# Endorser Peer Anonymization in Hyperledger Fabric for Consortium of Organizations

Dharani J[a], Sundarakantham K[a], Kunwar Singh[b], Mercy Shalinie S (Senior Member IEEE)[a]

[a]*Department of Computer Science and Engineering, Thiagarajar College of Engineering, Madurai, India*
[b]*Department of Computer Science and Engineering, National Institute of Technology, Trichy, India*

**Abstract**

Hyperledger Fabric is a unique permissioned platform for implementing blockchain in a consortium. It has a distinct transaction flow of execute-order-validate. During the execution phase, a pre-determined set of endorsing peers execute a transaction and sign the transaction response. This process is termed endorsement. In the validation phase, peers validate the transaction with reference to an endorsement policy. The identity of the endorsing organizations is obtainable to all the nodes in the network through the endorser signature and endorsement policy. Knowing this has led to serious vulnerabilities in the blockchain network.
In this paper, we propose a privacy-preserving endorsement system which conceals both endorser signature and endorsement policy. Endorser is anonymized by replacing the signature scheme with a scoped-linkable threshold ring signature scheme. Endorsement policy is secured using Pedersen commitments and non-interactive proof of knowledge of integer vector. We also achieve efficiency in the computation by employing non-interactive proof of co-prime roots. We provide the necessary security analysis to prove that the proposed work guarantees anonymity and unlinkability properties. A comparative analysis of our work with an existing framework is provided which shows that the proposed scheme offers higher level of security and it is optimal in terms of efficiency.

*Keywords:* Hyperledger Fabric, blockchain, linkable threshold ring signature, endorsement policy, non-interactive zero knowledge, proof of knowledge, privacy-preserving endorsement system

## 1. Introduction

Etymology dictates the origin of ledger as a service book with substantial magnitude that is kept in one place with open access. With time, physical ledgers hosted accounting information in an orderly fashion. Blockchain is a digital ledger that stores transactional data in a time-stamped manner. It is distributive in nature and requires all or majority of the participants to agree on the data being stored through consensus. The technology guarantees non-repudiation to the participants through the use of strong cryptographic hash functions. It constitutes chain of blocks with each block containing a set of transactions and the connectivity comes with every block storing the hash of previous block. There are three important variants of blockchain based on the need for authorization:

1. Public or Permissionless Blockchain: Any node can join the network anytime without any prior permission.
2. Private or Permissioned Blockchain: Only authorized entities within an organization are allowed to join the network.
3. Consortium Blockchain: This is a variant of permissioned blockchain where consortium of organizations are generally the participants of this network.

### *1.1. Hyperledger Fabric*

Hyperledger Fabric [1] an open source blockchain platform hosted by Linux Foundation is a consortium blockchain framework. It has a membership service provider that provides the identities and credentials to each of the engaging organization. It stands out from other permissioned framework for its unique transaction flow and the various privacy and security options offered by it: i)*Channels* allow a subset of parties to communicate without the other members even knowing the existence of such a channel, ii)*Identity Mixer / Idemix* to anonymize the clients with a zero-knowledge proof based signature scheme by Camenisch et. al., [2], and provides iii)*SideDB* for the participants to store sensitive information locally, with only the hashes of private information stored on-chain. Hyperledger as a community evolves continuously with researchers contributing to the current limitations [3]. Fabric Architecture workflow is depicted in figure 1

1. Organization 1 requests the services of Hyperledger Fabric through the Fabric client application.
2. The client sends a transaction proposal to the peers in the Fabric network as dictated by the endorsement policy.
3. Peers check the identity and signature of the client and proceed further only if it is a registered client. Peers may be Endorsers or Committers. Endorser peers hold the copy of chaincode which is executed and the peers sign the results and send it back to the clients. *Not all peers* execute the chaincode but only a subset based on the endorsement policy performs execution. Endorsement policy is a monotone logical expression of policy

*Email addresses:* `jdharani791@gmail.com` ( Dharani J), `kskcse@tce.edu` (Sundarakantham K), `kunwar@nitt.edu` (Kunwar Singh), `shalinie@tce.edu` (Mercy Shalinie S (Senior Member IEEE))

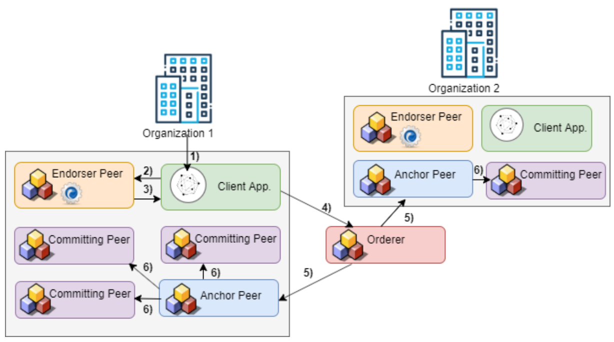


Figure 1: Hyperledger Fabric Workflow

principals such as "two out of three" or "$(Org1.peer \vee Org2.peer) or (Org1.member \wedge Org3.member)$". By allowing only a subset of the endorsers to execute a transaction fine grained privacy is guaranteed as other permissioned blockchain frameworks require all the nodes in the network to execute the transaction.

