prod_{i \in V} W_{i}^{x'_{i},} \prod_{(i,j) \in E} W_{(i,j)}^{x'_{i,j}}, X_{0_k} \right) e \left( h^\gamma b^\rho e^\omega \gamma - \tau, X_{0_0} \right) = e(\gamma, X) \wedge
\]

\[
\prod_{k=0}^{1} e \left( \prod_{i \in V'} W_{i}^{x'_{i},} \prod_{(i,j) \in E'} W_{(i,j)}^{x'_{i,j}}, X_{0_k} \right) = \prod_{i \in V'} \prod_{k=0}^{1} \prod_{(i,j) \in E'} e \left( W_{i}^{x'_{i},} \prod_{k=0}^{1} X_{0_k}^{x'_{i,j}}, X_{0_k} \right) \prod_{k=0}^{1} e \left( W_{(i,j)}^{x'_{i,j}}, X_{0_k} \right) \}
\]
where \( \{ m_{ik} \} = \text{MPEncode}(V_i'), \{ m_{(i,j)k} \} = \text{MPEncode}(E_{(i,j)}'), \zeta = r \times s_k U, \rho = s', \omega = r, \varepsilon_{i_1} = r_i u_1^{-1}, \varepsilon_{(i,j)_1} = r_{(i,j)} u_1^{-1}, \tau = t', \gamma = y, \mu_0 = u_0, \mu_1 = u_1 \) for the randomly chosen blinding factors \( r, y, r_i, r_{(i,j)}, u_0, u_1 \in \mathbb{Z}_p^* \). If \( G' \in G_U \) and \( n_{\text{max}} < |G'| \) where \( n_{\text{max}} = \max(\max(|V'_1|, \ldots, |V'_{|V'|-1}|)) \), the verifier can replace the right hand side of the last statement with

\[
\prod_{k=0}^{n_{\text{max}}} e \left( \prod_{i \in V'} W_{i,k}^{\alpha m_{ik}} \prod_{(i,j) \in E'} W_{(i,j)k}^{m_{(i,j)k}}, X_{0k} \right)
\]

during verification.

### A.8 disjoint Proof

Prover can interact with the verifier to construct the optimized disjoint proof as follows:

\[
PK \left\{ (\forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}), (\forall (i,j) \in E : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), \zeta, \rho, \omega, \tau, \gamma, \mu_0, \mu_1, \{ \alpha_k \}_{k=0}^{|V'|-1} : \right. 
\]

\[
\prod_{k=0}^{1} e \left( \prod_{i \in V} W_{i,k}^{\varepsilon_{i,k}}, X_{0k} \right) \prod_{(i,j) \in E} e \left( W_{(i,j)k}, \prod_{k=0}^{1} X_{(i,j)k}^{\varepsilon_{(i,j)_k}} \right),
\]

\[
e (h^s b^\rho c^\omega e^{r-t}, X_0) = e(v^\gamma, X) \wedge
\]

\[
e \left( \prod_{i \in V} W_{i,k}, X_{00} \right) = \prod_{i \in V} e \left( W_{i,k}, X_{0i} \right) \wedge \varepsilon_i [C_i] \in \Xi \wedge
\]

\[
e (a_{00}, W_{V_0}) = e \left( W_{V_0}, \prod_{k=0}^{[V'|-1]} X_{00}^{m_{0k}} \right) e \left( \prod_{k=0}^{[V'|-1]} a_{00}^{\alpha_k}, X_{00} \right) \wedge
\]

\[
1_G \neq W_V \wedge 1_G \neq \prod_{k=0}^{[V'|-1]} a_{00}^{\alpha_k},
\]

71
where \( \{m_k\} = \text{MP Encode}(\forall \bar{i} \in V') \). Next, the optimized disjoint* proof is as follows:

\[
\begin{align*}
PK \left \{ \forall i \in V : \varepsilon_{i_0^i}, \varepsilon_{i_1^i}, \forall (i, j) \in E : \varepsilon_{(i,j)_0^i}, \varepsilon_{(i,j)_1^i}, \forall \bar{i} \in V' : \{\alpha_i\}_{i=1}^{V'} \right \}, \\
\zeta, \rho, \omega, \tau, \gamma : \\
\prod_{i \in V} e \left( W_i, \prod_{k=0}^{1} X_{\varepsilon_{i_k^i}} \right) \prod_{(i,j) \in E} e \left( W_{(i,j)_0^i}, \prod_{k=0}^{1} X_{\varepsilon_{(i,j)_k^i}} \right) \\
\forall \bar{i} \in V' : \\
\prod_{i \in V} e \left( W_i, \prod_{k=0}^{1} X_{\varepsilon_{i_k^i}} \right) = \\
e \left( \prod_{i \in V} W_i, X_{0_i} \right) e \left( \prod_{i \in V} \alpha_{i_0^i}^0, X_{0_0} \right) \land 1_G \ne a_{0^i}^\alpha 
\end{align*}
\]

where \( v', \{W_i, W_{(i,j)}\} \) are the public inputs and \( \zeta = r \times s k_U, \rho = s', \omega = r, \varepsilon_{i_1} = r, \varepsilon_{(i,j)_1} = r, \tau = t' \) and \( \gamma = y \) for the randomly chosen blinding factors \( r, y, r, r, r_{i, j} \in \mathbb{Z}_p^* \).

