Riham AlTawy* and Guang Gong

Mesh: A Supply Chain Solution with Locally Private Blockchain Transactions

Abstract: A major line of research on blockchains is geared towards enhancing the privacy of transactions through anonymity using generic non-interactive proofs. However, there is a good cluster of application scenarios where complete anonymity is not desirable and accountability is in fact required. In this work, we utilize non-interactive proofs of knowledge of elliptic curve discrete logarithms to present membership and verifiable encryption proof, which offers *plausible anonymity* when combined with the regular signing process of the blockchain transactions. The proof system requires no trusted setup, both its communication and computation complexities are linear in the number of set members, and its security relies on the discrete logarithm assumption. As a use-case for this scenario, we present Mesh which is a blockchain-based framework for supply chain management using RFIDs. Finally, the confidentiality of the transacted information is realized using a lightweight *key chaining* mechanism implemented on RFIDs. We formally define and prove the main security features of the protocol, and report on experiments for evaluating the performance of the modified transactions for this system.

Keywords: Privacy, Blockchains, Proofs of knowledge, RFID, Supply chain, Discrete logarithm, ElGamal encryption

1 Introduction

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., health care, banking, and supply chain management applications). Generally, transactions privacy can be divided into in two parts, 1) the anonymity of participating entities, and 2) the confidentiality of the transacted information. Apart from currency transactions where complete anonymity lends itself well to cash transactions, in real world applications, complete anonymity is usually undesirable. In fact, some level of accountability is often required, for instance, a health care application where physicians are updating patients records or insurance claims that are submitted to smart contracts by claimers need the entities using the respective blockchain application to be identified. In such a scenario where transaction parties are identified (e.g., through certified credentials), confidentiality of information (not currency) can be achieved through symmetric encryption where only authorized entities have access to the encryption key.

Anonymity vs. Accountability. Identification of transacting entities is not always secure. For example, in supply chain solutions where organizations are committing tracking information to the blockchain. The identification of the submitting accounts enables tracking the goods because these accounts are associated with the physical locations of specific facilities. In this case, one needs a solution that fulfills both worlds, 1) anonymity of the sending entity to hinder tracking and b) accountability in the event of a dispute. Mesh offers a solution for this problem by utilizing membership and verifiable encryption proof based on proofs of knowledge of discrete logarithms to realize *plausible anonymity*. In such a context, plausible anonymity is defined as the property of being anonymous within a group with the possibility of being correctly identified when needed (as in group signatures). Our solution does not require changes to the transaction signing algorithm adopted by a given blockchain platform (e.g., ECDSA in Ethereum) [55], it only adds to the transaction an extra ciphertext argument and a membership proof that the ciphertext encrypts the identity of the transaction signer under an untrusted authority's public key. When needed, such anonymity can be verifiably revoked.

1.1 Related Work

Generic Proofs. Recently, there has been an exciting line of research on proofs of knowledge for general NP statements in the form of arithmetic circuit satisfiability [14, 18, 30, 37, 51]. However, most of them do not lend themselves smoothly to some applications because the resulting arithmetic circuit can be quite large so that either their communication or computation overheads

***Corresponding Author: Riham AlTawy:** University of Waterloo, E-mail: raltawy@uwaterloo.ca
Guang Gong: University of Waterlo, E-mail: ggong@uwaterloo.ca

become unpractical. For example, a common trait in most of the generic proof systems is that proving timings are always several magnitudes slower than verifying timings. In supply chains, proving is as important as verifying, given their highly dynamic nature where large volumes of products are processed, and their updates has to be committed swiftly because they may go from one owner to another in under an hour. Proving times of [14, 18, 37] range between half an hour to over 3 hours for a reasonable 2^{22} gate circuit.

Generic proofs include zkSNARKs [14] which instantiates Quadratic Arithmetic Programs (QAP) [30] and relies on the knowledge of exponent assumption (KOE) [26, 33]. zkSNARKS offer highly efficient proofs of size 288 Bytes at 128-bit security and verification time. However, they require a complicated trusted setup to generate common reference string (CRS).

A Cryptographically Strong Hash (CSH) function used to realize Scalable Computational Integrity (SCI) proof [12] with no trusted set up, however, the size of the proof and proving time are highly unpractical, roughly over 5 MB large in practice for an average size circuit. Following SCI, zkSTARK [13] offers a reduced but yet large proof size of around 200 KB at only 60-bit security for a 2^{19} gate circuit.

Bootle *et al.* [37] and Bünz *et al.* [18] proposed two related proof systems that rely on the discrete logarithm problem assumption (DLP). These two proofs are highly efficient in terms of proof size (logarithmic in circuit size), however, both proving and verifying timings are linear in the multiplicative complexity of the circuit (proving takes a minimum of 24 minutes for a circuit of 2^{22} gates [37]).

Membership Proofs. The proof adopted in Mesh turns the original blockchain signing scheme into a group signature scheme by enabling verifiable revocable anonymity. So conceptually, it cannot be directly adopted on other anonymous blockchains that offer irreversible anonymity such as ring signature schemes [50]. Particularly, in Monero, the underlying Cryptnote protocol [54] uses the Proof of partial knowledge [25] to enable a user to pickup a random group of users public key and provide a proof of knowledge that she knows a secret key corresponding to at least one of them (proof size and verification are linear in the group size. The proof itself is the signature on the transaction and hence, the notion of an authorized walled account is not available, thus, there is no accountability. On the other hand, some work on set membership proofs [21, 22, 53] can be conjuncted with verifiable encryption to generate group signatures. However, most of them are tailored towards specific applications and can not be practically adopted on public blockchains. For example, the work in [53] requires constant communication of $O(K)$ RSA-field elements for security parameter k to be sent between prover and verifier. Such a complexity is further asymptotically reduced to $O(\frac{k}{\log k - \log\log k}$ in [21]. However, verification involves $3 \log n$ pairings where n is the number of members in a group (one pairing requires 260000 gas $\approx$ 7 elliptic curve (EC) point multiplications [19, 49]). Schemes such as [22, 23] aim for logarithmic (in size of group) communication complexity and constant verifying algorithm. However, they also rely on pairings and the verifying algorithm requires a minimum of 81 pairings [23]. In [11], the proof is proposed for NFC ticketing systems so it outsources most of computation to verifiers and thus employs heavy pairing-based verifying algorithm. The proof adopted in Mesh is inspired by the literature on identity escrow [41], Schnorr and group signatures [24, 28, 52] and generalized in [47] has $O(n)$ communication and computation complexity, however, the computation relies only on EC multiplications and accordingly it is practical for smaller groups $n < 50$ as the current Ethereum block gas limit is around 8000000. To put things in perspective for a group of $n = 20$ members (See Sec. 5.2 for rationale), the proof is 1.3 KB in size and both proving and verifying timings are equal, and under 4 ms. Most importantly, $4n$ EC multiplications cost around 3200000 gas which is lower than performing the logarithmic or constant number of pairings in the above schemes. Additionally, with any set membership proof, one needs to include another proof for verifiable encryption to ensure correct revocation which is adds extra gas cost.

Mesh and IBM's Fabric. Last year IBM announced its Hyperledger Fabric and their blockchain-based solution for supply chain management [9, 35]. Fabric is a private permissioned blockchain where only certified members can transact, maintain and update the blockchain In what follows we compare Mesh and Fabric solutions.

- **Confidentiality.** Fabric enforces confidentiality through *channels* where only a group of participants create their separate local blockchain, keep their own ledger copies, and update it. There are several scalability issues within their model, as the number of such channels grows, the overall data structure is more likely to become unmanageable as it turns from one chain to an unstructured forest of chains. In other words, if we like to enforce transfer of ownership between s owners in a supply chain contract, we may need a minimum of *s* related channels on top of the actual blockchain. Also, since only few nodes maintain and update channels, the possibility of nodes collusion is higher than public blockchains, essentially because the state is agreed upon using consensus between only these nodes and is not publicly audited. On the other hand, in Mesh, everything is kept on one chain and we handle confidentiality through the use of lightweight symmetric key chaining, which is as scalable as the original blockchain.

Side database (SideDb) is Fabric's solution to scalability of channels. SideDb allows two or more entities within a channel to create a parallel outside private database to keep data in and only commit its hash to the channeled blockchain. However, unlike channels, such DBs can be purged while leaving its hash as a trace on channels. Such a solution deviates from the core immutability principal of blockchains even if a trace exists, what happens to the purged data is unknown.

- **Cost.** Private blockchains may not need to pay individual nodes but service fee is required for participation. Fabric's food supply chain solution (Food Trust) is offered in a subscription software as a service model (SaaS), where the monthly subscription costs between \$100 to \$10000, depending on the size of business, to trace the history of food products only [35]. In the event that an organization requires participation with standard support, a total monthly subscription between \$3800 and \$17700 is charged. On the other hand, an owner participation in Mesh follows a Pay per Use model with one data submission use ranges between (\$2 to \$4) depending on group size. Such a pricing model is known to be the most economically beneficial model for customers of cloud services and is adopted by Slack and Amazon Web Services [36]. Furthermore, in Mesh tracing is for free.

Generally, paying customers that go on private blockchains are controlled by the entities and developers owning it. Lack of public visibility, may enable the network to be manipulated by the developers who can enforce new rules to manipulate transactions. One has to wonder, why do we need to keep information away if there is a way to secure it on public blockchains.

Our contributions. In this work, we propose a framework that utilizes membership and verifiable encryption proofs by combining proofs of knowledge, with ElGamal encryption over elliptic curves and symmetric key encryption to provide accountability and privacy of transactions on public permissionless blockchains. We demonstrate the use of such a framework with Mesh which is a supply chain management solution. Mesh offers the following two security features:

– *Plausible Anonymity and Verifiable Identification.* By appending a ciphertext that encrypts the public identity of an authorized member, and a membership proof that the ciphertext indeed encrypts a valid member identity to the transaction, Mesh provides accountability and local anonymity of participating members with respect to their affiliated organization group. Thus, a smart contract managing a supply chain can verify the membership of a given authorized organization without learning the identity of the member. The addition of this extra two parameters to transactions turns the blockchain transaction signing scheme to a group signing scheme where members of a given organization share one blockchain credentials and each provides a publicly verifiable proof that the ciphertext encrypts their valid member identity. The associated ciphertext can only be decrypted by Mesh untrusted revocation manager according to a revocation clause policy. *Verifiable identification* is guaranteed because Mesh revocation manager supplies a proof that the revealed identity is indeed the encrypted one.

– *Forward Secrecy.* Using lightweight cryptographic permutation, and assuming the asymmetry in the backward and forward communication channels between an RFID tag and a reader, we propose a transfer of ownership and key chaining protocol that enables the tag to generate a new symmetric key for new owners to use in encrypting their blockchain updates. Moreover, the tag generates all the previous keys using the key chaining mechanism such that all previous updates can be verified by the current owner. Hence, the term *forward secrecy*, which is different than perfect forward secrecy [40], implies that all forward information updates are kept secret from previous owners.

We formally define and prove the above two security features, conduct experiments to measure the performance of the membership proof system proving and verifying procedures, and report the results relevant to the organization group size. We discuss the proof overhead in terms its computation and communication complexities, and present a comparison with existing generic proof systems.

2 Preliminaries

We write $x_{i=1}^{n}$ to denote the set $\{x_1,x_2,\cdots,x_n\}$. We denote the process of sampling y uniformly at random from $\mathbb{Z}_q$ by $y \in_R \mathbb{Z}_q$ or $y \xleftarrow{R} \mathbb{Z}_q$. Throughout the paper we let λ denote a security parameter, q be a large prime, and we define an elliptic curve $\mathcal{EC}$ over $\mathbb{F}_q$ by the set of solutions (x, y) satisfying $y^2 = x^3 + ux + v$ where all x, y, u, and $v \in \mathbb{F}_q$ and $4u^3 + 27v^2 \neq 0$. If not explicitly notated, we assume all operations are $\bmod q$. We let α and β denote primitive points in $\mathcal{EC}$. It is believed that the EC discrete logarithm problem is hard in EC groups. In other words, given a curve point $Q \in \mathcal{EC}$ and a secret value $e \in \mathbb{Z}_q$, a new curve point $P = eQ$ can be easily evaluated by adding Q to itself e times, however, given P and Q it is hard to find the secret e. Henceforth, we denote e by the EC discrete logarithm of point P to the base point Q, or simply the discrete log of P.

