Batching Anonymous and Non-Anonymous Membership Proofs for Blockchain Applications

Shihui Fu
University of Waterloo
Waterloo, ON, Canada
shihui.fu@uwaterloo.ca

Guiwen Luo
University of Waterloo
Waterloo, ON, Canada
guiwen.luo@uwaterloo.ca

Guang Gong, IEEE Fellow
University of Waterloo
Waterloo, ON, Canada
ggong@uwaterloo.ca

***Abstract*—Membership proof is a very useful building block for checking if an entity is in a list. This tool is widely used in many scenarios. For instance in blockchain where checking membership of an unspent coin in a huge set is necessary, or in the scenario where certain privacy-preserving property on the list or on the entity is required. When it comes to multi-user applications, the naive way that verifies the membership relations one by one is very inefficient. In this work, we utilize subvector commitment schemes and non-interactive proofs of knowledge of elliptic curve discrete logarithms to present two batched membership proofs for multiple users, i.e., batched non-anonymous membership proofs and batched anonymous membership proofs, which offer plausible anonymity assurance respectively on the organization group list and on the users when combined within the blockchain applications. The non-anonymous membership proof scheme requires a trusted setup, but its proof size is only one bilinear group element and is independent of both the size of list and the number of users. The anonymous membership proof scheme requires no trusted setup, and its proof size is linear in the size of organization group and is independent of the number of users. Their security relies respectively on the CubeDH and the discrete logarithm assumptions. Finally, as a use-case application scenario, we extend Mesh which is a blockchain based supply chain management solution to Mesh$^+$ which supports batched anonymous membership proofs.**

***Index Terms*—anonymous, batched membership proof, blockchain, IoT, supply chain**

I. Introduction

Blockchain is an innovative system that enables various applications in a decentralized, secure and transparent environment, such as supply chain managements, transaction executions and so on. Transaction transparency is the powerhouse of trust in blockchains, and in the same time a restraining factor for applications that require complete or even some degree of privacy (e.g., healthcare, banking, and supply chain management applications). Generally, transaction privacy can be divided into two parts: 1) the anonymity of participating entities, and 2) the confidentiality of the transacted information. In some real world applications, however, complete anonymity is usually undesirable. Instead, some level of accountability is often required, for instance, a healthcare application where physicians are updating patient records or insurance claims that are submitted to smart contracts by claimers needs the entities involved in the respective blockchain application to be identified. In such a scenario where transaction parties are identified (e.g., through certified credentials), confidentiality of information can be achieved through membership proofs where only these users whose membership relations are validated can access to the related service. All those scenarios require some kinds of membership proofs.

Membership proof is the problem to check whether an entity belongs to a public or private list. It has been showed that membership proof is a very useful building block employed widely in IoT-level applications. This primitive protocol is already used in many scenarios, with usage scenarios ranging from the well-known cryptocurrency like Bitcoin to the current emerging healthcare systems and so on. For instance in blockchain where checking the membership of an unspent coin in a huge set is necessary, or in the scenario where certain privacy-preserving property on the set or on the entity is required.

Apart from efficiency, some level of privacy is often required in many scenarios where the privacy of the entire list is more important, especially in healthcare area. For example, an organization or group (e.g., hospital, pharmacy, clinic) in healthcare needs to prove to the third arbitration that some entities or patients belong to the list of, say, suffering from some kinds of diseases in the event of a dispute. Certainly, in this case we do not want to reveal any information about the entire list (of patients) except for the controversial members.

Membership proof plays a very important role in the context of blockchain, mainly in the design of encrypted cryptocurrency, where the privacy protection of user identities is the future trend. For example, in Bitcoin, the blockchain should maintain a collection of “unspent transaction outputs” (UTXO) [1] to indicate those coins that are eligible for future transactions. Checking a coin x is in the set UTXO is mandatory before validating a transaction spending x. The same settings are given in Zcash cryptocurrency [5]. In Zcash, checking the membership of the unspent coins is done in a similar way but without revealing both the participating entities and the coin of transaction, which needs a zero-knowledge version of membership proof.

More recently, AlTawy and Gong [2] presented Mesh, a blockchain based supply chain management solution. By adding a ciphertext that encrypts the public identity of an authorized member and a membership proof string certificate that encrypts the valid membership relation in a transaction, Mesh can provide the participating members with local anonymity related to the organization group that they belong to. Therefore, the smart contract that manages the supply chain can verify the membership relations of a given authorized organization without knowing the identity of the members. However, the performance of the entire system is rather poor, since for each transaction, the blockchain system needs to communicate with the authentication server for validating the proof of membership.

A naive way to prove membership relations is to check if an element is in the list by going through all the elements. But this method has two obvious shortcomings: The first one is that the efficiency of this method is insufferable in many scenarios, especially when the size of the list or the number of relationships that need to be proved is very large. The second is that it does not have any privacy guarantees. These shortages are especially undesirable in blockchain applications where validator nodes (or clients) should run fast and some participants must be anonymous. In this work, we focus on the following three issues:

- **Scalability**: Is it possible to check that ℓ elements are part of a list S without having to checking the membership relations one by one?
- **Member privacy**: Is it possible to check that one or more elements x's are in a public list S for some encrypted elements or without revealing any information about x's?
- **List privacy**: Is it possible to check that one or more known elements x's are in S without revealing any information about the other elements in S?