4. On receiving the proposal response message the client forwards the transaction to the orderer.
5. The role of the orderer is to batch the transactions according to the timestamps into a block and broadcast the block onto the network for validation
6. Peers check the validity of transactions in the block. Validation ensures i) if every transaction has received sufficient endorsement (by checking the identity and signature of the endorsers) and ii) the key version number in the key-value store. The second check guarantees if the data read during the chaincode execution remains the same during the endorsement. If both the checks are passed then a transaction is marked as valid and committed in the ledger. Else, the transaction is marked as invalid and is not updated in the ledger state database.

Finally, the client is notified about the transaction status.

### *1.2. Limitations of Existing Architecture*

The endorser signature and endorsement policy leaks the respective organization and may lead to vulnerabilities such as DDoS attack. Andola et.al., [4] showed the effects of DDoS attack on the endorsing nodes of Hyperledger Fabric. The latency of the system is increased by 1.044 seconds and the number of transactions reduced from 125 to 100 per second.
Apart from securing the endorsers from any attack certain sensitive applications demand peer anonymization as a privacy feature. Consider a voting scenario in Fabric framework, where the participating organizations cast their votes by executing the respective chaincode on the endorser peers. During the validation process all the peers (of participating organizations) check the results of the transaction along with endorser signatures, i.e., in this context the choice of endorser organizations are revealed. Hence such applications demand that the endorsers be anonymized.
According to the Fabric community [3], the problem of securing endorser signature and endorsement policy is an open problem. This paper is motivated by the stated privacy challenges prevailing in Hyperledger Fabric. The paper proposes a solution with threshold ring signature scheme that is well known for screening the identity of the signers. It is also important in the above mentioned context that same endorsers should not be allowed to cast their votes more than once. For this, a scheme that prevents an endorser from signing more than once should be in place. This is achieved by employing a *linkable* threshold ring signature. The endorsement policies are secured through commitment scheme and a membership proof is generated by every endorser that the generated ring signature belongs to the committed organization in the endorsement policy.

### *1.3. Our Contributions*

- *Endorser anonymization* is achieved in Hyperledger Fabric by employing a *(transaction-level) linkable threshold ring signature scheme* which does not require any trusted key-dealer in a threshold setting.
- The scheme is efficient as it employs linkable threshold ring signatures of size $O(t)$ from logarithmic size individual signatures.
- Non-interactive zero-knowledge proof of knowledge of integer vector is employed along with Pedersen commitment to achieve a privacy-preserving endorsement policy.
- We provide formal security proofs for the privacy-preserving endorsement system in Hyperledger Fabric based on the security definitions given in [5].

## 2. Literature Review

In 2001, the idea of ring signatures were first proposed by Rivest et. al., [6] that allowed a signer to conceal its identity amongst a ring or group of signers. Originally it was developed for whistle-blowing applications that involved sending a piece of information from an authorized group but screening the exact individual who sent it. Initially security of ring signatures were proven secure only under random oracle model [7, 8]. Eventually ring signature schemes secure under standard model were developed but with the overhead of signature size being dependent on the ring size [9, 10]. Linkable ring signatures were first developed by Liu et al.,[11] which is more suitable for e-voting applications [7] that guarantees no individual can caste their vote twice. Recently Backes et. al., [12] proposed a linkable ring signature scheme where the signature size is logarithmic in the number of signers.
Threshold variants of ring signature [13] allowed t-out-of-n ($t < n$) participants to cooperate in the signing process. Literature has threshold ring signatures secure under random oracle model [14, 15, 16], standard model [17, 18, 19, 20] and post-quantum model [21, 22, 23, 24]. The schemes listed above suffer from atleast one of the shortcomings: i) linear signature size ii) secure only under general assumptions or generic group model iii) requires a trusted key dealer iv) needs interaction among signers to cooperate in the signing process. Recently, construction by Haque et.al., [25] overcomes all the aforesaid shortcomings

and provides threshold ring signature that are linear in the size of threshold $O(t)$ .
The first use of threshold ring signatures for Hyperledger Fabric was extensively studied in [26]. A major problem in implementing threshold signature schemes for blockchains is that, a trusted entity is needed to assemble the signature shares for verification. In handling this issue, the authors suggest that a submitting peer at the client interface will perform the aggregation of signatures. Threshold version of RSA and BLS signatures are employed in [26]. The drawbacks of these threshold signature schemes are two-fold: attributability of the signature and unlinkability of endorsers; i.e., how to trace a signature to its endorser if the need arises and how to ensure that one cannot tell if the same endorser endorsed two transactions.
In [27], Mazumdar and Ruj proposed an anonymous endorsement system for Fabric by constructing a constant-sized linkable ring signature (FCsLRS) to address the problem of endorser anonymization. The scheme is efficient as it generates constant sized signatures. Also, transaction oriented linkability is provided to avoid double signing (a construct similar to double spending) by the same endorser on the same transaction. The signature scheme is secure only under random oracle model. Authors have suggested employing a threshold version of the signature scheme that is efficient and secure under standard model as future work. In [28], a linkable threshold ring signature scheme is employed to anonymize the endorsers but results in an inefficient signature size and verification time. In a subsequent work [29], the authors employ an efficient signature scheme but prove its security under generic group model.
We propose an anonymization technique that makes use of a transaction-level linkable threshold ring signature scheme based on [25] that requires no set up, no trusted key dealer, non-interactive i.e., no interaction is needed among the signers, accountable, and is secure under standard model.