Zero Knowledge Proofs

A Zero knowledge (zk) proof is a protocol by which a prover P proves to a verifier V the truthfulness of some statement without conveying why [31]. For an NP language L, statements are of the form $u \in L$. w denotes a witness for a statement u if $(u, w) \in R$ where R is a polynomial time decidable binary relation associated with L. A zk proof has the following three properties; (a) *completeness*: a prover with a witness w for $u \in L$ can always convince the verifier, (b) *soundness*: except with negligible probability, a prover cannot convince a verifier when $u \notin L$, (c) zero-knowledge: The interaction should not reveal anything to the verifier about w [27]. Another notion is witness indistinguishability [27] which states that V cannot distinguish between two executions of the protocol which differ in the specific witness P is using. zk proofs are interactive protocols that require multiple rounds of interaction between a prover and receiver. However, non-interactive zero knowledge proofs (NIZKs) are possible where a common reference string (CRS) is shared by the prover and verifier and generated in a setup phase.

Definition 1 (Zero Knowledge Proof)**.** *A triple of procedures (setup, $\mathcal{P}$, $\mathcal{V}$) form a zk proof that $(u, w) \in R$ such that:*

- *$(crs) \leftarrow setup(1^\lambda)$ generates the CRS public parameters.*
- *$\pi \leftarrow \mathcal{P}(crs, w, u)$ run by the prover on a witness w to produce a proof π such that $(u, w) \in R$.*
- *$(\top/\bot) \leftarrow \mathcal{V}(crs, u, \pi)$. Verifies that π proves the knowledge of some witness for u such that $u \in L$.*

3 Mesh **Abstract Model**

Mesh system consists of three subsystems, Mesh server ($\mathcal{M}$ *server*), a blockchain, and RFID subsystems. An architectural schematic of Mesh showing the interaction of its entities is given in Figure 1. The green components are trusted to behave honestly, and Mesh handler is only responsible for passing messages between the blockchain and the server managers. We assume that Mesh server is hosted by some service provider, however, because its authorized list of accounts and main execution are published and verified on a public blockchain, we can assume a minimum trust level.

In its supply chain life cycle, we assume that a product is identified by its RFID tag and can be owned by l owners (a given supply chain participants) where each owner is a member of an organization group (shortly group) of some n facilities or members $(S^1, S^2, \cdots, S^n)$.

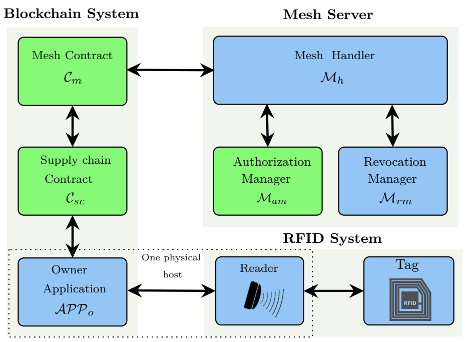
Blockchain System
Mesh Server
Mesh Contract $\mathcal{C}_m$
Mesh Handler $\mathcal{M}_h$
Supply chain Contract $\mathcal{C}_{sc}$
Authorization Manager $\mathcal{M}_{am}$
Revocation Manager $\mathcal{M}_{rm}$
RFID System
One physical host
Owner Application $\mathcal{APP}_o$
Reader
Tag
RFID

Fig. 1. Mesh system abstract architecture

3.1 Blockchain System

The blockchain side of a given instance of Mesh consists of two smart contracts and $l + 1$ wallet accounts that interact with these contracts.

$\mathcal{C}_{sc}$ contract. The main functionality of managing the supply chain activities is implemented in the supply chain smart contract $\mathcal{C}_{sc}$. It stores verified tracking and product information. It also can implement contractual clauses between involved stakeholders. For example, $\mathcal{C}_{sc}$ can handle service payments and enforce penalties if provisions are not met. We bypass these functionalities, and concentrate on the special privacy requirements that we want to fulfill. In Mesh, $\mathcal{C}_{sc}$ verifies the authorization of participating owners and stores authenticated private tracking history of the product. Most importantly it verifies proofs that the identity of a transaction signer within the organization is valid and can be revoked by $\mathcal{M}$ *server* when needed.

$\mathcal{C}_m$ **contract.** C_m is $\mathcal{M}$ *server's* onchain interface for providing services to $\mathcal{C}_{sc}$. C_m is essential because smart contracts lack Internet access, they can only communicate with onchain entities, which can be other contracts or wallet accounts. C_m stores a list of all authorized organizations public accounts and their members public *id*s. It is responsible for handling authorization and anonymity revocation requests by $\mathcal{C}_{sc}$. In such a setting, $\mathcal{C}_{sc}$ can be seen as a requester and $\mathcal{C}_m$ is the handler.

Wallet accounts. Each owner group maintains a blockchain wallet account $\mathcal{W}_i$ that is associated with a certified public and private key pair $(pk_i,\ sk_i)$, for $1 \leq i \leq l$. In other words, only organizations that are registered and their wallet credentials are authorized by $\mathcal{M}$ *server* are allowed to invoke functions in $\mathcal{C}_{sc}$. We assume these accounts are controlled by a blockchain interface $\mathcal{APP}_o$ that is available at all the members of every group. The remaining wallet account $\mathcal{W}_m$ is administered by $\mathcal{M}$ *server*. $\mathcal{W}_m$ is associated with $\mathcal{M}$ *server's* key pair $(pk_m,\ sk_m)$ and it is the only account that can update the accounts list in $\mathcal{C}_m$.

3.2 RFID System

The offchain side of Mesh includes the product which is identified by its RFID tag, and RFID readers that are operated by the members of organization groups.

Tag-Reader model. We assume the "*minimalist model*" for RFID tags [38] where the reader broadcasts with a signal stronger than that from tag-to-reader. Moreover, we assume that the tag shares an initial authentication key AK_0 with its first owner group member that is securely updated during a transfer of ownership session (see Sec. 6.2). The tag is assumed to maintain an internal counter that is incremented with each transfer of ownership and to support a `Setid(.)` and `Getid()` commands that enable the first owner to write the account of $\mathcal{C}_{sc}$ and remaining owners to read it, respectively. Finally, we assume that the tag has at least 2000 Gate Equivalents (GE) available for security functionalities.

3.3 $\mathcal{M}$ *server*

The part of Mesh which authorizes parties that can own products is denoted by $\mathcal{M}$ *server*. It verifies, updates and keeps a list of all manufactures, wholesale, third party logistic companies, and retailers that are allowed to use the system. In this sense, $\mathcal{M}$ *server* has a registration interface $\mathcal{M}_{am}$ that accepts memberships requests from organizations and updates $\mathcal{C}_m$ lists accordingly. The other part $\mathcal{M}$ *server* is $\mathcal{M}_{rm}$ which is responsible for handling revocation requests made by a requesting contract $\mathcal{C}_{sc}$ to $\mathcal{C}_m$. $\mathcal{M}$ *server* also implements a handling module $\mathcal{M}_h$ to monitor the state of $\mathcal{C}_m$, so that whenever a revocation request is pending, $\mathcal{M}_h$ passes it $\mathcal{M}_{rm}$, and then relays the response using $\mathcal{W}_m$ back to $\mathcal{C}_m$.

3.4 From Supply chains to Mesh

A supply chain is a collection of suppliers that partially contribute in the creation, warehousing, routing, and finally delivering a specific product to end customers. The chain is made up of links between such suppliers to enable consumers to have a chronologically ordered history of what they are purchasing. Each link carries information regarding time, cost, storage conditions, etc. All these links are stored in supply chain management systems which are centrally controlled repositories that furnish such history to consumers. All suppliers are authorized by a central entity that controls a given supply chain management system. Examples of retail supply chain management systems include coca cola, Amazon, etc. Additionally, such systems host a supply chain for each line of products, so for example Amazon's system has hundreds of supply chains for different products.

The concept of the chain is important, because it means than the next link cannot be reached without going through the previous one which perfectly resonates with blockchains. Accordingly, mapping supply chain entities to our system means that all suppliers are "owners" affiliated with organizational groups in Mesh where their wallet accounts get authorized by Mesh server, $\mathcal{M}_{am}$. The supply chain smart contract $\mathcal{C}_{sc}$ which lives on the blockchain serves as the supply chain management system that verifies owners authorizations and stores the chain of information.

3.5 Threat Model

In what follows, we give our assumptions regarding the security of the system's components and sources of threat.

- ***Contracts and blockchain communication.*** contracts contain publicly visible code, thus, they are trusted to execute correctly. Transactions are authenticated because they are identified by their source accounts. All communication is sent in the clear and thus lack confidentiality.

- ***RFID tags and readers.*** We assume that unauthorized tags (products) may be injected in the supply chain and unauthorized readers may be used to scan genuine tags. Following the *minimalist* security model [38], we capture the asymmetry between the forward and backward communication channels between tags and readers and thus, we get different eavesdropping ranges. In other words, eavesdropping on the reader to tag communications can be done within hundreds of meters but the other way is not possible unless the eavesdropping reader is within the vicinity of the tag (e.g., ten meters).

- ***Owners and group wallets.*** A wallet account operated by a given group member within the same organization maybe used to impersonate other members by submitting valid messages to the blockchain using the group credentials. Also, group members maybe curious to reveal the identity of a transaction sender and they may collude to get such a knowledge. Owners may try to corrupt the protocol by submitting updates encrypted with an unidentified key or revoking the anonymity of other owners before products arrive to their final destination.

- $\mathcal{M}_{am}$ ***and*** $\mathcal{M}_{rm}$***.*** We assume that $\mathcal{M}_{am}$ is trusted by all the organizations participating in Mesh to authorize their wallet accounts and members' identities, and publish them in $\mathcal{C}_m$. On the other hand, $\mathcal{M}_{rm}$ is not trusted to release the correct identity of a group member when it receives a revocation request.

3.6 Privacy goals

Since we are using a public blockchain, transaction source addresses and contents are seen by everyone whether involved in an ongoing supply chain contract or not. Accordingly, in what follows, we define the the goals of Mesh in terms of the offered anonymity and confidentiality.

Local Anonymity. Our system involves physical interactions between participating entities in the real world, hence the definition of anonymity is equivalent to that provided by the onion routing protocol. More precisely, a given entity is anonymous if its transactions can not be identified from those originating from its anonymity set which is the set of members of its organizational group. However, similar to onion routing, the anonymity of a given entity is guaranteed with respect to all observers of the blockchain except 1) the two entities that precede and follow it in the supply chain as such anonymity is revoked during physical pickup and drop-off, and 2) $\mathcal{M}_{rm}$ because, it is the revocation manager in Mesh.

Confidentiality. In a typical supply chain scenario, one requires the history of a given product to be viewed only by its current owner. In Mesh, we enforce confidentiality of updates as a form of transfer of ownership by generating a unique key for each new owner. Such a key is used to encrypt the committed updates so that all previous owners, as well as other observers of the blockchain cannot learn the current status of the product. Additionally, all previous keys are generated for the current owner so that history can be verified. By history we mean status, storage conditions, quantities, drop-off and pickup dates, and timings, thus no previous owners identification information is encrypted as not to violate the anonymity requirement. Typically, such identification and other public information can be available for consumers after products are at retailers so that they may verify products' chain of custody.

3.7 Limitations.

As with other physical systems, we have the following limitations arising from physical adversarial actions.

- **Blockchain vs. real world**. The blockchain provides an authenticated unaltered chain of custody. Thus, an end user is able to verify exactly by whom the product they intend to purchase has been assembled and made. However, at the end, we are still relying on such owners to be true in what they report and that physical product remains unaltered in transit. For that, blockchained supply chains has to rely on authenticated auditing processes to appropriately enable more trust in the coherence between the transacted records and the physical products [10].

- **RFID tagging**. We rely on RFID tags to perform lightweight computations that enable the confidentiality feature of Mesh. However, a long open problem is that physical attacks on RFID such as cloning and physical swapping may enable counterfeit products to be integrated in the supply chain. Solutions to cloning include lightweight physically un-clonable functions (PUFs) and crypto. Since, we assume cryptographic keys are shared between owners and RFID tags, then a third party cloning is not a threat. However, swapping attacks originate from RFID authorized owners and for that, we have to rely on authenticated auditing as discussed above in order to regulate the practices of owners.

- **Privacy enforcement**. We assume that owners are going to use the keys that are generated by RFID tags to encrypt updates. However, in the event that an adversarial owner do not use an RFID generated key, it can be identified by the following owner in the chain which is not able to verify the history of the product. Then using a multi-signature threshold scheme and a smart contract penalty protocol, its anonymity can be revoked and penalized by the smart contract. Assuming that t owners in a supply chain are honest, such an adversary can be always caught. At the end, similar to real life, such owners are primarily interested in maximizing their profit, so trying to corrupt a contract that they are going to be paid by seems counterintuitive.

4 Mesh Protocol: Blockchain and Server sides

In this section we first describe a high level overview of the blockchain side of Mesh through the life cycle of its protocol messages.