A. Our contributions

In this work, we present two batched membership proofs for multiple users, i.e., batched non-anonymous membership proofs and batched anonymous membership proofs, which offer plausible anonymity assurance respectively on the organization group and on the users when combined within the blockchain applications.

– *Batched non-anonymous membership proof.* By using a subvector commitment scheme introduced in [16], we propose a batched membership for multiple users, which offers privacy assurance on the list of members in the blockchain applications. In particular, we can check that one or more known users are in a private list in the same time without revealing any information about the other members in the list. The proof is only one bilinear group element, and is constant both in the size of list and the size of users. Furthermore, the proposed scheme supports to update the list members dynamically. The scheme requires a trusted setup to generate the public parameters and its security relies on the CubeDH assumption in the bilinear group model.

– *Batched anonymous membership proof.* By utilizing non-interactive proofs of knowledge of discrete logarithms introduced in [12], [18], [20], we propose a batched membership for multiple users, which offers privacy assurance on the users when combined within the blockchain applications. In particular, we can check that one or more private users are in a public list in the same time without revealing any information about the users' identities. The total proof is $n+9$ elliptic curve group elements, two base points and n public parameters where n is the organization group size. And the proof size is independent of the number of users. The n public parameters are simply chosen uniformly at random, thus the proof scheme requires no trusted setup and its security relies on the discrete logarithm assumption in the elliptic curve group.

– *An improved supply chain system Mesh$^+$.* We briefly mention the application of our batched anonymous membership proofs to Mesh [2]. The detailed analysis of performance of Mesh with batched anonymous membership proofs will be given in the full paper.

B. Related Work

In recent years, zero-knowledge succinct non-interactive argument of knowledge (zkSNARK) has attracted much interest from theory to practice. A lot of research work on zkSNARK for general NP statements in the form of arithmetic circuit satisfiability has been done [5], [6], [8], [9], [13]. However, most of them cannot be directly deployed in practical applications. Since the arithmetic circuit in practical applications may be very large, zkSNARKs for general NP statements are impractical due to the large communication overhead or the large computational workload. For example, a common shortage of most universal proof systems is that the proving time is always several magnitudes slower than the verification time, especially when proving the membership of many elements. These approaches do not natively support a batched operation (that is, proving the membership of multiple elements at the same time).

By using a general-purpose zkSNARK, a solution with a constant-size proof can be obtained based on the Merkle trees [17]. Specifically, we can arrange the list (size n) into a Merkle tree, and then to prove the membership relations of the elements in the list is equivalent to prove that there is a valid path connecting the claimed element on the leaf to the Merkle root. This needs to prove the correctness of about $\log n$ hash calculations. Thus, a proof of constant size will be generated if a preprocessing zkSNARK is used, such as [14], [19]. However, this solution also does not support the batched operations.

Another solution is the protocol proposed in [10]. Specifically, the authors presented a zero-knowledge version of membership proof of an RSA accumulator. However, as described in [4], if we want to have a high-level security, it requires a large group of prime order at least $q > 2^{519}$. For efficiency reasons in practice, we may not want to have such a large prime order group [4]. So the efficiency and flexibility are greatly reduced in practice.

II. Preliminaries

Throughout this work we write $[n]$ to denote the set $\{1, 2, \ldots, n\}$ and $\{x_i\}_{i\in[n]}$ to denote the set $\{x_1, \ldots, x_n\}$.

We use $\lambda \in \mathbb{N}$ to denote the security parameter, and $\mathsf{negl}(\lambda)$ the set of negligible functions in λ. The larger the value of λ, the higher the security we have. We denote the process of selecting s uniformly at random from a given set S by $s \leftarrow_R S$, and the output of an algorithm A on input x by $y \leftarrow A(x)$.

A. Subvector Commitments

A commitment scheme is a very important cryptographic primitive, which allows one to commit to a selected value, while maintaining the concealment of other values and also being able to reveal the committed value in the future. In this subsection, we briefly introduce the notions of subvector and subvector commitment. Loosely speaking, a subvector is an ordered subset of a given vector, and subvector commitments are a generation of vector commitment schemes [11] where the opening algorithms (see the definition below) are performed with respect to subvectors. We follow the notations of subvector commitments from Lai and Malavolta [16].

Definition 1 (Subvectors [16]): Let $n \in \mathbb{N}$ be a positive integer, Γ be a set, and $\boldsymbol{x} = (x_1, \dots, x_n) \in \Gamma^n$ be a vector. Let $J = (i_1, \dots, i_{|J|}) \subseteq [n]$ be an ordered subset. The J-subvector of $\boldsymbol{x}$ is defined as the vector $\boldsymbol{x}_J := (x_{i_1}, \dots, x_{i_{|J|}})$.