## 2.1. Preliminaries

*Notations*: The security parameter is denoted by $\lambda$. PPT indicates probabilistic polynomial time and the notation $out \leftarrow A(in)$ represents a probabilistic algorithm which takes the parameter $in$ as input and outputs $out$. The algorithm A here $out \leftarrow A(in; r)$ additionally takes as input the randomness used to generate the output. A function $neglgbl(.) : \mathbb{N} \rightarrow \mathbb{R}^{+}$ is called `negligible` if for all $c > 0$ there exists a $k_0$ such that $neglgbl(k) < 1/k^c \ \forall k > k_0$. Any function that is exponentially small is termed as a negligible function.
Preliminaries can be found in Appendix A.

### 2.1.1. Linkable Threshold Ring Signature

The proposed Endorser Anonymization Technique adapts to a linkable threshold ring signature scheme from [25] which in turn takes the framework from [12]. The generic construction of the scheme is presented below. The scheme employs the following components: i) $VRF$ - verifiable random function ii) $PKE$ - public key encryption scheme iii)$SPB$ - somewhere perfectly binding hashing iv) $F$ - one-way permutation function v) $NIWI$ - non-interactive witness indistinguishable proofs.

**Algorithm 1:** LTRS.KeyGen($1^\lambda$)

```
$(vk, sk) \leftarrow Gen_{VRF}(1^\lambda)$;
$(vk_L, sk_L) \leftarrow Gen_{VRF}(1^\lambda)$;
$(vk_{mal}, sk_{mal}) \leftarrow Gen_{VRF}(1^\lambda)$;
$(pk_1, sk_1) \leftarrow Gen_{PKE}(1^\lambda)$;
$(pk_2, sk_2) \leftarrow Gen_{PKE}(1^\lambda)$;
$sk_F \leftarrow \{0,1\}^{2\lambda}$;
$r_E \leftarrow PKE.R$;
$CT \leftarrow Enc(pk_2, sk_F; r_E)$;
$VK := (vk, vk_L, vk_{mal}, pk_1, pk_2, CT)$;
$SK := (sk, sk_L, sk_{mal}, sk_1, sk_2, r_E, VK)$;
Result: return (VK,SK).
```

Algorithm 1 illustrates the generation of a pair of verification and secret key $(VK, SK)$ by every signer. $VK$ is a composition of three verification keys from verifiable random function (VRF), two public keys from public key encryption (PKE) and $CT$ which is a commitment to a secret key of one-way permutation function $sk_F$. The anonymity property of ring signature scheme is guaranteed through the witness indistinguishability of NIWI proofs. $sk_F$ helps in this anonymity proof generation to switch between the witnesses without the use of a common reference string and trapdoor information. Secret key $SK$ is a composition of corresponding secret keys along with the verification key $VK$.
On input, a message $m$, a VRF generates a value and proof $(o, p)$. Given message $m$, $(o, p)$ and the verification key $vk$ anyone can verify the correctness of the evaluation. When proof $p$ is not known for a corresponding value $o$, the value looks pseudorandom. This is the idea behind the adapted threshold ring signature where the proof $p$ is encrypted to achieve (linkable) anonymity.
Algorithm 2 illustrates the scoped linkable threshold ring signature generation process. Scoped linkability is a fine grained notion of linkability. Signatures produced by same signer on the same scope is linkable, while signatures by same signers on different scopes are valid and remains unlinkable. In the context of blockchain, scope is defined as a transaction and each transaction has a transaction identifier $tid$. Signing algorithm takes as input the message $m$, signer secret key $SK$, ring of verification keys $R$, threshold value $t$ and the transaction id $tid$. Next, four pairs of VRF evaluations are generated: $(o, p); (o', p'); (o_L, p_L); (o_{mal}, p_{mal})$. First one is for the message $m$, second one is for the individual threshold $t^s$, next is for scope id to achieve scoped linkability and final pair is to guarantee non-malleability of NIWI proof. The individual threshold $t^s$ allows each signer to decide how many other signers should contribute in the signing process to make the threshold signature valid such that the individual thresholds together comes up to the total threshold $t$. To achieve non-slanderability property of linkable threshold ring signature, non-malleability of NIWI proof needs to be guaranteed. Non-slanderability ensures no efficient adversary could generate a malicious signature such that it links to an honestly generated signature. Non-malleability of NIWI proof is achieved by the use of strong one-time signa-

ture. As mentioned above, only the values of VRFs are available and the proofs are encrypted and available as cipher texts $ct, ct', ct_L, ct_{mal}$ respectively. PKE.R represents the randomness used in the encryption scheme PKE.