4.1 Message Flow

Figure 2 shows the flow of messages in the authorization or revocation protocols (depicted in red). An authorization request by O_i is in the form of a message $\mathtt{m}_1 = (\mathtt{acclist},\mathtt{c},\pi_{id})$. This message is created by $\mathcal{APP}_o$ and sent to $\mathcal{C}_{sc}$. acclist contains a list of owner accounts sorted by the order in which the products is going to be owned. The authorization request message $\mathtt{m}_1$ also carries $(\mathtt{c},\pi_{id})$, where c is the sender's encrypted identity and π_{id} is the proof defined in Definition 5. Following $\mathtt{m}_1$, $\mathcal{C}_{sc}$ appends $\mathcal{W}_1$ to acclist, generates $\mathtt{m}_2 = (\mathtt{acclist})$ and forwards it to $\mathcal{C}_m$ which in turns checks authorization, responds to $\mathcal{C}_{sc}$ with either $\top$ and a list of IDs of $\mathcal{W}_1$ group members or $\perp$. $\mathcal{C}_m$ also generates a unique identifier id (e.g., a counter) and stores it along with $\mathcal{C}_{sc}$ account to indicate contracts that are in service. Since $\mathcal{C}_m$ handles multiple requests from different instances of $\mathcal{C}_{sc}$ and responses from $\mathcal{M}_{rm}$, then id is communicated to all concerned entities and enables $\mathcal{C}_m$ to match requests with their corresponding responses.

Optionally, a given owner O_i may initiate the identity revocation protocol (shown as the red data flow in Figure 2). It starts by sending revocation request message $\mathtt{m}_1 = (\mathtt{acc},\mathtt{i},\mathtt{c},\pi_{id})$ to $\mathcal{C}_{sc}$ containing the organizational account $\mathtt{acc} = \mathcal{W}_{O_j}$ and a contract identifier to an encrypted identity i that was used in a previous transaction by one the organizational members. $\mathtt{m}_2 = (\mathtt{acc})$ and $\mathtt{m}_3 = (\mathtt{id},\mathtt{acc})$ are generated by the $\mathcal{C}_{sc}$ and $\mathcal{C}_m$, respectively to forward the revocation request to $\mathcal{M}_{rm}$ which in turn responds with $\mathtt{m}_4 = (\mathtt{id},\ \mathcal{ID}_i,\ \pi_r)$ indicating that $\mathcal{ID}_i$ is the identity encrypted in c and providing a proof π_r for the correct identification.

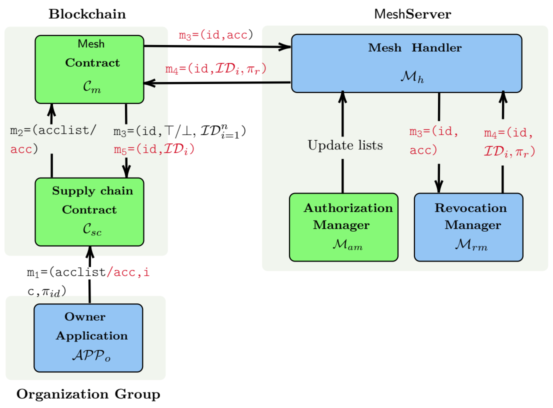
Blockchain
MeshServer
Mesh Contract $\mathcal{C}_m$
$\mathtt{m}_3$=(id,acc)
$\mathtt{m}_4$=(id,$\mathcal{ID}_i,\pi_r$)
Mesh Handler $\mathcal{M}_h$
$\mathtt{m}_2$=(acclist/acc)
$\mathtt{m}_3$=(id,$\top/\perp$, $\mathcal{ID}^n_{i=1}$)
$\mathtt{m}_5$=(id,$\mathcal{ID}_i$)
Update lists
$\mathtt{m}_3$=(id, acc)
$\mathtt{m}_4$=(id, $\mathcal{ID}_i,\pi_r$)
Supply chain Contract $\mathcal{C}_{sc}$
Authorization Manager $\mathcal{M}_{am}$
Revocation Manager $\mathcal{M}_{rm}$
$\mathtt{m}_1$=(acclist/acc,i c,π_{id})
Owner Application $\mathcal{APP}_o$
Organization Group

Fig. 2. Authorization/Revocation Message flow in Mesh protocol.

4.2 Protocol Specification

We adopt the same notational conventions for describing offchain programs and onchain contracts as described in [43]. We denote the process of transaction generation by *Authsend(.)*. We also use *Assert(cond)* to denote the process of verifying a condition, if not verified, then the function is aborted. In what follows, we give the specification of Mesh's onchain components, the offchain components which control wallet accounts are detailed in the extended version of our paper.Throughout the paper transaction and message are used interchangeably.

4.2.1 Supply chain contract $\mathcal{C}_{sc}$.

This is the onchain entity that manages the supply chain information of a product.In this work, we focus on the core functionality involving the privacy of transactions and confidential updates, so we skip details related to different revocation scenarios as well as fund transfers including payments and transaction fees. Figure 3 depicts a pseudo code of the $\mathcal{C}_{sc}$. The contract is first created by an authorized group member where authorizations is checked by issuing a *AuthorizeReq* to Mesh handling contract $\mathcal{C}_M$. The function Create accepts an ordered *acclist* from the first product owner which sets the authorized supply chain product owners. The function $\mathtt{VerifyId}(c,\pi_{id},\mathcal{ID}^n_{i=1})$ verifies the proof that is sent with owner generated transactions by running the identity verification algorithm defined in Definition 5. $\mathcal{C}_{sc}$ processes update requests in the function Update which maintains a log of private information submitted by owners wallet accounts. Along with each new up-

```
Supply Chain Contract $\mathcal{C}_{sc}$

Initialize: state:= intit, scList:= {}, $id$, $counter$ := 0. let $y_m$, $q$, $\alpha$,
        and $\beta$ denote hard coded CRS public parameters generated
        by Mesh.

Create: On receiving ($acclist$,$c$,$\pi_{id}$) from some $\mathcal{W}_O$
    \\ append $\mathcal{W}_O$ to acclist, send request and wait for response
    Send AuthorizeReq($\mathcal{W}_O \| acclist$) to $\mathcal{C}_m$
      Assert receiving ($id$, $\mathcal{ID}^n_{i=1}$) from $\mathcal{C}_m$
        Assert(VerifyId($c$,$\pi_{id}$,$\mathcal{ID}^n_{i=1}$))
         sclist = $\{\mathcal{W}_O \| acclist\}$
         $counter = counter + 1$
         sclist[$\mathcal{W}_O$].active = true
    \\ set contract identifier for each $c$ under $\mathcal{W}_O$
         sclist[$\mathcal{W}_O$].sidentity[$counter$]=$c$
         state:= created

Update: On receiving ($Enc_k(info)$,$c$,$\pi_{id}$) from some $\mathcal{W}_O$
    Assert state $\neq$ init
    Assert $\mathcal{W}_O \in$ sclist
    If(!sclist[$\mathcal{W}_O$]. active)
      Send MemberReq($\mathcal{W}_O$) to $\mathcal{C}_m$
      Assert receiving ($\mathcal{ID}^n_{i=1}$) from $\mathcal{C}_m$
    Assert(VerifyId($c$,$\pi_{id}$,$\mathcal{ID}^n_{i=1}$))
      sclist[$\mathcal{W}_O$].info.append= $Enc_k(info)$
      $counter = counter + 1$
      sclist[$\mathcal{W}_O$].sidentity[$counter$]= $c$.
      If $\mathcal{W}_O$ is last account in $acclist$
        tag public info
        state:= reached
      Else state:= updated

Revoke: On receiving ($\mathcal{W}_{O_i}$, $i$, $c$,$\pi_{id}$) from some $\mathcal{W}_O$
    \\ revokation after final destination only
    Assert state := reached
    Assert $\{\mathcal{W}_O, \mathcal{W}_{O_i}\} \in$ sclist
    Send MemeberReq($\mathcal{W}_O$) to $\mathcal{C}_m$
    Assert receiving ($\mathcal{ID}^n_{i=1}$) from $\mathcal{C}_m$
    Assert (VerifyId($c$,$\pi_{id}$,$\mathcal{ID}^n_{i=1}$))
      Send RevokeReq($id$,$\mathcal{W}_{O_i}$,sclist[$\mathcal{W}_O$].sidentity[$i$]) to $\mathcal{C}_m$
      on receiving($\mathcal{ID}$) from $\mathcal{C}_M$
        store sclist[$\mathcal{W}_O$].pidentity[$i$]=$\mathcal{ID}$ as a response

VerifyId($c$,$\pi_{id}$,$\mathcal{ID}^n_{i=1}$)
    \\ run verification algorithm of membership proof in Def. 5
  $v = (\pi_{id} \overset{?}{=} \mathsf{PoK}(x, k)$, given $c$, $y_m$, $q$, $\alpha$, $\beta$,$\{\mathcal{ID}^n_{i=1}\})$
Output $v$
```

Fig. 3. Mesh's supply chain contract.

date, a ciphertext encrypting the identity of the group member submitting it is added, so that when a valid revocation request is issued, members of an organizational group $\mathcal{W}_O$ that submitted updates can be retrieved. The updated information is encrypted with a secret key to protect its confidentiality. However, when the product reaches its final destination, some info are tagged as public to enable customer at retailers to check tracking history of a given product.

Anonymity Revocation. The function Revoke can be executed only when the contract state is *reached*. Otherwise, a corrupted owner can request the revocation of the anonymity of previous owner which violates the privacy of the products actively involved in the supply chain. On the other hand, anonymity revocation is a way to resolve disputes which can possibly happen in the middle of the contract. In such a case, where a dispute severely interrupts the supply chain cycle such as an owner holding off products or counterfeit products are discovered by auditing authorities, a T out of N with N being the owners involved in the contract, multisignature threshold scheme can be used to request anonymity revocation which in this case takes precedence over the original revocation policy. Smart contracts are very efficient in implementing such clauses using multisignature threshold schemes [56]. Particularly, using currency escrow mechanisms and majority agreement, an owner who initiates a revocation protocol before the end of the contract can be significantly penalized financially by the contract if the owners who enabled it through the multisignature threshold scheme found that the act is malicious. Disincentivized by financial loss, owners would be only inclined to initiate such procedure if it is absolutely necessary which works best towards protecting the privacy of the whole contract.

4.2.2 Mesh **handling contract $\mathcal{C}_M$.**

$\mathcal{M}$-*server*'s onchain functionality is implemented in $\mathcal{C}_m$ which primarily maintains lists of authorized organizations *orglist* along with the public $\mathcal{ID}$'s of their members. It only accepts Add/Delete requests from $\mathcal{M}$-*server* wallet account $\mathcal{W}_m$. $\mathcal{C}_m$ responds to authorization requests from a given $\mathcal{C}_{sc}$. It also stores identity revocation requests and waits for responses from $\mathcal{M}_h$ ($\mathcal{W}_m$) to forward to $\mathcal{C}_{sc}$. All revocation responses are accompanied with a proof π_r for verifiable decryption. The function VerifyRev(c,π_r,$\mathcal{ID}$) verifies the proof π_r which proves that the decrypted $\mathcal{ID}$ is indeed the one that was encrypted in c See Def. 6. Detailed contract is provided in the extended version of the paper.

5 Locally Private Transactions

We denote a transaction by *locally private*, if its local signing member identity within a known organization group is anonymous. Due to the accountability requirement, we want to make sure that $\mathcal{M}_{rm}$ is able to revoke such an anonymity when needed, and this fact is verifiable by the supply chain contract $\mathcal{C}_{sc}$ when it receives a transaction. For that we adopt the notion of identity escrow [41, 47] to build a proof which guarantees that the identity of a transaction signer is encrypted under $\mathcal{M}$ *server* public key y_m.

In what follows, we define the proofs of knowledge of representation [46], equality [29], and existential equality [47] in the EC discrete logarithms setting. Let $\mathsf{PoK}\{(x) : conditions\}$ denote the proof of knowledge of the witness x satisfying *conditions*.