Definition 2 (Subvector commitments [16]): A subvector commitment scheme is a tuple of four subprotocols $\mathsf{SVC} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open}, \mathsf{Verify})$:

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n; \omega)$: Takes as input the security parameter 1^λ, the vector size 1^n along with randomness ω, and outputs the public parameters pp which will be used by all parties. All the remaining algorithms are assumed to input implicitly pp which we will omit.
- $(\mathcal{C}, \mathcal{S}) \leftarrow \mathsf{Commit}(\boldsymbol{x})$: Takes as input a vector $\boldsymbol{x} \in \Gamma^n$ and outputs a commitment $\mathcal{C}$ and the secret opening hint $\mathcal{S}$.
- $\Lambda_J \leftarrow \mathsf{Open}(J, \boldsymbol{x}'_J, \mathcal{S})$: Takes as input an index subset $J \subseteq [n]$, a J-subvector $\boldsymbol{x}'_J$, and the secret opening hint $\mathcal{S}$ and outputs a proof Λ_J that $\boldsymbol{x}'_J$ is the J-subvector of the committed vector $\boldsymbol{x}$.
- $b \in \{0,1\} \leftarrow \mathsf{Verify}(\mathcal{C}, J, \boldsymbol{x}'_J, \Lambda_J)$: Take as input a commitment $\mathcal{C}$, an index subset $J \subseteq [n]$, a J-subvector $\boldsymbol{x}'_J$ along with a proof Λ_J. It outputs whether $\mathcal{C}$ is a commitment to the vector $\boldsymbol{x}$ and $\boldsymbol{x}'_J$ is the J-subvector of $\boldsymbol{x}$.

A subvector commitment scheme is correct if an honest committer can successfully convince the verifier of any index subset $J \subseteq [n]$ opening, and is compact if both the sizes of the commitment string $\mathcal{C}$ and the proof string Λ_J are independent of the sizes of the committed vector $\boldsymbol{x}$ and the index subset J. A subvector commitment scheme is position binding if no efficient adversary can convince the verifier that the committed vector $\boldsymbol{x}$ opens to different index set - subvector pairs $(J, \boldsymbol{x}_J)$ and $(I, \boldsymbol{x}'_I)$ such that there exists a common index $i \in I \cap J$ satisfying $x_i \neq x'_i$.

As we will use subvector commitment schemes to construct our non-anonymous membership proofs, it is not needed to define hiding for our purpose.

B. Proofs of Knowledge

In this subsection, we define the notion of proofs of knowledge, concretely, proof of representation [18], proof of equality [12], and proof of existential equality [20] in the discrete logarithms setting. Roughly speaking, a string is of knowledge if the prover can convince the verifier not only that the secret exists, but that he in fact knows such a secret.

Definition 3 (Proof of knowledge [21]): A proof is of knowledge if there exists a probabilistic polynomial time algorithm $\mathcal{E}$ such that, for any input $x \in \{0,1\}^*$ and proof $\pi \in \{0,1\}^*$ with $\Pr[\mathcal{V}^\pi(x) = 1] > \mathsf{negl}(\lambda)$, $\mathcal{E}^\pi(x)$ can extract an NP witness w for x with a non-negligible probability.

In the following, we use $\mathsf{PoK}\{x : \text{conditions}\}$ to denote the proof of knowledge of the witness x satisfying *conditions*. If not explicitly notated, we assume all base points and logarithms are on a group of prime order q and $\mathcal{H}$ is a collision resistant hash function.

Definition 4 (Proof of representation [18]): A message-dependent proof of knowledge of the EC discrete logarithms representation of y to the base points $(g_1, \dots, g_n)$ where $1 \leq n \leq q-2$, is given by $\mathsf{PoK}\{(x_1, \dots, x_n) : y = \sum_{i=1}^n x_i g_i\}$. The proof is the tuple $(r, s_1, \dots, s_n)$ and is generated non-interactively by the prover as follows:

- For each secret discrete logarithm x_i, compute
$$v_i \leftarrow_R \mathbb{Z}_q, \quad t_i := v_i g_i.$$
- For the message m, compute
$$r := \mathcal{H}\left(m, g_1, \dots, g_n, y, \sum_{i=1}^n t_i\right).$$
- For each secret discrete logarithm x_i, compute
$$s_i := v_i + r x_i.$$

Only the prover who knows a representation $y = \sum_{i=1}^n x_i g_i$ can construct the proof. It is showed [18], [20] that this proof is zero-knowledge about x_i, $1 \leq i \leq n$. The verification equation is
$$\mathcal{H}\left(m, g_1, \dots, g_n, y, \sum_{i=1}^n s_i g_i - r y\right) \stackrel{?}{=} r.$$

Definition 5 (Proof of equality [12]): A message-dependent proof of knowledge that the EC discrete logarithms of y_i to the base point g_i are equal for all $i \in \{1, \dots, n\}$ where $1 \leq n \leq q-2$, is given by $\mathsf{PoK}\{x : y_i = x g_i \text{ for all } i \in [n]\}$. The proof is the tuple (r, s) and is generated non-interactively by the prover as follows:

- Choose $v \leftarrow_R \mathbb{Z}_q$ and for each $i \in [n]$, compute
$$t_i := v g_i.$$
- For message m, compute
$$r := \mathcal{H}\left(m, \{g_i\}_{i \in [n]}, \{y_i\}_{i \in [n]}, \{t_i\}_{i \in [n]}\right).$$
- Compute
$$s := v + r x.$$