NIWI proof is used to prove that the signature is valid under the committed verification key of the signer. For this two hashing keys are chosen $hk^s$ and $hk^i$ that are computationally binding at indices $s$ and $i$ respectively. The first one corresponds to the signer and the other is random. Next, hash digests are calculated as $h^s = Hash(hk^s, R)$ and $h^i = Hash(hk^i, R)$. Verification keys $VK^s$ and $VK^i$ are committed to as $(hk^s, h^s); (hk^i, h^i)$. Next, the signer calculates a NIWI proof for statement $x$ and witness $w$. Next, the partial signature $\rho$ is generated which is a composition of VRF evaluations: unique value $o$ for the message $m$, unique value $o'$ for the individual threshold $t^s$, unique value $o_L$ for the transaction id $tid$, $o_{mal}$ to achieve non-malleability; corresponding encryption of VRF proofs; hashing keys; NIWI proof and transaction id $tid$. To achieve non-malleability, the partial signature is signed with the one-time signature. Finally, the ring signature $\sigma^s$ is generated, that contains the partial signature, one-time signature on the partial signature and the verification key of one-time signature. Each signer broadcasts its signature $\sigma^s$. The final threshold ring signature is a concatenation of all the individual signatures: $\sigma = (\sigma^1, \sigma^2, ..., \sigma^t)$.

The algorithm 3 verifies the threshold ring signature and outputs 1 if it is valid or 0 otherwise. Validators verify every $\sigma_j$ upto the threshold $t$. First it checks on NIWI proof and subsequently verifies the one-time signature. Next, it verifies if the VRF value is unique for each endorsement to avoid double-signing. Finally, counts the individual $\sigma^j$s to check if the count is greater than or equal to the threshold $t$. And if all of this verifies to 1 then the threshold signature $\sigma$ is accepted. Algorithm 4 outputs *linked* if the same signer has signed for a scope more than once or *unlinked* otherwise.

Linkable threshold ring signatures should satisfy the following properties:

- Correctness: It ensures that when the key generation algorithm and signature generation algorithms are executed promptly, then the verification algorithm accepts the generated signature with overwhelming probability.

- Unforgeability: It guarantees that no PPT adversary can successfully generate a valid signature $\sigma_A$ on a message $m$ with respect to a ring $R$ such that $Verify(m, \sigma_A, R) = 1$ for $VK_A \notin R$.

- Linkable Anonymity: Anonymity of ring signatures guarantees that the signature does not reveal the signer's identity. Linkable anonymity guarantees signer anonymity but allows anyone to check if two signatures were generated by the same ring member.

- Transaction-oriented linkability: It ensures no two signatures were generated by the same endorser for same transaction.

**Algorithm 2:** LTRS.Sign($m, SK, R, t, tid$))

```
//Every signer s $\in S, |S| \geq t$
$o \leftarrow Eval(sk^s, m||R)$;
$p \leftarrow Prove(sk^s, m||R)$;
$o' \leftarrow Eval(sk^s, t^s||m||R)$;
$p' \leftarrow Prove(sk^s, t^s||m||R)$;
$o_L \leftarrow Eval(sk^s_L, tid)$;
$p_L \leftarrow Prove(sk^s_L, tid)$;
$(vk_{sOTS}, sk_{sOTS}) \leftarrow Gen_{sOTS}(1^\lambda)$;
$o_{mal} \leftarrow Eval(sk^s_{mal}, vk_{sOTS})$;
$p_{mal} \leftarrow Prove(sk^s_{mal}, vk_{sOTS})$;
$r_{ct}, r_L, r_{mal} \leftarrow PKE.R$;
$ct \leftarrow Enc(pk^s_1, p; r_{ct})$;
$ct_L \leftarrow Enc(pk^s_1, p_L; r_L)$;
$ct_{mal} \leftarrow Enc(pk^s_1, p_{mal}; r_{mal})$;
$(hk^s, shk^s) \leftarrow Gen_{SPB}(1^\lambda, N, s)$;
$h^s \leftarrow Hash(hk^s, R)$;
$\tau^s \leftarrow Open(hk^s, shk^s, R, s)$;
//Pick other ring member $i \neq s$
$i \leftarrow [N] \backslash s$;
$r_{E_0}, r_{E_1} \leftarrow PKE.R$;
$(hk^i, shk^i) \leftarrow Gen_{SPB}(1^\lambda, N, i)$;
$h^i \leftarrow Hash(hk^i, R)$;
$\tau^i \leftarrow Open(hk^i, shk^i, R, i)$;
//Call on the NIWI for the statement $x$ and $w$
$x = (m, R, o, o_{mal}, o_L, h^s, h^i, hk^s, hk^i, vk_{sOTS},$
     $ct, ct_{mal}, tid, ct_L, \varsigma_{sOTS})$
$w = (VK^s, VK^i, s, i, \tau^s, \tau^i, p, p_{mal}, p_L, sk^{i_0}_F, sk^{i_1}_F,$
     $r_{ct}, r_{E_0}, r_{E_1}, r_L, r_{mal})$
$\pi \leftarrow NIWI.Prove(x, w)$;
$\rho := (o, o', o_L, o_{mal}, ct, ct', ct_L, ct_{mal}, hk^s, hk^i, \pi, t^s, tid)$;
$\varsigma \leftarrow Sign_{sOTS}(sk_{sOTS}, \rho)$;
$\sigma^s := (\rho, \varsigma, pk_{sOTS})$;
//Final threshold ring signature constructed by the
  client
Result: return $\sigma = \{\sigma^j\}^t_{j=1}$.
```