The right hand side of second statement in the original proof requires \( 2|V'| \) pairings yet this optimized version requires only \( 1 + |V'| \) pairings. This is because \( e \left( \prod_{i \in V} W_i, X_{0_i} \right) \) can be accumulated for every \( i \) while the \( |V'| \) elements of \( e \left( \prod_{i \in V} W_i \right)^t a_{i_0^i}^\alpha, X_{0_0} \) are sufficient to avoid the conflicts with the last statements.

Table 4 shows a similar complexity relationship for disjoint* and disjoint predicates. Using the same setting from Section 7.5, the disjoint* has a total of \((30n + (5n + 23)n' + 29n + 18)M_1 \) while disjoint has a total of \((27m + 1/2n(n + 71) + |\Xi| + 5n' + 192)M_1 \). At \( n = 100, m = 1000, \) we get \((5n' + 36742)M_1 \) for disjoint and \((516n' + 32942)M_1 \) for disjoint*. The disjoint predicate is faster than disjoint* when \( n' \geq 7 \). On the other hand, at \( n = 1000, m = 100, \) we get \((5n' + 539392)M_1 \) for disjoint and \((5016n' + 32042)M_1 \) for disjoint* which then produces a threshold \( n' \geq 101 \).
A.9  edge(i, j)

Let $E' = (i^*, j^*)$, the edge(i, j) statement can be proved as below:

$$\text{PK}\left\{ (\forall i \in V : \varepsilon_{i0}, \varepsilon_{i1}), (\forall (i, j) \in E \setminus (i^*, j^*) : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), \zeta, \rho, \omega, \tau, \gamma, \mu_0, \mu_1 : \right.$$ 

$$\prod_{k=0}^{1} e \left( \prod_{i \in V} W'_i \cdot \prod_{(i,j) \in E \setminus E'} W'_{(i,j)} \cdot X_{0k} \right) e(W'_{(i^*, j^*)}, X_2 X'_{1}^{i^*} X_0) = e(v', X) \left\} \right.$$

where $\{W'_i, W'_{(i,j)}\}$ and $W'_{(i^*, j^*)}$ are public inputs.

A.10  connected(i, j, ℓ)

The optimized connected statement for ℓ edges $E'$ connecting i and j is as below:

$$\text{PK}\left\{ (\forall i \in V : \varepsilon_{i0}, \varepsilon_{i1}), (\forall (i, j) \in E \setminus E' : \varepsilon_{(i,j)_0}, \varepsilon_{(i,j)_1}), \{ \varepsilon_{i0}, \varepsilon_{i1} \}_{i=1}^{\ell}, \{ \varepsilon_{j0}, \varepsilon_{j1} \}_{j=1}^{\ell}, \zeta, \rho, \omega, \tau, \gamma, \mu_0, \mu_1 : \right.$$ 

$$\prod_{k=0}^{1} e \left( \prod_{l=1}^{\ell} W'_l \cdot \prod_{(i,j) \in E' \setminus E'} W'_{(i,j)} \cdot X_{0k} \right) e(H, X_2 X'_{1}^{l}, X_0) = e(v', X) \left\} \right.$$
where \( \{ W'_i, W'_{i,j} \} \) and \( W'_1, \ldots, W'_\ell \) are public inputs. The correctness can be verified from the following:

\[
e(W'_1, X'_{0_2}) \cdot (X'_{0_2})^{e_{1,2}} X'_{0_1} (X'_{0_2})^{e_{1,0}} \cdot \ldots \cdot e(W'_{\ell-1}, X'_{0_2}) \cdot (X'_{0_2})^{e_{1,2}} X'_{0_1} (X'_{0_2})^{e_{1,0}} \cdot e(W'_\ell, X'_{0_2}) \cdot (X'_{0_2})^{e_{1,2}} X'_{0_1} (X'_{0_2})^{e_{1,0}} = e(W'_1, X'_{0_2}) \cdot (X'_{0_2})^{e_{1,2}} X'_{0_1} (X'_{0_2})^{e_{1,0}} \cdot e(W'_1, X'_{0_2}) \cdot (X'_{0_2})^{e_{1,2}} X'_{0_1} (X'_{0_2})^{e_{1,0}} \cdot e(W'_1, X'_{0_2}) \cdot (X'_{0_2})^{e_{1,2}} X'_{0_1} (X'_{0_2})^{e_{1,0}}
\]

which is an extension of the original connected proof. This optimization reduces the number of pairings used to represent the connection from \( \ell \) to only 3 pairings.