Only the prover who knows the discrete logarithms $\log_{g_i} y_i, i \in [n]$ and all of them are equal, can construct the proof and convince the verifier. It is showed [12], [20] that this proof is also zero-knowledge. It does not leak any information about x. The verification equation is

$$\mathcal{H}\left(m, \{g_i\}_{i\in[n]}, \{y_i\}_{i\in[n]}, \{sg_i - ry_i\}_{i\in[n]}\right) \overset{?}{=} r.$$

Next, we introduce the proof of knowledge that there exists at least one of the n discrete logarithms ($\log_{g_1} y_i$'s) which is equal to another logarithm ($\log_{g_2} z$). To simplify notations, in the following definition we assume that the relation holds for $i = 1$, e.g., the prover knows that $x = \log_{g_1} y_1 = \log_{g_2} z$ and wants to convince the verifier that but without revealing for which i this relation holds.

Definition 6 (Proof of existential equality [20]): A message-dependent proof of knowledge that there exists one EC discrete logarithm of the y_i, $i \in [n]$ to the base g_1 is equal the discrete logarithm of z to the base g_2, without revealing any information about which y_i this relation holds, is given by $\mathsf{PoK}\{x : z = xg_2 \text{ and } \exists\, i \in [n] \text{ s.t. } y_i = xg_1\}$. The proof is the tuple $(r_1, \ldots, r_n, s_1, \ldots, s_n)$ and is generated non-interactively by the prover as follows:

- Choose $v_i \leftarrow_R \mathbb{Z}_q$ for each $1 \le i \le n$ and $r_i \leftarrow_R \mathbb{Z}_q$ for each $2 \le i \le n$, and compute
  - $c_1 := v_1 g_1$ and $c_i := v_i g_1 - r_i y_i$ for $2 \le i \le n$,
  - $d_1 := v_1 g_2$ and $d_i := v_i g_2 - r_i z$ for $2 \le i \le n$.
- For message m, compute
  - $r := \mathcal{H}(m, g_1, g_2, \{y_i\}_{i\in[n]}, z, \{c_i\}_{i\in[n]}, \{d_i\}_{i\in[n]})$.
  - $r_1 := r - \sum_{i=2}^{n} r_i$.
- Compute
  - $s_1 := v_1 + r_1 x$ and $s_i := v_i$ for $2 \le i \le n$.

It is showed that this proof does not reveal any information about which of the n witnesses the prover knows [20]. The verification equation is

$$\mathcal{H}\left(m, g_1, g_2, \{y_i\}_{i\in[n]}, z, \{s_i g_2 - r_i y_i\}_{i\in[n]},\right.$$
$$\left.\{s_i g_1 - r_i z\}_{i\in[n]}\right) \overset{?}{=} \sum_{i=1}^{n} r_i.$$

Noticed that all three proofs of knowledge are message-dependent, we can remove the message m from the computations of hash challenges to make the definition message independent if necessary.

III. Non-Anonymous Membership Proofs

In this section, we describe our non-anonymous membership proof schemes for multiple users. Formally, given one organization group with n members $\{\mathsf{ID}_1, \ldots, \mathsf{ID}_n\}$, for any index set $I \subseteq [n]$ we want to prove that $\mathsf{ID}'_i \in \{\mathsf{ID}_1, \ldots, \mathsf{ID}_n\}$ for any $i \in I$ but without revealing any information about other members ID_j for $j \in [n] \setminus I$.

We propose our direct construction from bilinear groups where the CubeDH assumption holds. Our construction is inspired by the work of Lai and Malavolta [16]. Without loss of generality, we assume our ID space $\mathcal{ID}$ is $\mathbb{Z}_q$. For a more general ID space, we can use a collision resistant hash function to hash the IDs to $\mathbb{Z}_q$.

- $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^n; \omega)$. This setup algorithm takes as input the security parameter 1^λ, the organization group size 1^n, and a randomness ω. It outputs the public parameters pp as follows.
  - $(q, \mathbb{G}, \mathbb{G}_T, G, e) \leftarrow \mathsf{GGen}(1^\lambda; \omega)$ where $\mathbb{G}, \mathbb{G}_T$ are multiplicative groups of prime order q, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a non-degenerate bilinear map, and G is a generator for $\mathbb{G}$.
  - $\forall\, i \in [n], z_i \leftarrow_R \mathbb{Z}_q$.
  - $\forall\, i, i' \in [n]$, $G_i := G^{z_i}$, $H_{i,i'} := G^{z_i z_{i'}}$.
  - $\mathsf{pp} := \left(q, \mathbb{G}, \mathbb{G}_T, G, \{G_i\}_{i\in[n]}, \{H_{i,i'}\}_{i,i'\in[n], i\neq i'}, e\right)$.
- $(\mathcal{C}, \mathsf{aux}) \leftarrow \mathsf{Commit}(\boldsymbol{x})$: The committing algorithm inputs an ordered organization group public ID vector $\mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_n) \in \mathbb{Z}_q^n$. It outputs a commitment string $\mathcal{C}$ and some secret opening hint aux (auxiliary information in Definition 2) as follows.
  - $\mathcal{C} := \prod_{i\in[n]} G_i^{\mathsf{ID}_i}$.
  - $\mathsf{aux} := \mathsf{ID} = (\mathsf{ID}_1, \ldots, \mathsf{ID}_n)$.
- $(\mathcal{C}', \mathsf{aux}') \leftarrow \mathsf{UpdateCommit}(\mathcal{C}, J, \mathsf{ID}_J, \mathsf{ID}'_J)$: The updating commitment algorithm inputs a comitment $\mathcal{C}$, an index subset $J \subseteq [n]$, two vectors of length $|J|$. It updates the positions in J from ID_J to ID'_J to produce a new commitment $\mathcal{C}'$ and opening hint information aux' as follows.
  - $\mathcal{C}' := \mathcal{C} \cdot \prod_{i\in J} G_i^{\mathsf{ID}'_i - \mathsf{ID}_i}$.
  - aux' is the updated organization group ID vector.