Definition 2 (Proof of representation). *A proof of knowledge of the EC discrete logarithms representation of y to the base points $\alpha_{i=1}^{j}$, where $1 \leq j \leq (q-2)$ is given by $\mathsf{PoK}\{(x_{i=1}^{j}) : y = \sum_{i=1}^{j} x_i\alpha_i\}$. The proof is the tuple $(c, r_{i=1}^{j})$ and is generated non-interactively by the prover as follows:*

- *Commitment: For each secret EC discrete logarithm x_i, a commitment t_i is evaluated by $v_i \xleftarrow{R} Z_q$, $t_i = v_i\alpha_i$.*
- *Challenge: Using a collision resistant hash-function $\mathcal{H}$, the challenge $c = \mathcal{H}(\alpha_{i=1}^{j}, y, \sum_{i=1}^{j} t_i)$ is computed.*
- *Response: For each secret x_i, a response is given by $r_i = v_i + cx_i$.*

Definition 3 (Proof of equality). *A proof of knowledge that the EC discrete logarithms of y_1 and y_2 to the base points α_1 and α_2, respectively, are equal is given by $\mathsf{PoK}\{(x) : y_1 = x\alpha_1 \wedge y_2 = x\alpha_2\}$. The proof is the tuple (c, r) and is generated non-interactively as follows:*

- *Commitment: The prover chooses $v \xleftarrow{R} Z_q$ and computes $t_1 = v\alpha_1$, and $t_2 = v\alpha_2$.*
- *Challenge: Using a collision resistant hash-function $\mathcal{H}$, the challenge $c = \mathcal{H}(\alpha_1, \alpha_2, y_1, y_2, t_1, t_2)$ is computed.*
- *Response: $r = v + cx$.*

Proving the knowledge of either the EC discrete logarithm of y_1 *or* y_2 is more difficult because the verifier does not know which one is the right solution. In other words, the prover wants to prove that she knows either x_1 or x_2 where $y_1 = x_1\alpha$ and $y_2 = x_2\alpha$. A proof for partial knowledge for solving this problem is proposed in [25]. Combining the proof of partial knowledge with proof of equality, one can generate a proof for existential equality as has been shown in [47]. Moreover, a systematized methodology for developing proof systems by combining different proofs about discreet logarithms has been proposed by Camenisch (cf. Sec. 4 in [20]) which is also used to generate the proof system in Definition 4. More precisely, one can generate the commitment, challenge and response for a proof of knowledge of x such that $y = x\alpha \wedge (\exists|_{1\leq i\leq n}: y_i = x\beta)$ by evaluating the modified Cartesian product of the knowledge sets of both proofs of equality and partial knowledge [20]. To simplify the notation of Definition 4, we assume that the prover knows x such that $y = x\alpha \wedge y_1 = x\beta$.

Definition 4 (Proof of existential equality). *A proof of knowledge that the EC discrete logarithm of point y to the base point α is equal to at leaset one of the discrete logarithms of points $y_{i=1}^{n}$ to the base point β is given by $\mathsf{PoK}\{(x) : y = x\alpha \wedge (\exists|_{1\leq i\leq n}: y_i = x\beta)\}$. The proof is the tuple $(c_{i=1}^{n}, r_{i=1}^{n})$ and is generated non-interactively as follows:*

- *Commitment: The prover chooses $v_{i=1}^{n} \xleftarrow{R} Z_q$ and $c_{i=2}^{n} \xleftarrow{R} Z_q$, and computes:*
  - $s_1 = v_1\alpha$ and $s_i = v_i\alpha - c_i y$
  - $t_1 = v_1\beta$ and $t_i = v_i\beta - c_i y_i$ for $2 \leq i \leq n$.
- *Challenge: Using a collision resistant hash-function $\mathcal{H}$, the challenge*
  - $c = \mathcal{H}(\alpha, \beta, y, y_{i=1}^{n}, t_{i=1}^{n}, s_{i=1}^{n})$ is computed.
  - $c_1 = c - \sum_{i=2}^{n} c_i$.
- *Response: The prover calculates:*
  - $r_1 = v_1 + c_1 x$, and $r_i = v_i$ for $2 \leq i \leq n$

The non-interactive proof of existential equality has been shown to be sound, complete and witness indistinguishable proof of knowledge in [47]. For completeness, we give the security proofs in Appendix B.

5.1 Transaction Signing Scheme

In what follows, we define the scheme ($\mathcal{K}$, $\mathcal{S}$, $\mathcal{P}$, $\mathcal{V}$, $\mathcal{R}$) where public key encryption is performed using ElGamal scheme over EC [42]. We assume, the chosen curve of prime order q parameters and the primitive points α and β are known by all entities as well as $\mathcal{M}$ *server* public encryption key y_m where $y_m = x_m\beta$. We assume that an owner wallet key credentials $\mathcal{W}_O : (pk_o, sk_o)$ are generated and registered at $\mathcal{M}_{am}$, and securely shared with all the n members of the group.

1. **Key setup** $\mathcal{K}$. This key generation process is run by each member to generate their public identity $\mathcal{ID}_i$ that is registered at $\mathcal{M}_{am}$ and stored in $\mathcal{C}_m$.

   $$(x_i, \mathcal{ID}_i) \leftarrow \mathcal{K}(1^\lambda)$$
   $$x_i \xleftarrow{R} Z_q, \mathcal{ID}_i = x_i\alpha$$

2. **Tx signature generation** $\mathcal{S}$. The identity is first encrypted using ElGamal scheme under y_m, then utilizing the membership proof system defined in Definition 5, a proof π_{id} is constructed.

   $$(\sigma, c, \pi_{id}) \leftarrow \mathcal{S}(m, sk_o, x_i, y_m)$$
   $$k \xleftarrow{R} Z_q, c = (c_1, c_2) = (k\beta, ky_m + x_i\alpha)$$
   $$\pi_{id} = \mathsf{PoK}\{(x_i) : c_2 = ky_m + x_i\alpha \wedge (\exists|_{1\leq i\leq n}: \mathcal{ID}_i = x_i\alpha)\}$$

   Now a member forms a blockchain transaction signature by first hashing the message and the ciphertext then signing them by the group's secret key, $\sigma =$

$\Sigma_{sk_O}(\mathcal{H}(m,c))$ where $\Sigma_k(.)$ denotes the blockchain transaction signing procedure using key k.

3. **Signature verification** $\mathcal{V}$. The transaction signature on m and c is first verified using the group's public key pk_o. Then the proof π_{id} is verified against c and the set of member identities $\mathcal{ID}_{i=1}^n$.

$$(\top/\bot) \leftarrow \mathcal{V}(\sigma, m, c, \pi_{id}, pk_O, \mathcal{ID}_{i=1}^n)$$

4. **Identity revocation**. $\mathcal{M}_{rm}$ decrypts c using x_m and recovers the encrypted identity $\mathcal{ID}$. Moreover, it generates a proof that the encrypted identity in c is equal to the revealed $\mathcal{ID}$.

$$(\mathcal{ID}, \pi_r) \leftarrow \mathcal{R}(c, x_m)$$
$$\mathcal{ID} = c_2 - x_m c_1$$
$$\pi_r = \mathsf{PoK}\{(x_m) : c_2 - \mathcal{ID} = x_m c_1\}$$

5.2 Membership Proof System.

A signing member need to generate a proof for the following two statements.
- c encrypts a valid identity form (i.e., $\mathcal{ID} = x\alpha$).
- The encrypted identity is a member of the set, i.e., $\mathcal{ID} \in \{\mathcal{ID}_1, \mathcal{ID}_2, \cdots, \mathcal{ID}_n\}$.

Common Reference String (CRS). Similar to all non-interactive proofs, some public parameters have to be known to both sender and verifier prior to engaging in a proof protocol. Some CRSs are highly structured, and can possibly foster a backdoor, thus requiring a cumbersome trusted setup to choose them such as that used in zkSNARKs [15]. Other CRSs are simply chosen uniformly at random and require not trusted setup such as ours and the ones used in [18, 37]. Our CRS primarily contains the curve parameters, two primitive points α and β which are hard-coded in the contracts and generated by Mesh-server, Mesh's public key y_m, and the public IDs of members of a given group, which are chosen by each member privately and published by $\mathcal{M}_{am}$ in $\mathcal{C}_m$. The randomness of α and β can be guaranteed by requiring $\mathcal{H}_M$ to generate them by hashing a small public seed, such a method has been used in [17]. Unlike zkSNARKs where the challenge is hard-coded, in Mesh, all challenges are freshly and uniquely generated by the prover using commitments and the Fiat-Shamir heuristic.

Formally, the identity proof system is defined as follows:

Definition 5 (Membership Proof π_{id}). *Given c=(c_1, c_2), where $c_1 = k\beta$, $c_2 = ky_m + \mathcal{ID}$, and $\mathcal{ID} = x\alpha$, a proof that c encrypts a valid member identity from a set of identities $\{\mathcal{ID}_1, \mathcal{ID}_2, \cdots, \mathcal{ID}_n\}$ is given by* $\mathsf{PoK}\{(x,k) : c_2 = ky_m + x\alpha \wedge c_1 = k\beta \wedge (\exists|_{1\le i\le n}: c_2 - \mathcal{ID}_i = ky_m)\}$ *and can be realized using*

$$\mathsf{PoK}\{(x,k) : c_2 = ky_m + x\alpha\} \wedge$$
$$\mathsf{PoK}\{(k) : c_1 = k\beta \wedge (\exists|_{1\le i\le n}: c_2 - \mathcal{ID}_i = ky_m)\}$$

In Figure 4, we give the details of the membership proof generation and verifications algorithms. For simplicity and without loss of generality, we assume the prover is a group member with public identity $\mathcal{ID}_1$.

Commitment. The prover with $\mathcal{ID}_1$ chooses $z_1, z_2, v_{i=1}^n, w_{i=2}^n \xleftarrow{R} Z_q$, and computes:
$u_1 = z_1 y_m$, $u_2 = z_2\alpha$
$t_1 = v_1\beta$ and $t_i = v_i\beta - w_i c_1$, for $i \in [2,n]$
$s_1 = v_1 y_m$ and $s_i = v_i y_m - w_i(c_2 - \mathcal{ID}_i)$, for $i \in [2,n]$

Challenge. Using a collision resistant hash-function $\mathcal{H}$ and message m, the following challenges are computed:
$ch_r = \mathcal{H}(m, y_m, \alpha, c_2, u_1 + u_2)$.
$ch_e = \mathcal{H}(m, \beta, y_m, c_1, c_2 - \mathcal{ID}_{i=1}^n, t_{i=1}^n, s_{i=1}^n)$.

Response. The prover first sets $ch_i = w_i$ for $i \in [2,n]$, then evaluate the following:
$r_1 = z_1 + kch_r$, $r_2 = z_2 + xch_r$.
$ch_1 = ch_e - \Sigma_{i=2}^n ch_i$.
$r_3 = v_1 + kch_1$, and $r_{i+2} = v_i$ for $i \in [2,n]$
The prover sends $\pi_{id} = (ch_r, ch_{i=1}^n, r_{i=1}^{n+2})$ to the verifier.

Verification The verifier checks the following:
$\mathcal{H}(m, y_m, \alpha, c_2, r_1 y_m + r_2\alpha - ch_r c_2) \overset{?}{=} ch_r$
$\mathcal{H}(m, \beta, y_m, c_1, c_2 - \mathcal{ID}_{i=1}^n, t_{i=1}^{'n}, s_{i=1}^{'n}) \overset{?}{=} \Sigma_{i=1}^n ch_i$, where
$t_i^{'} = r_{i+2}\beta - ch_i c_1$, $s_i^{'} = r_{i+2} y_m - ch_i(c_2 - \mathcal{ID}_i)$

Fig. 4. Membership proof π_{id} generation and verification procedures.

Definition 6 (Proof of revocation π_r). *Given $c = (c_1, c_2)$, where $c_1 = k\beta$, $c_2 = ky_m + \mathcal{ID}$, a proof that the identity $\mathcal{ID}'$ which is revealed by $\mathcal{M}_{rm}$ is indeed the one encrypted in c under y_m (i.e., $\mathcal{ID}' = \mathcal{ID}$) is given by* $\mathsf{PoK}\{(x_m) : c_2 - \mathcal{ID}' = x_m c_1 \wedge y_m = x_m\beta\}$.

π_r generation and verifications algorithms are given in Figure 5.

Commitment. the prover chooses $v \xleftarrow{R} Z_q$, and computes $t_1 = vc_1$, $t_2 = v\beta$.

Challenge. Using a collision resistant hash-function $\mathcal{H}$, compute:
$ch = \mathcal{H}(m, c_2 - \mathcal{ID}', y_m, c_1, \beta, t_1, t_2)$.

Response. the prover evaluates $r = v + x_m ch$, and sends $\pi_r = (ch, r)$ to the verifier.

Verification The verifier checks the following:
$\mathcal{H}(m, c_2 - \mathcal{ID}', c_1, rc_1 - ch(c_2 - \mathcal{ID}'), r\beta - chy_m) \overset{?}{=} ch$

Fig. 5. π_r generation and verification procedures.

Security. In Appendix C, we provide security proofs for the completeness, soundness, and witness indistinguishablity of *membership proof* when utilizing the chosen ElGamal variant over EC [42].

The above proof ensures the privacy of the sender because its identity is encrypted using ElGamal encryption over EC which is semantically secure under the decisional Diffie-Hellman assumption [16]. Assuming that the private keys of all the $\mathcal{ID}$s are known only by their corresponding members, the *membership proof* ensures that the valid identity of the transaction signer is indeed encrypted in c under y_m, formally.