**Algorithm 3:** LTRS.Verify($m, R, \sigma, t, tid$)

```
//Parse signature
$\sigma = (\rho, \varsigma, pk_{sOTS})_{j=1}^{t}$;
$\rho := (o, o', o_L, o_{mal}, ct, ct', ct_L, ct_{mal}, hk^s, hk^i, \pi, t^s, tid)$;
$count = 0$;
for $j \in [t]$ do
    $h' := Hash(hk^s, R)$;
    $h'' := Hash(hk^i, R)$;
    $x := (m, R, o, ct, ct_L, ct_{mal}, h', h'', hk^s, hk^i, tid)$;
    $b \leftarrow NIWI.Verify(x, \pi)$;
    $b \leftarrow b \wedge Verify_{sOTS}(pk_sOTS, \rho, \varsigma)$;
    if $b = 1 \wedge \sigma^j.o \neq \sigma^k.o\ \forall k \in [j-1]$) then
        $count++$;
    else
        return 0;
    end
end
if $count \geq t$ then
    return 1;
else
    return 0;
end
```

**Algorithm 4:** LTRS.Link($\sigma_1, \sigma_2$)

```
Let $t_i := |\sigma_i|, i \in \{1, 2\}$
for $(j,k) \in [t_1] \times [t_2]$ do
    if $\sigma_1^j.o_L = \sigma_2^k.o_L$ then
        return 1
    end
end
return 0
```

- Non-Slanderability: It ensures that no corrupt signer acting as adversary $A$ can succeed in forging a signature such that it is linkable with signature created by another honest endorser.
- Claimability and Repudiability: It allows a signer to claim a signature that he produced by revealing its identity. Repudiability allows the signer to prove that he did not generate a particular signature.

*Individual Ring Signature Size:* $|\sigma|$ is poly($\lambda$) and does not depend on the endorser set $|E|$. SPB.Verify has a circuit depth of log(n).poly($\lambda$) whereas the remaining algorithms in SPB are independent of n (endorser set) and are executed with circuits of depth poly($\lambda$). Hence, the signature size is log(n).poly($\lambda$).

## 3. Proposed Privacy-Preserving Endorsement System

In order to ensure privacy of endorsement system three attributes that leak information about endorsers have to be handled. They are endorser identity, endorser signature and the endorsement policy. An endorser identity is only a technical identity and does not directly imply true identity hence this can be discarded from the checklist. Every transaction response holds an endorser certificate and a signature. And every node in the network checks against the endorsement policy to validate a transaction. This reveals the organizations involved in the endorsement process. Therefore when it comes to preserving the privacy of endorsement system it is the *i) endorser signature and ii) endorsement policy* that should be secured.

### *3.1. Basic Idea*

The endorsement in a privacy-preserving endorsement system in Hyperledger Fabric contains two components. The *first component* is the endorser signature which is a ring signature that preserves the privacy of the signers. It is also a proof that it belongs to one of the organizations in the ring. The ring is a composition of verification keys of policy principals in the endorsement policy. The individual signatures by every endorser is aggregated, into a single threshold signature, by the client once threshold number of signers sign the transaction response. The validators receive a single threshold signature instead of multiple individual signatures. The fact that threshold signatures provide privacy to its signers, meaning that a t-out-of-n threshold signature does not tell the verifier which t of the n signers created the signature distinguishes it from multi-signatures. Multi-signatures, on the other hand enforce accountability as the verifier learns which are the signers who created the signature. Moreover, there is no threshold restrictions in a multi-signature whereas, in a threshold signature scheme the number of signers should be equal to or more than the specified threshold. The *second component* of an endorsement is a membership proof that the identity and secret key of the endorser used to produce a ring signature belongs to the committed organization in endorsement policy. This is because the endorsement policy is not available in clear to the validators. It is committed to ensure privacy of the participating organizations.

It is only the parties authorized to create an endorsement policy will have access to the policy in the clear. Parties authorized to invoke the chaincode will have access to the policy in committed form. That is, only organization pseudonyms will be available to the invoking parties instead of the actual organization identity. The efficiency of the verification process is improved by aggregating the membership proofs from every endorser into a single proof.
In the proposed work, every validating node receive a linkable threshold ring signature and a membership proof along with the transaction.

### *3.2. Fabric's Endorser Anonymization Technique: FEAT*

The first requirement to achieve privacy-preserving endorsement system is securing the endorser signature. As already stated, endorser signature on a transaction response reveals the organizations involved in a transaction. So a cryptographic tool is needed that preserves the privacy of the signer and at the same time allows the signer to prove that it came from a designated set. A ring signature exactly captures the aforesaid requirement. It allows a signer to hide its identity amongst a ring (or group) of potential signers. The advantage of employed signature scheme is that it has the property of claimability and repudiability which allows to track the misbehaving endorsers. The description of the scheme is given in algorithm 5.