Noted that for an organization group with n members, the commitment $\mathcal{C}$ is only one bilinear group element and independent of the size n of the organization group. If the members in the organization group are unchanged, the commitment $\mathcal{C}$ is also unchanged. Furthermore, we can also use $\mathsf{UpdateCommit}$ to update the IDs of some users in the organization group dynamically and efficiently.

Now given an index set $I \subseteq [n]$ with size $1 \le \ell \le n$ and a vector ID'_I, the proof of membership relations that $\mathsf{ID}'_i = \mathsf{ID}_i$ for any $i \in I$ is the bilinear group element Λ_I and is generated by the prover as follows.

- $\Lambda_I \leftarrow \mathsf{Open}(I, \mathsf{ID}'_I, \mathsf{aux})$: The opening algorithm takes as input an index subset $I \subseteq [n]$, an ID vector ID'_I of length $|I|$ along with the secret opening hint aux (auxiliary information). It outputs a proof Λ_I that $\mathsf{ID}'_i = \mathsf{ID}_i$ for any $i \in I$ as follows.
  - $\Lambda_I := \prod_{i\in I} \prod_{i'\notin I} H_{i,i'}^{\mathsf{ID}_{i'}}$.

The prover sends the proof Λ_I to the verifier, and the verifier does the following verification.

- $b \in \{0,1\} \leftarrow \mathsf{Verify}(\mathcal{C}, I, \boldsymbol{x}'_I, \Lambda_I)$: The verification algorithm takes as input a commitment string $\mathcal{C}$, an index subset I, an ID vector ID'_I of length $|I|$ along with a proof Λ_I. It outputs 1 (i.e., it accepts) if and only if the following conditions hold.
  - $\mathsf{ID}'_i \in \mathbb{Z}_q$ for each $i \in I$,

– $e\left(\frac{\mathcal{C}}{\prod_{i\in I} G_i^{\mathsf{ID}'_i}}, \prod_{i\in I} G_i\right) = e(\Lambda_I, G)$.

The proof alone is only one bilinear group element and is constant both in the size n of the organization group and the size ℓ of users. We refer to [16] for a detailed security proof.

IV. Anonymous Membership Proofs

In this section, we consider the anonymous membership proofs for multiple users, which means that we want to prove some users are members of one organization group but without revealing for whom this membership relation holds. This is to say that if we have ℓ entities who are claimed to be in an organization group, we need to generate a batched proof for the following statements:

- Each element (see c_i below) encrypts a valid identity form (e.g., $\mathsf{ID}'_i = x_i\alpha$ where α is one primitive base point) for $1 \le i \le \ell$;
- The encrypted identities are members of the organization group, i.e., $\mathsf{ID}'_i \in \{\mathsf{ID}_1, \dots, \mathsf{ID}_n\}$ for $1 \le i \le \ell$.

First, we define the public identity ID_i's generation and encryption schemes where public key encryption is performed using ElGamal scheme over EC [15]. We assume that the public parameters which consist of the chosen elliptic curve, its order q, two primitive points α and β, and the organization group's public encryption key y_m where $y_m = x_m\beta$, are shared by all entities. Here $x_m \in_R \mathbb{Z}_q$ is the organization group's secret encryption key.

1. **Identity gereration** IDGen. This identity generation process is run by each member to generate their public identity ID_i that is registered at the organization group manager and stored in the organization group server.
$(x_i, \mathsf{ID}_i) \leftarrow \mathsf{IDGen}(1^\lambda)$:
$$x_i \leftarrow_R \mathbb{Z}_q$$
$$\mathsf{ID}_i := x_i\alpha.$$
2. **Identity encryption** IDEnc. Because of anonymity, the public identity is needed to be encrypted using ElGamal scheme under the organization group's public encryption key y_m.
$(c_{i,1}, c_{i,2}) \leftarrow \mathsf{IDEnc}(x_i, y_m)$:
$$k_i \leftarrow_R \mathbb{Z}_q$$
$$c_{i,1} := k_i\beta$$
$$c_{i,2} := k_i y_m + x_i\alpha = k_i y_m + \mathsf{ID}_i.$$

Now a member can sign a blockchain transaction by first hashing the message and the ciphertext then signing them by the shared organization group's signature key [2].