Theorem 1. *Given two primitive curve points β and α, let the ciphertext $c = (c_1, c_2)$ where $c_1 = k\beta$ and $c_2 = ky_m + \mathcal{ID}_i$ be ElGamal encryption over EC of the identity $\mathcal{ID}_i$ under the public key of $\mathcal{M}$-server y_m. Under the assumption of the intractability of the ECDL problem, membership proof guarantees that the transaction signer knows the corresponding secret x_i such that $\mathcal{ID}_i = x_i\alpha$ and $\mathcal{ID}_i$ is a valid group member identity.*

Proof. Membership proof is made of the conjunction of two proofs, namely, *proof of representation* and *proof of existential equality*. The *proof of representation* $\mathsf{PoK}\{(x, b) : c_2 = by_m + x\alpha\}$ guarantees that the i-th member of a group knows any pair of values $(x, b) \in Z_q$ where $c_2 = by_m + x\alpha$. On the other hand, the *proof of existential equality* $\mathsf{PoK}\{(d) : c_1 = d\beta \wedge (\exists|_{1\leq i\leq n}\colon (c_2 - \mathcal{ID}_i) = dy_m)\}$ ensures that the i-th group member knows a value d where $c_1 = d\beta$ and that there exists at least one valid $\mathcal{ID}$ point, out of the public n $\mathcal{ID}$s, that when subtracted from c_2 leads to the point dy. Therefore, it follows that $b = d$ and $x\alpha$ equals to one of the valid $\mathcal{ID}$ points. In other words, $x\alpha \in \{x_1\alpha, x_2\alpha, \cdots, x_n\alpha\}$. □

According to the soundness proof of the membership proof in Appendix C, no entity may construct a valid proof with non negligible probability without the knowledge of a secret identity, x_i corresponding to one of the public identities $\mathcal{ID}$s. Accordingly, impersonation of a given member by another group member or $\mathcal{M}_{rm}$ is thwarted. Moreover, all group members are protected against false identification by $\mathcal{M}_{rm}$ by *proof of revocation*, formally.

Theorem 2. *Given the ciphertext $c = (c_1, c_2)$, where $c_1 = k\beta$, $c_2 = ky_m + \mathcal{ID}_i$ and c encrypts the public identity of the i-th group member, $\mathcal{ID}_i$, under the public key y_m of $\mathcal{M}$-server. The i-th member is protected against false identification by $\mathcal{M}$-server.*

Proof. The proof of revocation ensures that $\mathcal{M}$-server knows a given $x_m \in Z_q$ such that the point $c_2 - \mathcal{ID}_i$ is equal to $x_m c_1$ and that $y_m = x_m\beta$. Since c_2 and c_1 are public values generated by the transaction signer as $c_1 = k\beta$, $c_2 - \mathcal{ID}_i = ky_m$, therefore, under the assumption that the ECDL problem is intractable, the revealed $\mathcal{ID}'$ is equal to the encrypted identity $\mathcal{ID}_i$ if $\mathcal{ID}'$ is equal to $c_2 - x_m c_1$. Formally $c_2 - \mathcal{ID}_i = x_m c_1 \wedge y_m = x_m\beta$. Otherwise, $\mathcal{M}$-server has to find another x' that decrypts c to another identity and this contradicts the ECDL hardness assumption. □

Communication and computation overhead.

For each transaction, extra transaction parameters of 2 curve points and $2n + 3$ field elements are sent. The proof alone is of length $2n + 3$ elements and is linear in the size of the organizational group n. Assuming 128-bit security, we require $q = 256$ which gives an overall overhead of $2n + 7$ field elements. Using point compression, one point can be encoded in 32 bytes in addition to one bit to encode the least significant bit of the y coordinate of each point. Overall, the communication overhead is equal to $(2n + 5)$(32 bytes) + 2 bits. The computation cost of proving requires $4n$ point multiplications to compute the challenges and 3 field multiplications for the responses. Verifying cost is almost equal to proving cost with $4n + 3$ point multiplications.

How large can** n **be? the choice of n determines the size of the anonymity set, the size of the CRS as well as the communication and computation overheads. Primarily, for practical realization on Ethereum, we need the verifying execution gas cost to be less than the block gas limit with is slightly over 8 million. Given that one EC multiplication requires 40000 gas [49], n should less than 50. Nevertheless, we have researched the average number of major supply chain stakeholders (see Table 2), and based on these numbers, we recommend that organizations carry a hierarchical subgroup structures where each child subgroup is issued new blockchain credentials; such subgroups can be categorized under geographical locations around the world which makes their internal group identity management more logical.

6 Mesh **Protocol: RFID side**

Confidentiality of transactions is established by encrypting their payload using symmetric keys generated by the RFID tag. In what follows, we give the message flow for the mutual authentication between a the RFID tag and a reader that is operated by an owner member and how transfer of ownership is accomplished.

6.1 Tag-Reader Mutual Authentication

RFID authentication has been extensively studied in the literature for more than fifteen years [39, 44]. Most of such proposals avoided the use of cryptographic primitives and opted for simple operations to accommodated the constrained hardware area of the tag. However, as such tags are getting stronger, their standards are gradually defining cryptographic primitives. For instance, the second version of Electronic Product Code (EPC) Class 1 Gen 2 standard for RFID tags [7], defines pseudorandom bit generator, encryption and authentication functionalities. Also, the currently ongoing NIST lightweight cryptographic standardization competition aims for standardizing cryptographic primitives for tiny devices such as RFID tags. All of these recent indicators suggest that having crypto-enabled RFIDs is expected in the near future. EPC tags are heavily used in retail supply chains. Stateful Near Field Communication (NFC) tags are used in healthcare, pharmaceutical, and electronics supply chains. In what follows, we give quantitative metrics of both communication (data rate), computation (power consumption and security hardware area in GE) capabilities, security levels, dedicated investment in security of both EPC and RFID tags [7, 38]. Intuitively, if a given cryptographic primitive operates under such numbers, its implementation can be practically realized on these tags. Examples of such designs include [8, 34] which may realize more than one cryptographic functionality using the same circuitry.

Tag	Data rate kbps	Power	Security bits	Sec. area (approx)	Sec. Cost
NFC	106–424	125 mW - 1 W	128	5 kGE	$0.01
EPC	26.7–128	125 mW - 1 W	80	2 kGE	$0.01

Table 1. Bit security, and the corresponding dedicated GE area and cost in both NFC and RFID tags

In [38], Juels proposed the *"minimalist"* model for RFID tags which captures the asymmetry between reader-to-tag and tag-to-reader transmission ranges. Intuitively, readers transmit at much higher power than tags, thus, their messages are vulnerable to attacks at much greater distances than tag-to-reader communications. Figure 6 depicts a 4-move mutual authentication protocol that is secure against man-in-the middle and reply attacks because of the keyed MAC scheme and the use of fresh nonces. We assume that initially both tag and reader share t_i and Ak where t_i is an index to their shared key AK.

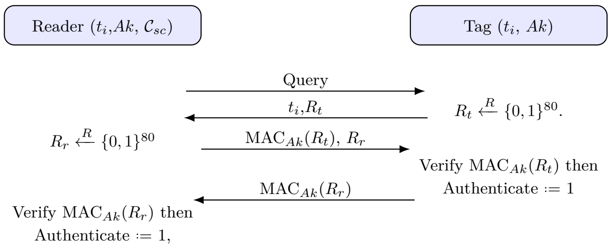
Reader (t_i,Ak, $\mathcal{C}_{sc}$)
Tag (t_i, Ak)
Query
t_i,R_t
$R_t \xleftarrow{R} \{0,1\}^{80}$.
$R_r \xleftarrow{R} \{0,1\}^{80}$
MAC$_{Ak}(R_t)$, R_r
Verify MAC$_{Ak}(R_t)$ then Authenticate := 1
MAC$_{Ak}(R_r)$
Verify MAC$_{Ak}(R_r)$ then Authenticate := 1,

Fig. 6. A 4-message mutual authentication protocol.

The tag maintains a counter *cn* that is initialized to zero and then incremented when it changes ownership. After executing the above protocol, a reader can issue one of the following two commands to the tag.

1. `setId`($\mathcal{C}_{sc}$). This command can only be issued by the first owner (i.e., $cn = 0$) and it instructs the tag to store the supply chain contract account as its unique identification code, after that, the counter is incremented.
2. `getId()` This command can be issued by any authorized reader at anytime and it returns $\mathcal{C}_{sc}$.

6.2 Transactions Confidentiality: *Forward Secrecy* on the Blockchain

We enforce *forward secrecy* where new owners can decrypt previously submitted information and past owners are prevented from accessing future information. We note that the adopted notion of forward secrecy is different than that of *Perfect Forward Secrecy* that is defined in [40] for key agreements which intuitively ensures the security of past session keys in the event of a compromise of the long term key which was used in generating them.

Transfer of Ownership. In order to enforce the defined notion of forward secrecy in our protocol, we need to first implement a transfer of ownership mechanism and couple it with a one way key chaining algorithm. Particularly, a transfer of ownership mechanism is an amendment to the mutual authentication protocol depicted in Figure 6. In Mesh we choose to implement it as a command, `newOwner()` that is issued by a new owner. This command instructs the tag to issue new authentication credentials to the new owner after being authenticated using the old credentials that have been handed to her by the old owner. In this context, we rely on the asymmetry between the eavesdropping ranges on the forward and backward channels as stated in the RFID minimalist model (see Section 3.5) to assume that an old owner can not intercept this one session. Once the transfer of ownership is established, the counter cn is incremented and a new encryption key is generated by the tag and securely sent to the reader.

Key chaining. A new owner should have access to all the previous encryption keys as well as her current new key. Accordingly, we let the tag generate all the keys sequentially which results in a key chain. More precisely, we assume that the tag stores a secret seed s and with each `newOwner()` command, it runs its Pseudo Random Bit Generator (PRBG) on this seed and cn to generate a new secret key. If $cn > 1$, then it generates all the previous keys to provide a new owner with a key chain that enables the decryption of all previous information.

Definition 7 (Key chain). *Given a secret seed s, and the number of all previous owners $cn - 1$, a keychain of size cn is denoted by $\mathcal{KC}_{cn}$ and is a set of keys $\{k_1, k_2, \cdots, k_{cn}\}$ that is generated by the tag to the cn-th owner where $k_i = PRBG(s \oplus i)$, and $i = 1, 2, \cdots, cn$.*

At any time, the $i^{th} - 1$ owner knows $\{k_1, k_2, \cdots, k_{i-1}\}$ and only the i-th owner learns k_i, hence, the notion of the forward secrecy that is given in Definition 9.

7 Security

Proofs for Theorem 3 (resp. 4) are given in Appendix D (resp. E). Other attack scenarios are discussed in Appendix F.

7.1 Plausible Anonymity

We define *plausible anonymity*, as a condition necessary for providing both local anonymity and accountability of transaction sender. Formally, we define it as follows.

Definition 8 (Plausible Anonymity of Transactions). *We say that a transaction from a wallet account $\mathcal{W}_O$ is plausibly anonymous if for any PPT adversary $\mathcal{A}$ who is not $\mathcal{M}_{rm}$ and does not engage with a transaction sender in a physical pickup or drop-off scenario, a security parameter λ, and the negligible function $\epsilon(.)$ the following two conditions hold:*

- *$\mathcal{A}$ cannot reveal the identity of the transaction sender, formally $Pr[(\mathcal{ID}) \leftarrow \mathcal{A}^{\mathcal{R}}(c, 1^\lambda)] \leq \epsilon(\lambda)$.*
- *$\mathcal{A}$ cannot cause an honest verifier to accept a transaction with c and π_{id} pair, where c encrypts a valid group $\mathcal{ID}$ for which $\mathcal{A}$ does not know the corresponding secret x. Formally $Pr[(\sigma, c, \pi_{id}) \leftarrow \mathcal{A}^{\mathcal{S}}(m, sk_o, 1^\lambda, y_m) : \mathcal{C}_{sc}\cdot VerifyId(m, c, \pi_{id}, \mathcal{ID}_{i=1}^n) = 1] \leq \epsilon(\lambda)$*

Theorem 3. *Assuming that Σ is a secure signature scheme,* Mesh *transactions achieve plausible anonymity under Definition 8.*

7.2 Forward Secrecy

Assuming that transaction information is encrypted using secret keys generated by the tag's PRBG with a secret seed, s, and a counter, cn, such that the i-th key $k_i = \mathsf{PRBG}(s \oplus i)$, our notion of forward secrecy is defined as follows:

Definition 9 (Mesh forward secrecy). *Given a product that is currently owned by the i-th owner in the supply chain, the i-th encrypted information c_i, and the symmetric encryption algorithm $\mathcal{E}_\lambda(.)$, for all polynomially bounded previous $l = i - 1$ owners and blockchain observer, no useful information can be deduced from c_i, while the i-th owner $\mathcal{I}$ can decrypt any c_{i-j}, where $j = 0, 1, \cdots, i-1$. Formally,* $\mathsf{PR}[m_i \leftarrow \mathcal{A}^{\mathcal{D}}(c_i, 1^\lambda)] \leq \epsilon(\lambda) \wedge \mathsf{PR}[m_i \leftarrow \mathcal{I}^{\mathcal{D}}(c_{i-j}, 1^\lambda)_{j=0}^{i-1}] = 1$, *where $\mathcal{D}(.)$ denotes the decryption function.*

Theorem 4. *Assuming that the encryption algorithm $\mathcal{E}_\lambda(.)$ is secure, the indistinguishability of the* PRBG*, the tag is physically secure, and that encryption keys are generated by the tag and known only by their respective owner,* Mesh *protocol achieves forward secrecy as defined in Definition 9.*

8 Performance Evaluation

We have implemented the proof of membership proving and verifying algorithms, and the new transaction signing scheme in C for the purpose of performance evaluation. We have used openSSL and crypto++ libraries, and 128-bit security Koblitz elliptic curve secp256k1[1] because it is the one used in Ethereum's ECDSA. All the reported measurements are taken on an Intel i7-6700U CPU 3.40 GHz and 16.0 GB RAM.