**Algorithm 5:** FEAT - Fabric's Endorser Anonymization Technique

```
GenerateKeys($1^\lambda$):
$(VK, SK) \leftarrow FEAT.KeyGen(1^\lambda)$ : generates a
  verification key $VK$ and secret key $SK$ pair.
GenerateSignature($m, SK, R, t$):
$\sigma_s \leftarrow FEAT.Sign(m, SK, R, t, tid)$ : signs a transaction
  using the linkable threshold ring signature.
AssembleSignature($\sigma = \{\sigma_s\}_{s=1}^{t}$): Client receives $t$
  number of signatures stated by the endorsement
  policy and aggregates it into a single signature.
VerifySignature($m, R, \sigma, t$):
$b_1 \leftarrow FEAT.Verify(m, R, \sigma, t, tid)$ : Verifies if the
  single threshold ring signature is valid.
$b_2 \leftarrow FEAT.Link(\sigma_1, \sigma_2) : b = b_1 \wedge b_2$ : Verification
  passes only if all the previous verification function
  returns true.
```

**Generate Keys**
Every endorser generates a pair of keys: verification and a signing key $(VK, SK)$. Client initiates a transaction and forwards a transaction proposal to every endorser as stated by the endorsement policy.
**Generate Signature**:
Endorsers on receiving a proposal message execute the transaction and sign the transaction response using the threshold ring signature instead of a regular signature. Each endorser signs the message $m$ with respect to the ring $R$ for a transaction denoted by an unique identifier *tid*. Linkable property of signature prevents an endorser from signing more than once on the same transaction. The same endorser is allowed to sign different transactions remaining unlinkable. This implies that given two endorsements on different transactions it is infeasible to tell if they were made by same endorser or not. The endorsed transaction is then returned back to the client.
**Assemble Signature**
A client receives transaction response from $t$ number of endorsers. It aggregates the individual signatures into a single endorsement and appends it to the transaction and forwards it for validation.
**Verify Signature**
*Verification* involves checking the validity of individual signatures within the threshold signature and checking if sufficient number of endorsements are equal to the threshold $t$. Verifiers also check no endorsers have endorsed twice on the same transaction through the link function.

#### *3.2.1. Instantiation of Linkable Threshold Ring Signature Scheme*

Algorithms 6, 7, 8, and 9 are instantiations of the LTRS scheme described in the previous section. Let $\mathbb{G}$ be a bilinear group of prime order $p$ such that $|G| = p$ and $p$ is a $k$-bit prime and $\langle g \rangle$ be a generator of the group $\mathbb{G}$.

**Algorithm 6:** FEAT.KeyGen($1^\lambda$)

```
$(g^v, v) \leftarrow Gen_{VRF}(1^\lambda)$
$(g^c, c) \leftarrow Gen_{VRF}(1^\lambda)$
$(g^{v_L}, v_L) \leftarrow Gen_{VRF}(1^\lambda)$
$(g^{v_{mal}}, v_{mal}) \leftarrow Gen_{VRF}(1^\lambda)$
$(g^\alpha, \alpha) \leftarrow Gen_{PKE}(1^\lambda)$
$(g^\beta, \beta) \leftarrow Gen_{PKE}(1^\lambda)$
$sk_F \leftarrow \{0,1\}^{2\lambda}$
$\gamma_E \leftarrow_\$ \mathbb{Z}_q$
$CT \leftarrow Enc(g^\beta, sk_F; \gamma_E)$
$VK := (g^v, g^{v_L}, g^{v_{mal}}, g^\alpha, g^\beta, CT)$
$SK := (v, v_L, v_{mal}, \alpha, \beta, \gamma_E, VK)$
Result: return (VK,SK)
```

*Key Generation* The secret and public keys for VRF are calculated as in Dodis and Yampolskiy VRF scheme [30] as follows: $(v, v_L, v_{mal})$ are sampled from $\mathbb{Z}_p^*$ and the corresponding public keys are computed as $(g^v, g^{v_L}, g^{v_{mal}})$ respectively. Then the key pairs for public-key cryptosystem (PKE) are calculated with ElGamal encryption scheme as: $(g^\alpha, \alpha)$ and $(g^\beta, \beta)$.

*Signing.* Every endorser signs the transaction result with respect to a ring $R$. The ring is defined as the set of valid endorsers for a given endorsement policy. Signing starts by performing VRF evaluations with Dodis and Yampolskiy scheme [30]. Four pairs of VRF evaluations are generated: $(o, p)$; $(o', p')$; $(o_L, p_L)$; $(o_{mal}, p_{mal})$. First one is for the chaincode result $m$, second one is for the individual threshold $t^s$, next is for transaction id to achieve scoped linkability and final pair is to guarantee non-malleability of NIWI proof. The proofs of VRF evaluations are encrypted with the ElGamal encryption scheme. The individual threshold $t^s$ and aggregate threshold $t$ depend on the endorsement policy. The individual threshold value $t^s$