In the sequel, we assume the ℓ claimed organization group members have public identities $\mathsf{ID}'_1, \dots, \mathsf{ID}'_\ell$, respectively. Formally, the anonymous membership proof scheme for multiple entities is defined as follows.

Definition 7 (Multiple anonymous membership proofs): Let ℓ, n be two positive integers and $\ell \le n$. Given ℓ encrypted identities $c_i = (c_{i,1}, c_{i,2})$, where $c_{i,1} = k_i\beta$, $c_{i,2} = k_i y_m + \mathsf{ID}'_i$ and $\mathsf{ID}'_i = x_i\alpha$ for $1 \le i \le \ell$, a multiple anonymous membership proof that c_i's encrypt ℓ valid member identities from a set of identities $\{\mathsf{ID}_1, \dots, \mathsf{ID}_n\}$ is given as

$$\mathsf{PoK}\left\{(k_1,\dots,k_\ell,x_1,\dots,x_\ell) : \forall\, i \in [\ell], \begin{smallmatrix} c_{i,2}=k_i y_m + x_i\alpha \\ \text{and } c_{i,1}=k_i\beta \text{ and} \\ \exists\, j_i\in[n] \text{ s.t.} \\ c_{i,2}-\mathsf{ID}_{j_i}=k_i y_m \end{smallmatrix}\right\}.$$

When combined with the ideas of proof of representation and proof of existential equality introduced in Section II, a naive way is to verify the ℓ membership relations one by one, namely one action per item as follows.

– For $1 \le i \le \ell$, verify

$\mathsf{PoK}\{(k_i, x_i) : c_{i,2} = k_i y_m + x_i\alpha\}$ and

$\mathsf{PoK}\{k_i : c_{i,1} = k_i\beta$ and $\exists\, j_i \in [n]$ s.t. $c_{i,2} - \mathsf{ID}_{j_i} = k_i y_m\}$.

It is easy to see that this scheme is very inefficient, in that the prover needs to generate ℓ proofs of representation and ℓ proofs of existential equality, and the verifier needs to verify 2ℓ equations for ℓ membership relations. The proof size is at least $(2n+3)\cdot\ell$ elliptic curve group elements.

To solve these problems more efficiently, we introduce two additional "random" scalar values t_1 and t_2 into the above scheme to combine ℓ items to a single item. Concretely, let

$$t_1 = \mathcal{H}\left(m, \alpha, \beta, y_m, \{\mathsf{ID}_i\}_{i\in[n]}, \{c_{i,1}\}_{i\in[\ell]}\right),$$
$$t_2 = \mathcal{H}\left(m, \alpha, \beta, y_m, \{\mathsf{ID}_i\}_{i\in[n]}, \{c_{i,2}\}_{i\in[\ell]}\right).$$

For the ℓ proofs of representation $\mathsf{PoK}\{(k_i, x_i) : c_{i,2} = k_i y_m + x_i\alpha\}$, we can instead do the following batched proof.

$$\mathsf{PoK}\Bigg\{(k_1,\dots,k_\ell,x_1,\dots,x_\ell) : \sum_{i=1}^{\ell} t_1^i c_{i,2} = \left(\sum_{i=1}^{\ell} t_1^i k_i\right) y_m + \left(\sum_{i=1}^{\ell} t_1^i x_i\right)\alpha\Bigg\}.$$

Now the adversity is bound to $\sum_{i=1}^{\ell} t_1^i c_{i,2}$, $\sum_{i=1}^{\ell} t_1^i k_i$ and $\sum_{i=1}^{\ell} t_1^i x_i$. A standard argument (Schwartz-Zippel Lemma) tells us that, for example, for any $\{x_i : i \in [\ell]\} \ne \{x'_i : i \in [\ell]\}$, it holds

$$\Pr_{t_1 \leftarrow_R \mathbb{Z}_q}\left[\sum_{i\in[\ell]} t_1^i x_i = \sum_{i\in[\ell]} t_1^i x'_i\right] \le \frac{\ell}{q}.$$

Therefore, if we model the hash function $\mathcal{H}$ as a random oracle [3], the soundness is at most $3\ell/q$ by the union bound, which is negligible in λ.

By the definition 4 of proof of representation, the proof consists of 3 elliptic curve group elements and is constant in the size n of the organization group. Now let us look at the second proof of existential equality. We can also follow the previous idea and do the following batched proof.

$$\mathsf{PoK}\Bigg\{(k_1,\dots,k_\ell) : \sum_{i=1}^{\ell} t_2^i c_{i,1} = \left(\sum_{i=1}^{\ell} t_2^i k_i\right)\beta \text{ and } \exists\, j_1,\dots,j_\ell \in [n] \text{ s.t. } \sum_{i=1}^{\ell} t_2^i c_{i,2} - \sum_{i=1}^{\ell} t_2^i \mathsf{ID}_{j_i} = \left(\sum_{i=1}^{\ell} t_2^i k_i\right) y_m\Bigg\}. \quad (1)$$

Let us once again look carefully at the definition 6 of proof of existential equality. A natural problem is that we cannot directly apply the scheme of proof of existential equality to our case. The reason is that existential equality captures the case where the logarithm of *one* of the y_i, $i \in [n]$ to one base point is equal to the discrete logarithm of z to another base point. However, in our case we need to prove that there exist ℓ entries out of n members that satisfy some given equations. Of course, a naive way is to encode all the possibilities of ℓ ordered tuples out of the n members. But this is unacceptable as the total number (ℓ-permutations out of n) of possibilities is very large.