8.1 Measurements Criteria

We have fixed a message of size 128 bytes to satisfy the bounds of all the transaction fields [55], in addition to some bytes in the data field which where the update information is placed. A critical parameter that affects both the proof size, and proving and verifying

1 Recommended parameters can be found in Sec. 2.7 in http://www.secg.org/SEC2-Ver-1.0.pdf

Table 2. Number of members/facilities of some of the major supply chain stakeholders

Organization	Number of Members			
	N. America	S. America	Europe	Asia & Oceania
Walmart [6]	5768	4085	642	501
Amazon [45]	339	1	158	252
P&G [5]	130			
Nestlé [1]	158		146[a]	109
Mearsk [4]	639[b]			
India Post [3]	154910[c]			

[a] Including MENA region.
[b] Total number of fleet vessels.
[c] World's largest postal network.

timings and is the number of members in the group, n. Accordingly, we have searched for the number of facilities (e.g., warehouses, logistic vehicles, outlets, etc...) affiliated with some[2] of the world largest supply chain organizations in order to make our experiments realistic. Table 2 depicts the distribution of the number of members of such organizations over their main operative geographical regions.

Following the sample statistical distributions in Table 2, and as has been pointed out in Section 5.2, we decided to run our experiments on $n \leq 100$. Mainly because we need to optimize our efficiency and gas cost (See Sec. 5.2). The size of membership proof π_{id} is linear in the number of members and ranges from 736 bytes to 6.3 KB for groups of 10 to 100 members. On the other hand, proof of revocation π_r is always equal to 96 bytes. Figure 7 depicts the proving and verifying timings for $10 \leq n \leq 100$. Both timing are almost identical. Both these timings are linear in the number of members and asymptotically linear in circuit size, however the order of growth for reasonable sized groups is small because of the large cost of point multiplications. For the chosen range of n, proving and verification times vary between 4.7 ms and 19.5 ms.

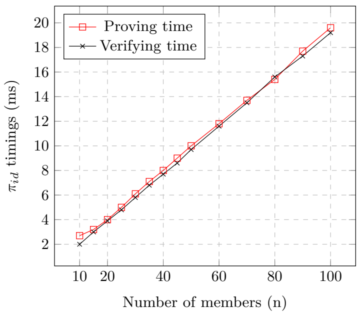

Fig. 7. Timing of membership proof.

2 We included organizations which have been affiliated with IBM's supply chain solution [32, 48] and BiTA [2]

Table 3. Comparison between membership proof and other generic proof systems for 10, 50, and 100 members membership arithmetic circuit

System	#Gates (N) (KG)[a]	π_r size (KB)	Timing		Trusted setup	Security Assumption
			Prove	Verify		
Membership proof	57	0.75	2.5 ms	2.0 ms	No	DLP
	262	3.2	10.0 ms	9.7 ms		
	518	6.3	19.6 ms	19.2 ms		
Bulletproofs [18][b]	57	1.4	44.8 s	2.4 s	No	DLP
	262	1.5	200.9 s	7.8 s		
	518	1.6	397.3 s	15.4 s		
zkSTRAKs[13][c]	57	>200	253 ms	66 ms	No	CSH
	262		517 ms	341 ms		
	518		825 ms	748 ms		
Bootle *et al.*[37][d]	57	3.4	15.5 s	5.2 s	No	DLP
	262	3.8	71.0 s	24.1 s		
	518	4.0	140.2 s	46.8 s		
zkSNARKs[15][e]	57	288 B	9.1 s	5 ms	Yes	KOE
	262		41.9 s	5.6 ms		
	518		82.9 s	6.4 ms		

[a] $N = 2560(2 + 2n)$ and the numbers are rounded below.
[b] Logarithmic communication of 2log(N)+13 elements. Prove and verify times are linear in N and are approximately evaluated from Table 3 in [18].
[c] All timings are approximated from timing graphs in [13].
[d] Logarithmic communication of 6log(N)+13 elements, prove (resp. verify) computation takes $12N$ (resp. $4N$) exponentiations, and one group exponentiation requires around 22.6 μs [37].
[e] Input x = 322, 1607, 3213 bytes, verifying takes 4.8 + 0.0005x, and proving takes 0.16 ms/gate [15].

8.2 Comparison to Generic Proof Systems

In order to compare we need to translate our *"I know x and k such that $c_2 = ky_m + x\alpha \wedge \exists|_i$: $ID_i = x\alpha$"* statement to an arithmetic circuit that is satisfied by our witnesses x and k. Such a statement involves witnesses to ElGamal encryption and proof of existential equality of EC discrete logarithm which are both computed by EC point multiplication. Given that on average 20λ arithmetic operations are required for one EC multiplication using projective coordinates [37], the number of circuit gates, henceforth denoted by N, of the resulting circuit blows up as n increases. We depict in Table 3 the lower bounds on the number of circuit gates using an estimate on the minimum number of EC multiplications which is approximated by $2n + 2$, where two point multiplications are required for proving the knowledge of representation of c_2 and $2n$ multiplication for the set existential equality. When $\lambda = 128$, one point multiplication requires around $20\lambda = 2560$ fan-in 2 multiplicative gates. We provide comparison measurements when $n = 10$, 50, and 100 which results in $N = 57$, 262, 518 Kilo Gates (KG). We compare the membership proof with the following systems:

- Bulletproofs [18]. At 128-bit security using th same Koblitz secp256k1 EC parameters as our implementation and a typical desktop.
- zkSTARK [13]. The systems is not EC-based an its security relies on the collision resistance of hash

functions. Reported numbers are at 80-bit security and the proving algorithm has been run on a powerful server with 32 AMD cores at clock speed of 3.2GHz, with 512GB of DDR3 RAM, so longer proving times are expected on typical desktops.

- Bootle *et al.* [37]. At 128-bit security using an OpenSSL curve. Since the exact curve is not explicitly mentioned in the paper and their experiments are over a 256-bit prime field, we assume that secp256k1 is used. Experiments are run on a typical desktop.
- zkSNARK [15]. At 128-bit security using a customized pairing-friendly parameterization of a Barreto-Naehrig curve over a 256-bit prime field (cf. Sec. 4.1 [15] and Sec. 3.1 in [14] for parameters and rational). Tests conducted on a typical desktop.

It can be seen that zkSNARKs offer the smallest proof size at just 288 bytes, and a slowly increasing verification timing, however, proving times are slow and grows linearly with gate count. On the other hand, π_{id} requires no trusted setup has a reasonable proof size at 6.3 KB for 2^{19} circuit and at the same time competitive proving and verifying timings at less than 6.5 ms. zkSTARK has large proof size of more than 200 KB and the proving and verifying times are relatively slow of around 800 ms 518 KG circuit. Both proving and verifying timings of π_{id} are significantly faster than those of Bulletproof and Bootle *et al.* systems for all circuit sizes. Even though the proof sizes of Bulletproof are less than those of π_{id}, proving times of around 7 minutes may be unpractical given the real time nature of supply chain activity.

8.3 Gas Cost Estimation

Since the main functionality of $\mathcal{C}_{sc}$ is verifying proofs, we have tested a prototype for the verification functionality by implementing a function that essentially performs $4n$ iterations of EC point multiplication and addition to compute the commitments for n member groups. The function references the helping pre-compiled contract EIP-196 [49] that implements EC basic operations on the Barreto-Naehrig curve over 256-bit prime field. In our experiment, we have created test-cases with $n = 10$, 15, and 20. We have chosen these number because one EC multiplication costs around 40000 gas [49], so larger groups $n > 50$ may not be realizable on Ethereum as the current block gas limit of 8000046. Table 4 shows the gas cost and the corresponding price in USD for both the transaction and execution of the function. At the time of carrying out our experiment, December, 2018, the ether exchange rate is 1 ether = 106 USD, and the median gas price is approximately 10 Gwei = 10×10^{-9} ether.

Table 4. Gas cost associated with the various functions

# Members	Transaction cost		Execution cost	
(n)	gas	USD	gas	USD
10	1727330	1.8	1719080	1.8
15	2575995	2.7	2406712	2.5
20	3434660	3.6	3405420	3.6

Recall that `verifyID` performs hashing on $2n + 4$ inputs, which if implemented using SHA3 Keccak-256, has a fixed gas cost of 30 + 6 gas per input. However, such a price is negligible compared to the $4n$ EC multiplications which dominates the overall cost. According to these results, if Mesh is implemented on Ethereum, we recommend that groups stay within the range of 20 to 30 members in order to reduce cost and optimize the efficiency given the required security in terms of the anonymity set size.

9 Conclusion

In this work, we have presented a solution for blockchain applications that require privacy of their transactions and accountability at the same time. In order to realize sender anonymity, we have utilized a membership proof that requires no trusted setup and its security relies on the discrete logarithm assumption. We have reported on implementation measurements and shown that such a proof is more efficient in terms of either communication or computation overheads than using generic proofs for arithmetic circuit satisfiability. Both overheads are linear in the number of set members. We have proposed Mesh to demonstrate a practical solution for supply chain management over public permissionless blockchains with a guarantee of plausible anonymity. We have shown that using RFIDs, confidentiality of transactions is provided using a lightweight symmetric key chaining mechanism. The security properties of Mesh are formally defined, security proofs, and arguments for adversarial scenarios are provided. For future work, we plan to provide an open source prototype for Mesh-server and perform practical analysis on the security models adopted in the RFID subsystem.

10 Acknowledgment

The work is supported by NSERC SPG research grant and NSERC PDF fellowship. The authors would like to thank Kalikinkar Mandal and Hisham S. Galal for their help in providing the computational results.

References

[1] Néstle annual review 2017, 2017. https://www.nestle.com/asset-library/documents/library/documents/annual_reports/2017-annual-review-en.pdf.

[2] Blockchain in Transport Alliance (BiTA), 2018. https://bita.studio/members/.

[3] India post annual report 2017, 2018. https://www.indiapost.gov.in/VAS/DOP_PDFFiles/AnnualReportEnglish2016-17.pdf.

[4] Maersk line, 2018. https://www.maersk.com/about.

[5] P&G manufacturing facilities locations, 2018. https://www.pgcareers.com/our-locations.

[6] Walmart location facts, 2018. https://corporate.walmart.com/our-story/our-locations.

[7] Epc™ radio-frequency identity protocols generation-2 uhf rfid standard version 2. EPCglobal Inc. Specification documents, Jul. 2018. https://www.gs1.org/sites/default/files/docs/epc/gs1-epc-gen2v2-uhf-airinterface_i21_r_2018-09-04.pdf.

[8] R. AlTawy, R. Rohit, M. He, K. Mandal, G. Yang, and G. Gong. sLiSCP-light: Towards hardware optimized sponge-specific cryptographic permutations. *ACM Trans. Embeded Computing Systems*, 17(4):81:1–81:26, 2018.

[9] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. Cocco, and J. Yellick. Hyperledger Fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, pages 30:1–30:15, 2018.

[10] Shireesh Aptea and Nikolai Petrovsky. Will blockchain technology revolutionize excipientsupply chain management? 7(3):76–78, 2016.

[11] Ghada Arfaoui, Jean-François Lalande, Jacques Traoré, Nicolas Desmoulins, Pascal Berthomé, and Said Gharout. A practical set-membership proof for privacy-preserving nfc mobile ticketing. In *PETs*, pages 25–45, 2015.