is 1 for the basic policy operators (AND, OR, and threshold). Whereas the $t^s$ value varies for each policy principal in a compound endorsement policy. A detailed description about setting the threshold values is given in the subsequent section. The final threshold $t$ is an aggregate of individual thresholds of all the endorsers. To achieve non-slanderability property of linkable threshold ring signature, non-malleability of NIWI proof needs to be guaranteed. Non-slanderability ensures no efficient adversary could generate a malicious signature such that it links to an honestly generated signature. Non-malleability of NIWI proof is achieved by the use of strong one-time signature. As mentioned above, only the values of VRFs are available and the proofs are encrypted and available as cipher texts $ct, ct', ct_L, ct_{mal}$ respectively. PKE.R represents the randomness used in the encryption scheme PKE.
Somewhere Perfectly Binding (SPB) hash function serves as a commitment to the signer's verification key which is binding at position $s$. For each position within the ring, the hash digest generated for the ring is unique. Combined with NIWI proofs, it allows a signer to prove that the signer's verification key is present in the ring $R$ at position $s$. The SPB hash possess index hiding property i.e., given the hashing key and hash digest it is computationally infeasible to tell apart to which index position the hashing key and the digest are generated. The instantiation of SPB hash function is adapted from [12]. Operations are performed over an abelian group $\mathbb{G}$ of prime order $p$. It is assumed that every group element is represented efficiently by $log(p) + c = \lambda + c$ bits, with $c$ being a constant. Certain elliptic curve groups and multiplicative groups of finite fields possess this property. With SPB hashing the ring $R$ of verification keys with cardinality $N$ is compressed into a ring with two verification keys $(VK_s, VK_i)$ which are binding at positions $s$ and $i$, respectively. NIWI proof is used to prove that the signature is valid under the committed verification key of the signer. A $2 - to - 1$ hash function which is an instantiation of SPB hash [12, 31] is employed. For this a hashing keys $hk^j$ for $j \in \{s, i\}$ that is computationally binding at index $s$ and $i$ are computed. Index $s$ holds the signer's verification key and index $i$ is chosen randomly within the ring $R$. The hashing key $hk$ is computed as follows.
$Gen_{SPB}(1^\lambda, N, indx \in \{0, 1\})$:

- Randomly choose $a$ and $b$ from $(\mathbb{Z}_p \backslash \{0\})^d$ and $w$ from $\mathbb{Z}_p^d$, where $d$ is the block size required to store the ring of verification keys.
- Compute $A^j \leftarrow w.a^T + (1 - indx).I$ and $B^j \leftarrow w.b^T + indx.I$, where $A$ and $B$ are matrices $A, B \in \mathbb{Z}_p^{m \times n}$ and $j \in \{s, i\}$.
- Compute $g_0 \leftarrow g^{a^T}, g_1 \leftarrow g^{b^T}, G_0^j \leftarrow g^A, G_1^j \leftarrow g^B$
- $hk^j \leftarrow (g_0, g_1, G_0^j, G_1^j)$

Based on the value of index $indx \in \{0, 1\}$, the hashing key is computationally binding to the appropriate verification key. By choosing $indx = 0$ the hashing key is generated for the verification key $VK_s$, and for $indx = 1$, the hashing key is generated for the verification key $VK_i$.
Next, hash digest is calculated as $h = Hash_{SPB}(hk, (VK_s, VK_i))$:

- $(hk^j, shk^j) \leftarrow (g_0, g_1, G_0^j, G_1^j; r_{SPB}^j)$ for $j = \{s, i\}$;
- $h^s \leftarrow (g_0^{VK_s}.g_1^{VK_i}, (G_0^s)^{VK_s}.(G_1^s)^{VK_i})$;
- $h^i \leftarrow (g_0^{VK_s}.g_1^{VK_i}, (G_0^i)^{VK_s}.(G_1^i)^{VK_i})$;

$G_0^s$ and $G_0^i$ are the values of $G_0$ for signer $s$ and random choice $i$. Similarly, $G_1^s$ and $G_1^i$ are the values of $G_1$ for signer $s$ and random choice $i$. The opening information $\tau^j$ is generated as $Open_{SPB}(hk^j, shk^j, R, j)$:

- $\tau^j \leftarrow (r_{SPB}^j, j, VK_j)$;

Verification keys $VK^s$ and $VK^i$ are committed to as $(hk^j, h^j)$ for $j \in \{s, i\}$. Opening information $\tau$ is the randomness, index and the verification key. Next, the signer calculates a NIWI proof for statement $x$ and witness $w$. Next, the partial signature $\rho$ is generated which is a composition of VRF evaluations: unique value $o$ for the message $m$, unique value $o'$ for the individual threshold $t^s$, unique value $o_L$ for the transaction id $tid$, $o_{mal}$ to achieve non-malleability; corresponding encryption of VRF proofs; hashing keys; NIWI proof and transaction id $tid$. To achieve non-malleability, the partial signature is signed with the one-time signature. Finally, the ring signature $\sigma^s$ is generated, that contains the partial signature, one-time signature on the partial signature and the verification key of one-time signature. At the end of the signing process, each endorser forwards its signature $\sigma^s$ to the client in the transaction response. Client now concatenates all the received signatures into a single threshold signature as: $\sigma = (\sigma^1, \sigma^2, ..., \sigma^t)$.

*Verification.* All the committing peers validate the endorser signatures with respect to the endorsement policy.
*Link.* Transaction-level linkability is a fine grained notion of linkability. Endorsements produced by same endorser on the same transaction is linkable, while endorsements by same endorsers on different transactions are valid. During verification, in addition to the verification procedure described in algorithm 8 the Link procedure is also invoked. Algorithm 9 depicts the description of the algorithm which performs a pairwise check. It compares the $o_L$ values of every signature within the threshold signatures. The algorithm returns Unlinked if every $o_L$ is unique and linked otherwise.