Instead of using a proof of existential equality, we consider the proof of equality. Concretely, let

$$y = \sum_{i=1}^{\ell} t_2^i \mathsf{ID}_{j_i},$$

then we can rewrite (1) as

$$\mathsf{PoK}\left\{ (k_1, \ldots, k_\ell) : \sum_{i=1}^{\ell} t_2^i c_{i,1} = \left(\sum_{i=1}^{\ell} t_2^i k_i \right) \beta \text{ and } \sum_{i=1}^{\ell} t_2^i c_{i,2} - y = \left(\sum_{i=1}^{\ell} t_2^i k_i \right) y_m \right\}. \quad (2)$$

From Definition 5 (proof of equality), the proof alone is of length 2 elliptic curve group elements and is constant in the size n of the organization group as well.

Of course, we must still prove that $y = \sum_{i=1}^{\ell} t_2^i \mathsf{ID}_{j_i}$. Let $\boldsymbol{w}$ be a vector of length n where the j_i-th entity is equal to t_2^i for $i \in [\ell]$ and 0 elsewhere. Therefore, y can be regarded as the inner product of $\boldsymbol{w}$ and the public ordered vector $\mathsf{ID} := (\mathsf{ID}_1, \ldots, \mathsf{ID}_n)$. Equivalently, we need to prove the relation $y = \langle \boldsymbol{w}, \mathsf{ID} \rangle$ without revealing any information about $\boldsymbol{w}$. In the following we describe how to do that.

First, we need a setup algorithm to generate n public parameters,

$$(e_1, \ldots, e_n) \leftarrow_R \mathbb{Z}_q^n.$$

Now, the proof is the tuple $(r, u, s, z_1, \ldots, z_n)$ and is generated non-interactively by the prover as follows:

- Choose $v_i \leftarrow_R \mathbb{Z}_q$ for each $1 \le i \le n$, $\rho_1 \leftarrow_R \mathbb{Z}_q$, $\rho_2 \leftarrow_R \mathbb{Z}_q$ and compute
  - $u := \sum_{i=1}^{\ell} t_2^i e_{j_i} \alpha + \rho_1 \alpha$,
  - $d_1 := \sum_{i=1}^{n} v_i \mathsf{ID}_i$,
  - $d_2 := \sum_{i=1}^{n} v_i e_i \alpha + \rho_2 \alpha$.
- For message m, compute
  - $r := \mathcal{H}(m, \alpha, y, t_2, \{\mathsf{ID}_i\}_{i \in [n]}, \{e_i\}_{i \in [n]}, u, d_1, d_2)$, where $\mathcal{H}$ is a collision resistant hash function.
- Compute
  - $z_i := r x_i + v_i$ for $1 \le i \le n$,
  - $s := r\rho_1 + \rho_2$.

The verification equation is

$$\mathcal{H}\left(m, \alpha, y, t_2, \{\mathsf{ID}_i\}_{i \in [n]}, \{e_i\}_{i \in [n]}, u, \sum_{i=1}^{n} z_i \mathsf{ID}_i - ry, \sum_{i=1}^{n} z_i e_i \alpha + s\alpha - ru \right) \overset{?}{=} r.$$

We can show that this proof of $y = \langle \boldsymbol{w}, \mathsf{ID} \rangle$ is perfectly complete, honest-verifier zero-knowledge (without revealing any information about $\boldsymbol{w}$) and computationally sound when q is exponential larger than ℓ and n. The detailed security analysis is given in the full version. The proof alone consists of $n + 3$ elliptic curve group elements apart from the n public parameters $(e_1, \ldots, e_n)$ and is only dependent of the organization group size n.

Putting all things together, we obtain the final batched anonymous membership proof scheme. The entire proof is also perfectly complete, honest-verifier zero-knowledge (without revealing any information about the ℓ membership relations) and computationally sound when q is exponential larger than ℓ and n. The total proof size is $n + 9$ elliptic curve group elements (counting the claimed inner product y), two base points and n public elements.

As in Mesh [2], the final scheme ensures the privacy of entities, because we use ElGamal encryption scheme to encrypt the identities of entities, which is semantically safe under the Diffie-Hellman assumption [7]. More concretely, assuming that the private keys of all IDs are only known by their corresponding members, the anonymous membership proofs can ensure the valid identity (encrypted in c under y_m) of the signer in the transaction.

V. Mesh$^+$ with batched anonymous membership proofs

In this section, we use Mesh [2] as a use-case to show how our scheme of batched anonymous membership proof can be used to improve the performance greatly.