[12] E. Ben-Sasson, I. Bentov, A. Chiesa, A. Gabizon, D. Genkin, M. Hamilis, E. Pergament, M. Riabzev, M. Silberstein, E. Tromer, and M. Virza. Computational integrity with a public random string from quasi-linear pcps. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 551–579. Springer, 2017.

[13] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Report 2018/046, 2018. https://eprint.iacr.org/2018/046.

[14] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In R. Canetti and J. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, pages 90–108. Springer, 2013.

[15] Eli Ben-Sasson, Alessandro Chiesa, Eran Tromer, and Madars Virza. Succinct non-interactive zero knowledge for a von neumann architecture. Cryptology ePrint Archive, Report 2013/879, 2013. https://eprint.iacr.org/2013/879, *updated in 2015*.

[16] D. Boneh. The decision diffie-hellman problem. In J.. Buhler, editor, *Algorithmic Number Theory*, pages 48–63. Springer, 1998.

[17] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[18] B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *IEEE Symposium on Security and Privacy (SP)*, pages 315–334, 2018.

[19] Vitalik Buterin and Christian Reitwiessner. Eip 197: Precompiled contracts for optimal tate pairing check on the elliptic curve alt_bn128, 2017. https://eips.ethereum.org/EIPS/eip-197.

[20] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical report, Dept. of Computer Science, ETH Zurich., 1997.

[21] Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008*, pages 234–252. Springer, 2008.

[22] Sébastien Canard, Iwen Coisel, Amandine Jambert, and Jacques Traoré. New results for the practical use of range proofs. In Sokratis Katsikas and Isaac Agudo, editors, *Public Key Infrastructures, Services and Applications*, pages 47–64. Springer, 2014.

[23] Rafik Chaabouni, Helger Lipmaa, and Bingsheng Zhang. A non-interactive range proof with constant communication. In Angelos D. Keromytis, editor, *Financial Cryptography and Data Security*, pages 179–199. Springer, 2012.

[24] D. Chaum and E. van Heyst. Group signatures. In D. Davies, editor, *EUROCRYPT '91*, pages 257–265. Springer, 1991.

[25] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo G. Desmedt, editor, *CRYPTO*, pages 174–187. Springer, 1994.

[26] I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In J. Feigenbaum, editor, *Advances in Cryptology — CRYPTO '91*, pages 445–456. Springer, 1992.

[27] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *ACM Symposium on Theory of Computing*, STOC '90, pages 416–426. ACM, 1990.

[28] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. Odlyzko., editor, *Advances in Cryptology — CRYPTO' 86*, pages 186–194. Springer, 1987.

[29] E. Fujisaki and T. Okamoto. Statistical zero knowledge protocols to prove modular polynomial relations. In B. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 16–30. Springer, 1997.

[30] R. Gennaro, C. Gentry, B. Parno, and M. Raykova. Quadratic span programs and succinct NIZKs without PCPs.

In T. Johansson and P. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, pages 626–645. Springer, 2013.

[31] S Goldwasser, S Micali, and C Rackoff. The knowledge complexity of interactive proof-systems. In *The ACM Symposium on Theory of Computing*, STOC '85, pages 291–304. ACM, 1985.

[32] T. Groenfeldt. IBM ibm and Maersk apply blockchain to container shipping. https://www.forbes.com/sites/tomgroenfeldt/2017/03/05/ibm-and-maersk-apply-blockchain-to-container-shipping.

[33] J. Groth. Short pairing-based non-interactive zero-knowledge arguments. In M. Abe, editor, *Advances in Cryptology - ASIACRYPT 2010*, pages 321–340. Springer, 2010.

[34] J. Guo, T. Peyrin, and A. Poschmann. The photon family of lightweight hash functions. In P. Rogaway, editor, *CRYPTO*, pages 222–239. Springer, 2011.

[35] IBM.com. IBM Food Trust Marketplace. https://www.ibm.com/us-en/marketplace/food-trust/purchase, accessed: 22 Aug. 2018.

[36] Aferdita Ibrahimi. Cloud computing: Pricing model. *Journal of Advanced Computer Science and Applications*, 8(6):434–441, 2017.

[37] j. Bootle, A. Cerulli, P. Chaidos, J. Groth, and C. Petit. Efficient zero-knowledge arguments for arithmetic circuits in the discrete log setting. In M. Fischlin and JS. Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, pages 327–357. Springer, 2016.

[38] A. Juels. Minimalist cryptography for low-cost RFIS tags (extended abstract). In C. Blundo and S. Cimato, editors, *Security in Communication Networks*, pages 149–164. Springer, 2005.

[39] A. Juels and S.. Weis. Authenticating pervasive devices with human protocols. In V. Shoup, editor, *Advances in Cryptology – CRYPTO 2005*, pages 293–308. Springer, 2005.

[40] J. Katz, R. Ostrovsky, and M. Yung. Forward secrecy in password-only key exchange protocols. In *Proceedings of the 3rd International Conference on Security in Communication Networks*, SCN'02, pages 29–44. Springer, 2003.

[41] J. Kilian and E. Petrank. Identity escrow. In H. Krawczyk, editor, *Advances in Cryptology — CRYPTO '98*, pages 169–185. Springer, 1998.

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

[43] Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. In *2016 IEEE Symposium on Security and Privacy*, pages 839–858, 2016.

[44] Z. Li, G. Gong, and Z. Qin. Secure and efficient LCMQ entity authentication protocol. *IEEE Trans. Information Theory*, 59(6):4042–4054, 2013.

[45] mwpvl international. Amazon distribution network, 2018. http://www.mwpvl.com/html/amazon_com.html.

[46] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. In E. Brickell, editor, *Advances in Cryptology — CRYPTO' 92*, pages 31–53. Springer, 1993.

[47] H. Petersen. How to convert any digital signature scheme into a group signature scheme. In B. Christianson, B. Crispo, M. Lomas, and M. Roe, editors, *Security Protocols*, pages 177–190. Springer, 1998.

[48] B. Peterson. IBM told investors that it has over 400 blockchain clients — including Walmart, Visa, and Nestlé. https://www.businessinsider.com/ibm-blockchain-enterprise-customers-walmart-visa-nestl-2018-3?IR=T.

[49] Christian Reitwiessner. Eip 196: Precompiled contracts for addition and scalar multiplication on the elliptic curve alt_bn128, 2017. https://eips.ethereum.org/EIPS/eip-196.

[50] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT*, pages 552–565. Springer, 2001.

[51] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza. Zerocash: Decentralized anonymous payments from bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[52] C. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[53] B. Schoenmakers. Interval proofs revisited. In *International Workshop on Frontiers in Electronic Elections*, 2005.

[54] Nicolas van Saberhagen. Cryptonote v 2.0, 2013. https://cryptonote.org/whitepaper.pdf.

[55] GAVIN Wood. Ethereum: A secure decentralised generalised transaction ledger. 2014. http://gavwood.com/paper.pdf.

[56] Fan Zhang, Philip Daian, Gabriel Kaptchuk, Iddo Bentov, Ian Miers, and Ari Juels. Paralysis proofs: Secure dynamic access structures for cryptocurrencies and more, 2018. In Bitcoin magazine https://eprint.iacr.org/2018/096.pdf.

A ElGamal Encryption over EC

ElGamal has a 2:1 ciphertext plaintext ratio, thus, it makes sense to use EC instead of plain Galois fields.

Definition 10 (ElGamal over EC [42]). *Let $f(m) : m \to P_m$ denote a publicly known bijective function which maps a message m to a curve point P_m, the ElGamal scheme defines the following three procedures:*

- *Key generation: $(pk, sk) \leftarrow \mathcal{K}(1^\lambda)$. Given a primitive point $\alpha \in \mathcal{EC}$, choose a secret value $x \in_R Z_q$, and evaluate $y = x\alpha$. The public key $pk = (y, \alpha)$, and the secret key $sk = x$.*
- *Encryption: $(c_1, c_2) \leftarrow \mathcal{E}(pk, m)$. Randomly choose $k \in_R Z_q$, and calculate $c_1 = k\alpha$, and $c_2 = f(m) + ky$.*
- *Decryption: $m \leftarrow \mathcal{D}(sk, c_1, c_2)$. $m = f^{-1}(c_2 - xc_1)$.*

B Security Proofs for Existential Equality Proof

In what follows, we give the completeness, soundness and witness indistinguishability proofs for Definition 4.

Completeness. It can be seen that an honest prover will always succeed in constructing a valid proof since the following equivalences hold.

- For $i = 1$: $s'_1 = r_1\alpha - c_1 y = s_1$, $t'_1 = r_1\beta - c_1 y_1 = t_1$
- For $1 \le i \le n$: $s'_i = r_i\alpha - c_i y = s_i$, $t'_i = r_i\beta - c_i y_i = t_i$
  $\mathcal{H}(\alpha, \beta, y, y^n_{i=1}, t'^n_{i=1}, s'^n_{i=1}) = \sum^n_{i=1} c_i$

□

Soundness. Given $\mathcal{H}$ and following the Fiat-Shamir heuristic, it is assumed that the values $t^n_{i=1}$, and $s^n_{i=1}$ had been fixed before c was computed. Let us assume, if in an interactive setting, that the transaction signer is able to compute proofs for two different challenges (c^1, c^2) after committing the values $t^n_{i=1}$, and $s^n_{i=1}$ by evaluating the corresponding responses $r^n_{i=1,1}$ and $r^n_{i=1,2}$, where $r_{i,j}$ denote the i-th response to c^j, without knowing at least one secret x such that $y = x\alpha \wedge (\exists|_{1\le i\le n}$: $y_i = x\beta)\}$ and for all $i \in [1, n]$ and $j \in \{1, 2\}$, the following equivalences hold.

$$r_{i,j}\alpha = s_i + c^j_i y,\ r_{i,j}\beta = t_i + c^j_i y_i,\ c^j = \Sigma^n_{i=1} c^j_i$$

Then, there exists at lease one pair of tuples ($r_{i,1}$,c^1_i) and ($r_{i,2}$,c^2_i) such that $c^1_i \ne c^2_i$ and $c^1_i y = r_{i,1}\alpha - s_i$ and $c^2_i y = r_{i,2}\alpha - s_i$, and $c^1_i y_i = r_{i,1}\beta - t_i$ and $c^2_i y_i = r_{i,2}\beta - t_i$ which implies that $y(c^1_i - c^2_i) = (r_{i,1} - r_{i,2})\alpha$ and $y_i(c^1_i - c^2_i) = (r_{i,1} - r_{i,2})\beta$. In other words, the prover knows some $x = \frac{r_{i,1} - r_{i,2}}{c^1_i - ci^2}$ such that $y = x\alpha$ and $y_i = x\beta$, which contradicts the initial assumption that the signer does not know at least one secret x such that $y = x\alpha \wedge (\exists|_{1\le i\le n}$: $y_i = x\beta)\}$. □

Witness indistinguishablity. following Definition 3.1 in [[27]], given that the responses r_i for $i \in [2, n]$ are random values and r_1 is evaluated by adding a random value to the product of the witness and the challenge, and the challenges $c^n_{i=2}$ are also random and c_1 is random under the assumption that $\mathcal{H}(.)$ behaves like a random oracle [28], then it follows that all $s^n_{i=1}$ and $t^n_{i=1}$ are random. Therefore, all the transcripts (c_i, r_i, s_i) and (c_i, r_i, t_i) have the same distribution as in a single Schnorr signature [52], they additionally satisfy $c = \Sigma^n_{i=1} c_i$ which does not leak any useful information about the witness x as it is essentially the sum of random values. Even a distinguisher, who knows the possible witnesses, cannot tell which witness the prover knows. Thus, the proof is witness indistinguishable in the sense of [27]. □

C ECDL Membership Proof Security

Theorem 5. *Given a random collision resistant hash function $\mathcal{H}(.)$, the Proof of identity in Definition 5 is a sound, complete and witness indistinguishable proof of knowledge of the values $x_i, k \in Z_q$ satisfying $\mathcal{ID}_i = x_i\alpha$, $c_1 = k\beta$, ($c_2 = ky_m + \mathcal{ID}_i$) for some $i \in \{1, 2, \cdots, n\}$.*

Soundness. Since the hash function is preimage resistant, we can assume that the values u_1, u_2, $t^n_{i=1}$, and $s^n_{i=1}$ had been fixed before ch_r and ch_e were computed. Assuming that the transaction signer is able to compute proofs for two different challenges (ch^1_r, ch^1_e) and (ch^2_r, ch^2_e) after committing the values u_1, u_2, $t^n_{i=1}$, and $s^n_{i=1}$ by evaluating the corresponding responses $r^{n+2}_{i=1,1}$ and $r^{n+2}_{i=1,2}$ without knowing any of the secret identities x_i for some $i \in \{1, 2, \cdots, n\}$ and a value k such that for all $i \in [1, n]$ and $j \in \{1, 2\}$, the following equivalences hold.

$$r_{1,j}y_m + r_{2,j}\alpha = u_1 + u_2 + ch^j_r c_2$$
$$r_{i+2,j}\beta = t_{i,j} + ch^j_i c_1$$
$$r_{i+2,j}y_m = s_{i,j} + ch^j_i (c_2 - \mathcal{ID}_i),$$
$$ch^j_e = \sum^n_{i=1} ch^j_i$$

Then there exists at lease one tuple ($r_{1,1}$, $r_{2,1}$, $r_{i+2,1}$,ch^1_r, ch^1_e) and ($r_{1,2}$, $r_{2,2}$, $r_{i+2,2}$,ch^2_r, ch^2_e) such that the following equivalences hold.