### A.11 isolated \((i, j)\)

Let \( E' = ((i^*, j), (i, j^*)) \), the optimized isolated proof is as below:

\[
PK \left\{ \left( \forall i \in V : e_{i_0, i_{1_1}}, (\forall (i, j) \in E : e_{(i, j)_0}, e_{(i, j)_1}), \right) \right. \\
\left( i_{1_3}, i_{2_1}, \ldots, i_{\ell_1} \right), (\forall i, j) \in E : e_{(i, j)_0}, e_{(i, j)_1}, \ldots, e_{(i, j)_\ell}, \xi, \rho, \omega, \tau, \gamma, \right.
\mu_0, \mu_1, \mu_2, \{ \alpha_k \}_{k=0}^{\min(\{ E' \}, |E'|) - 1} ;
\left. \prod_{i \in V} e \left( W'_i, \prod_{k=0}^{1} X'_{i_{1_k}} \right) \prod_{k=0}^{2} e \left( \prod_{(i, j) \in E} W'_{\ell_{(i, j)_k}}, X'_{i_{2_k}} \right) \right.
\left. \left( h^{\xi} b^{\rho} c^{\omega} v^\tau, X_0 \right) = e(v^\gamma, X) \land \right.
\left. \left( \forall i^* \in E_{i^*}, j^* \in E_{j^*} \land \right) \phi_{\text{disjoint}(E_{i^*})}(E_{i^*}) \right\}
\]

74
and the expanded form is as follows:

\[
PK \left\{ \left( \forall i \in V : \varepsilon_{i_0}, \varepsilon_{i_1}, \varepsilon(i,i) \right), \left( \forall (i,j) \in E : \varepsilon(i,j)_0, \varepsilon(i,j)_1, \varepsilon(i,j)_2 \right), \zeta, \rho, \omega, \tau, \gamma, \nu_0, \nu_1, \nu_2; \right. \\
\left. \{ a_k \}_{k=0}^{\min(|E_1|,|E_2|)-1} : \right. \\
\prod_i e \left( W^r_1, \prod_{k=0}^{m} W^{\varepsilon_{i_k}}_1 \right) \prod_{k=0}^{m} e \left( \prod_{(i,j) \in E} W^{\varepsilon(i,j)}_1 \right) \cdot \\
\prod_{(i,j) \in E} e \left( h^{(i,j)}_0, X_0 \right) = e(h^{(i,j)}_0, X_0) \land \\
\prod_{l=1}^{\ell_{i_0}} e \left( \prod_{k=0}^{m} W^{v_{l_k}}_{E_{l_k}}, X_0 \right) = e \left( W^r_{E_{l_k}}, X_0 \right) \land \\
\prod_{l=1}^{\ell_{i_1}} e \left( \prod_{k=0}^{m} W^{v_{l_k}}_{E_{l_k}}, X_0 \right) = e \left( W^r_{E_{l_k}}, X_0 \right) \land \\
\prod_{l=1}^{\ell_{i_2}} e \left( \prod_{k=0}^{m} W^{v_{l_k}}_{E_{l_k}}, X_0 \right) = e \left( W^r_{E_{l_k}}, X_0 \right) \land \\
\prod_{l=1}^{\ell_{i_0}} e \left( \prod_{k=0}^{m} W^{v_{l_k}}_{E_{l_k}}, X_0 \right) = e \left( W^r_{E_{l_k}}, X_0 \right) \land \\
\prod_{l=1}^{\ell_{i_1}} e \left( \prod_{k=0}^{m} W^{v_{l_k}}_{E_{l_k}}, X_0 \right) = e \left( W^r_{E_{l_k}}, X_0 \right) \land \\
\prod_{l=1}^{\ell_{i_2}} e \left( \prod_{k=0}^{m} W^{v_{l_k}}_{E_{l_k}}, X_0 \right) = e \left( W^r_{E_{l_k}}, X_0 \right) \land \\
1 \neq \prod_{k=0}^{m} a_k \land \\
\right\}
\]
If $|E_j^*| > |E_i^*|$, the last three statements are changed to:

$$e(W_{E_i^*}^r, X_0) = e(W_{E_i^*}^r, W_i^r) e\left(\prod_{k=0}^{[E_i^*]-1} a_{0_k}^{\alpha_k}, X_0\right) \wedge$$

$$1_{\otimes} \neq \prod_{k=0}^{[E_i^*]-1} a_{0_k}^{\alpha_k}.$$