Mesh system consists of three subsystems, Mesh server, a blockchain smart contract, and RFID subsystems. Mesh server's functionality is to authorize participate parties, verify, update and keep a list of participants that are allowed to use the system as well as to revoke the parties in the system which are malicious or whose private keys are exposed. In its supply chain life cycle, a product is identified by its RFID tag and can be owned by n owners (given supply chain participants) where each owner is a member of an organization group of n members. On the blockchain, the supply chain smart contract verifies the membership proof, and if the verification is successful it stores authenticated private tracking history of the product.

In the work of Mesh [2], it mentioned that it could consider the subset of the organization group for efficiency consideration, but it did not present any mechanisms for achieving this goal. As we have mentioned in Section IV, the naive scheme which was used in Mesh cannot support a batched verification with performance better than ℓ times verifying a single membership proof.

Now we can apply our scheme of the batched anonymous membership proof system to Mesh's architecture. The resulting system is referred to as Mesh$^+$. In order to do so, we need to add one more step for selecting a subgroup leader to combine the proofs of the subgroup of ℓ participants in an organization with size $n \geq \ell$ and to generate a batched membership proof. The organization group's public-key will be the Mesh server's public-key. So, the property of locally anonymity is preserved, since Mesh server can remove this anonymity once some malicious behaviours are observed from this subgroup, and those which are malicious can be individually identified.

Since each organization group in supply chain management may be involved in many product chains, for example, for Amazon, it has more than 12 million product supply chains, Walmart, about 75 million products where Walmart has developed their food supple chain management for traceability system with Hyperledger Fabric. As the authors of Mesh [2] has reported, if the organizations would like to use public blockchain systems, such as Eutherum, the gas fees are high. So the authors recommended the organization group size n should be less than 50, since for larger organization groups $n > 50$, it runs over the current block gas limit set by Ethereum.

However, this problem is resolved in Mesh$^+$. The detailed analysis of the implementation cost of Mesh$^+$ will be given in the full paper of this work.

VI. Concluding remarks

In this work, we present two batched membership proofs for multiple users, i.e., batched non-anonymous membership proofs and batched anonymous membership proofs, which offer plausible anonymity assurance respectively on the organization list and on the users when deployed in the blockchain applications. The former only needs one bilinear group element regardless of the list size and the number of users that need to be proved, and supports to update the list members dynamically and efficiently. The latter does not need any trusted setup, and the size of the proof is also independent of the number of users that need to be proved. As a use-case scenario, we present an extended blockchain based supply chain management system Mesh$^+$, which supports a batched verification for many membership proofs in the same time.

References

[1] Bitcoin - open source P2P money. https://bitcoin.org/en/.

[2] Riham AlTawy and Guang Gong. Mesh: A supply chain solution with locally private blockchain transactions. *Proceedings on Privacy Enhancing Technologies*, 2019(3):149–169, 2019.

[3] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS 1993*, pages 62–73. ACM, 1993.

[4] Daniel Benarroch, Matteo Campanelli, Dario Fiore, and Dimitris Kolonelos. Zero-Knowledge Proofs for Set Membership: Efficient, Succinct, Modular. *Cryptology ePrint Archive*, Report 2019/1255. IACR, 2019.

[5] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *S&P 2014*, pages 459–474. IEEE Computer Society, 2014.

[6] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for C: Verifying program executions succinctly and in zero knowledge. In *CRYPTO 2013*, volume 8043 of *Lecture Notes in Computer Science*, pages 90–108. Springer, 2013.

[7] Dan Boneh. The decision diffie-hellman problem. In *Algorithmic Number Theory, 1998*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.

[8] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Jens Groth, and Christophe Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In *EUROCRYPT 2016*, volume 9666 of *Lecture Notes in Computer Science*, pages 327–357. Springer, 2016.

[9] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *S&P 2018*, pages 315–334. IEEE Computer Society, 2018.

[10] Jan Camenisch and Anna Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *CRYPTO 2002*, volume 2442 of *Lecture Notes in Computer Science*, pages 61–76. Springer, 2002.

[11] Dario Catalano and Dario Fiore. Vector commitments and their applications. In *PKC 2013*, volume 7778 of *Lecture Notes in Computer Science*, pages 55–72. Springer, 2013.

[12] Eiichiro Fujisaki and Tatsuaki Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In *CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 1997.

[13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without PCPs. In *EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 626–645. Springer, 2013.

[14] Jens Groth. On the size of pairing-based non-interactive arguments. In *EUROCRYPT 2016*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.

[15] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

[16] Russell W. F. Lai and Giulio Malavolta. Subvector commitments with application to succinct arguments. In *CRYPTO 2019*, volume 11692 of *Lecture Notes in Computer Science*, pages 530–560. Springer, 2019.

[17] Ralph C. Merkle. A certified digital signature. In *CRYPTO 1989*, volume 435 of *Lecture Notes in Computer Science*, pages 218–238. Springer, 1989.

[18] Tatsuaki Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In *CRYPTO 1992*, volume 740 of *Lecture Notes in Computer Science*, pages 31–53. Springer, 1992.

[19] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *S&P 2013*, pages 238–252. IEEE Computer Society, 2013.

[20] Holger Petersen. How to convert any digital signature scheme into a group signature scheme. In *Security Protocols 1997*, volume 1361 of *Lecture Notes in Computer Science*, pages 177–190. Springer, 1997.

[21] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In *TCC 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.