1. $ch^1_r \ne ch^2_r, ch^1_e \ne ch^2_e$

2.a. $r_{1,1}y_m + r_{2,1}\alpha = u_1 + u_2 + ch^1_r c_2$

2.b. $r_{1,2}y_m + r_{2,2}\alpha = u_1 + u_2 + ch^2_r c_2$

3. $r_{i+2,1}\beta = t_i + ch^1_i c_1,\ r_{i+2,1}y_m = s_i + ch^1_i(c_2 - \mathcal{ID}_i)$

4. $r_{i+2,2}\beta = t_i + ch^2_i c_1,\ r_{i+2,2}y_m = s_i + ch^2_i(c_2 - \mathcal{ID}_i)$

(1), (2.a) and (2.b) imply that

5. $ch^1_r c_2 = r_{1,1}y_m + r_{2,1}\alpha - (u_1 + u_2)$

6. $ch^2_r c_2 = r_{1,2}y_m + r_{2,2}\alpha - (u_1 + u_2)$

by subtracting (6) from (5), we get

$c_2(ch^1_r - ch^2_r) = (r_{1,1} - r_{1,2})y_m + (r_{2,1} - r_{2,2})\alpha$, thus

$$c_2 = \frac{r_{1,1} - r_{1,2}}{ch^1_r - ch^2_r} y_m + \frac{r_{2,1} - r_{2,2}}{ch^1_r - ch^2_r}\alpha$$

In other words, a signer knows some

$k = \frac{r_{1,1} - r_{1,2}}{ch^1_r - ch^2_r}$ and $x = \frac{r_{2,1} - r_{2,2}}{ch^1_r - ch^2_r}$ such that

$c_2 = ky_m + x\alpha$

(1), (3), and (4) imply that

$$ch_i^1 c_1 = r_{i+2,1}\beta - t_i, ch_i^2 c_1 = r_{i+2,2}\beta - t_i$$
$$ch_i^1(c_2 - \mathcal{ID}_i) = r_{i+2,1}y_m - s_i,$$
$$ch_i^2(c_2 - \mathcal{ID}_i) = r_{i+2,2}y_m - s_i, \text{ thus,}$$
$$c_1(ch_i^1 - ch_i^2) = (r_{i+2,1} - r_{i+2,2})\beta$$
$$(c_2 - \mathcal{ID}_i)(ch_i^1 - ch_i^2) = (r_{i+2,1} - r_{i+2,2})y_m$$

It follows that the signer knows some

$$k = \frac{r_{i+2,1} - r_{i+2,2}}{ch_i^1 - ch_i^2} \text{ such that } c_1 = k\beta \wedge c_2 - \mathcal{ID}_i = ky_m$$

and that for this given $(c, \mathcal{ID}_i)$ pair, $k = \frac{r_{1,1}-r_{1,2}}{ch_r^1 - ch_r^2} = \frac{r_{i+2,1}-r_{i+2,2}}{ch_i^1-ch_i^2}$, and $\exists|_i$ s.t $ID_i = x_i\alpha \wedge x_i = \frac{r_{2,1}-r_{2,2}}{ch_r^1-ch_r^2}$, which contradicts the initial assumption that the signer does not know any secret identity x_i for some $i \in \{1, 2, \cdots, n\}$ and the value of k. □

Completeness. It can be seen that an honest prover will always succeed in constructing a valid proof since the following equivalences hold.

$$\begin{aligned}
\text{- } u_1' + u_2' &= r_1 y_m + r_2\alpha - ch_r c_2 \\
&= (z_1 + k ch_r)y_m + (z_2 + x_i ch_r)\alpha \\
&\quad - ch_r(ky_m + x_i\alpha) \\
&= z_1 y_m + k ch_r y_m + z_2\alpha + x_i ch_r\alpha - ch_r k y_m \\
&\quad - ch_r x_i \alpha \\
&= z_1 y_m + z_2\alpha \\
&= u_1 + u_2
\end{aligned}$$

Therefore $\mathcal{H}(m, y_m, \alpha, c_2, u_1' + u_2') = ch_r$.
- For $i = 1$:

$$\begin{aligned}
t_1' &= r_3\beta - ch_1 c_1 = (v_1 + k ch_1)\beta - ch_1(k\beta) \\
&= v_1\beta + k ch_1\beta - ch_1 k\beta = v_1\beta \\
&= t_1 \\
s_1' &= r_3 y_m - ch_1(c_2 - \mathcal{ID}_1) \\
&= (v_1 + k ch_1)y_m - ch_1(ky_m) \\
&= v_1 y_m + k ch_1 y_m - ch_1 k y_m = v_1 y_m \\
&= s_1
\end{aligned}$$

- For $1 \leq i \leq n$:

$$\begin{aligned}
t_i' &= r_{i+2}\beta - ch_i c_1 = (v_i)\beta - w_i c_1 \\
&= t_i \\
s_i' &= r_{i+2}y_m - ch_i(c_2 - \mathcal{ID}_i) = (v_i)y_m - w_i(c_2 - \mathcal{ID}_i) \\
&= s_i
\end{aligned}$$

Therefore $\mathcal{H}(m, \beta, y_m, c_1, (c_2 - \mathcal{ID}_i)_{i=1}^n, t_{i=1}^{'n}, s_{i=1}^{'n}) = \Sigma_{i=1}^n ch_i$. □

Witness indistinguishablity. following Definition 3.1 in [27], all the responses r_i for $i \in [1, n+2]$ are either a random value or evaluated by adding a random value to the product of the witness and the challenge. Given that challenges are also random under the assumption that $\mathcal{H}(.)$ behaves like a random oracle, all the tuples $(r_1, r_2, ch_e, r_i + 2, ch_i)$ have the same distribution, they additionally satisfy $ch_e = \Sigma_{i=1}^n ch_i$ which does not leak any information about the witness, x_i, k. Therefore, the proof is witness indistinguishable as the verifier learns no information about which witness the prover knows. □

D Proof of Theorem 3

Proof. For the first condition in Definition 8, $\mathcal{A}$ may attempt either of the following:

- $\mathcal{A}$ outputs the correct identity $\mathcal{ID}$ by running the revocation algorithm $\mathcal{R}(.)$ on the observed c. However, since $c = (c_1, c_2) = (k\beta, ky_m + \mathcal{ID})$ is ElGamal EC encryption of the identity $\mathcal{ID}$, and ElGamal encryption is proven to be semantically secure under the decisional Diffie-Helman assumption, then the success probability of $\mathcal{A}$ is reduced to the success probability of a simulator $\mathcal{B}$ that only has access to any auxiliary information about $\mathcal{ID}$, e.g., its length or more generally $h(\mathcal{ID})$ where $h(.)$ is any polynomially bounded function. Formally, $Pr[\mathcal{A}^{\mathcal{R}}(1^\lambda, c, h(\mathcal{ID})) = \mathcal{ID}] - Pr[\mathcal{B}^{\mathcal{R}}(1^\lambda, h(\mathcal{ID})) = \mathcal{ID}] \leq \epsilon(\lambda)$
- $\mathcal{A}$ tries to establish a link between c and the identity by testing c against encrypting $c_i = \mathcal{E}_{y_m}(\mathcal{ID}_i)$ for $1 \leq i \leq n$. Since the encryption scheme is probabilistic where a fresh secret nonce is used with each c, it follows that the encryption of the same identity results in different c's with each transaction.

For the second condition in Definition 8, $\mathcal{A}$ is restricted the set of adversaries to those who have access to the wallet $\mathcal{W}_O$ signing credentials, i.e., members of the same group who can produce valid signatures. Otherwise, $\mathcal{A}$ may need to break the signing scheme and produce a forged signature $\Sigma_{sk_o'}(\mathcal{H}(m, c))$ without the knowledge of sk_o and this contradicts the assumption that Σ is a secure signature scheme. $\mathcal{A}$ may attempt either of the following:

- Without knowing the secret identity x where $ID = x\alpha$, $\mathcal{A}$ can generate a given k, evaluate c, and try evaluating π_{id} such that $\mathcal{C}_{sc} \cdot \texttt{VerifyId}(m, c, \pi_{id}, \mathcal{ID}_{i=1}^{n}) = 1$. In this case, $\mathcal{A}$ is able to generate a valid proof of knowledge π_{id} to a witness x that she does not know, which according to the soundness proof in Appendix C can happen with negligible probability. Formally, $Pr[\mathcal{C}_{sc} \cdot \texttt{VerifyId}(m, c, \pi_{id}, \mathcal{ID}_{i=1}^{n}) = 1 \wedge c = \mathcal{E}_{y_m}(k, x\alpha) \wedge \pi_{id} = \mathsf{PoK}(k, x') \wedge x' \neq x] \leq \epsilon(\lambda)$.

– $\mathcal{A}$ tries to use a recorded c and π_{id} to send a transaction and not be held accountable. Recall that the proof generation algorithm in Definition 5 evaluates the challenge by incorporating the transaction payload m in $\mathcal{H}(.)$ which results in a message dependent proof. In other words, assuming that $\mathcal{H}(.)$ is collision resistant hash function, with negligible probability, $\mathcal{A}$ is able to replay a previous π_{id} in a new transaction.

□

E Proof of Theorem 4

Proof. (Sketch) Since the encryption keys are generated by the tag using a securely stored secret seed s and a publicly known counter cn, the result of xoring s by cn is still secret, and accordingly, each new encryption key k_i is generated by a secret seed that is not known to the previous l owners and all blockchain observers. Additionally, given the indistinguishability assumption, it follows that even that inputs of the PRBG are related, the corresponding outputs are not. □

F Adversarial Scenarios.

Plausible anonymity ensures that members are held accountable to their transactions and that their identity is locally anonymized within their owner groups. Forward secrecy guarantees the confidentiality of the transaction information. In what follows, we list some security concerns in the form of adversarial scenarios and show how Mesh handles them.

Corrupted $\mathcal{M}_{rm}$. Mesh allows a revocation manager $\mathcal{M}_{rm}$ to reveal this identity when a request is raised by the contract. A corrupted manager can attempt to frame another member by responding with a different identity than the decrypted one. Mesh mitigates such an attack by requiring that $\mathcal{M}_{rm}$ provides a proof of correct decryption π_r (see Section 5) with the revealed identity. According to theorem 2, the success of $\mathcal{M}_{rm}$ in revealing a different identity and providing a valid π_r is equivalent to solving the ECDL problem.

Malicious Owners. A corrupted owner may try to interfere with the normal execution of the protocol in several scenarios. Nevertheless, similar to real world contracts, smart contracts can be implemented to enforce penalties on owners that deviate from the expected procedure. Assuming the majority of owners are honest and are looking for the best outcome for their business, such penalties can be enforced by a t out of N multisignature threshold scheme. As indicated earlier, we assume multisignature transactions are used for transfer of product ownership. Accordingly, a penalized owner has to have owned the product at some point, and given the honest majority assumption, contract penalties are assumed to be fairly enforced. Owner adversarial scenarios are listed below.

– Impersonate and frame an honest member which according to Theorem 3 may happen with negligible probability.
– Revoke the confidentiality of updates by submitting information in the clear. In this case, the identity of the owner can be revealed. In other words, because of the associated proof of identity, π_{id}, the contract accepts update information only after verifying π_{id}, thus a member cannot repudiate the transaction. In such scenario, we assume certain penalties can be enforced by the contract when prompted the majority of authorized owners.
– Use different key than the one produced by the tag or invoke `newOwner()` multiple times to get multiple keys and corrupt owner count. In such a scenario, a corrupted owner may deny the remaining rightful owners from accessing history information on the blockchain or cause confusion in the event of multiple keys. However, such an owner can directly be identified by the following owner who is expected to check the product information on the blockchain by decrypting it using $\mathcal{KC}$ that is generated by the product tag, then penalties may be executed by majority of owners. If not identified then all following owners that do not identify a previous violation share the penalties.
– Request the revocation of anonymity of a previous owner. The contract accepts revocation requests from authorized owners only after the product has reached its final destination (i.e., currently owned by the last owner in the list). In such a case, anonymity revocation requests do not pose security violation.