## Parameter Setting for Threshold Ring Signature

There are three basic variants of endorsement policies: i) AND(of members/signers) ii) OR(of signers) iii) Threshold (t-out-of-n signers) and a compound policy involving more than one basic operator (AND, OR). All three variants of endorsement policies can be implemented efficiently with the proposed signature scheme. The parameters $(t, n)$ are set by the client for the (t-out-of-n) linkable threshold ring signature depending on the endorsement policy as described below:
**AND variant:** (Org1.peer AND Org 2.peer AND Org3.peer), client sets $t = n = 3$ for the linkable threshold ring signature.

```
Algorithm 7: FEAT.Sign($m, SK, R, t, tid$))
  //Every signer s $\in S, |S| \geq t$
  $o \leftarrow e(g,g)^{1/((m\|R)+v)}$;
  $p \leftarrow g^{1/((m\|R)+v)}$;
  $o' \leftarrow e(g,g)^{1/((t^s\|m\|R)+v)}$;
  $p' \leftarrow g^{1/((t^s\|m\|R)+v)}$;
  $o_L \leftarrow e(g,g)^{1/(tid+v_L)}$;
  $p_L \leftarrow g^{1/(tid+v_L)}$;
  $(vk_{sOTS}, sk_{sOTS}) \leftarrow Gen_{sOTS}(1^\lambda)$;
  $o_{mal} \leftarrow e(g,g)^{1/(vk_{sOTS}+v_{mal})}$;
  $p_{mal} \leftarrow g^{1/(vk_{sOTS}+v_{mal})}$;
  $\gamma_{ct}, \gamma_L, \gamma_{mal} \leftarrow_\$ \mathbb{Z}_q$;
  $ct \leftarrow p \times g^{\alpha.\gamma_{ct}}$;
  $ct_L \leftarrow p_L \times g^{\alpha.\gamma_L}$;
  $ct_{mal} \leftarrow p_{mal} \times g^{\alpha.\gamma_{mal}}$;
  $(hk^s, shk^s) \leftarrow Gen_{SPB}(1^\lambda, N, s)$;
  $h^s \leftarrow Hash_{SPB}(hk^s, R)$;
  $\tau^s \leftarrow Open_{SPB}(hk^s, shk^s, R, s)$;
  //Pick other ring member $i \neq s$
  $i \leftarrow [N] \backslash s$;
  $r_{E_0}, r_{E_1} \leftarrow_\$ \mathbb{Z}_q$;
  $(hk^i, shk^i) \leftarrow Gen_{SPB}(1^\lambda, N, i)$;
  $h^i \leftarrow Hash_{SPB}(hk^i, R)$;
  $\tau^i \leftarrow Open_{SPB}(hk^i, shk^i, R, i)$;
  //Call on the NIWI
  $x = (m, R, o, o_L, o_{mal}, h^s, h^i, hk^s, hk^i, vk_{sOTS},$
        $ct, ct_c, ct_{mal}, tid, ct_L, \varsigma_{sOTS})$
  $w = (\tau^s, \tau^i, p, p_{mal}, p_L, sk_F^{i_0}, sk_F^{i_1},$
        $r_{E_0}, r_{E_1}, \gamma_{ct}, \gamma_L, \gamma_{mal})$
  $\pi \leftarrow NIWI.Prove(x, w)$;
  $\rho := (o, o', o_L, o_{mal}, ct, ct_L, ct_{mal}, hk^s, hk^i, \pi, tid)$;
  $\varsigma \leftarrow Sign_{sOTS}(sk_{sOTS}, \rho)$;
  $\sigma^s := (\rho, \varsigma, pk_{sOTS})$;
  //Final threshold ring signature constructed by the
   client
  Result: return $\sigma = \{\sigma^j\}_{j=1}^t$.
```

```
Algorithm 8: FEAT.Verify($m, R, \sigma, t, tid$)
  //Parse signature
  $\sigma = (\rho, \varsigma, pk_{sOTS})_{j=1}^t$;
  $\rho := (o, o', o_L, o_{mal}, ct, ct_L, ct_{mal}, hk^s, hk^i, \pi, tid^j)$;
  $count = 0$;
  for $j \in [t]$ do
    $h' := Hash(hk^s, R)$;
    $h'' := Hash(hk^i, R)$;
    $x = (m, R, o, o_L, o_{mal}, h^s, h^i, hk^s, hk^i$
          $vk_{sOTS}, ct, ct_{mal}, tid, ct_L, \varsigma_{sOTS})$
    $b \leftarrow NIWI.Verify(x, \pi)$;
    $b \leftarrow b \wedge Verify_{sOTS}(pk_{sOTS}, \rho, \varsigma)$;
    if $b = 1 \wedge \sigma^j.o \neq \sigma^k.o \; \forall k \in [j-1]$) then
      $count++$;
    else
      return 0;
    end
  end
  if $count \geq t$ then
    return 1;
  else
    return 0;
  end
```

```
Algorithm 9: FEAT.Link($\sigma_1, \sigma_2$)
  Let $t_i := |\sigma_i|, i \in \{1,2\}$
  for (j,k) $\in [t_1] \times [t_2]$ do
    if $\sigma_1^j.o_L = \sigma_2^k.o_L$ then
      return 1
    end
  end
  return 0
```

**OR variant:** (Org1.peer OR Org 2.peer OR Org3.peer), client sets $t = 1$ and $n = 3$ for the linkable threshold ring signature.
**Threshold Variants:** can be expressed using the basic logical operators as shown below:

– OutOf ( 2, 'Org1.peer', 'Org2.peer', 'Org3.peer' ), client sets $t = 2$ and $n = 3$ for the linkable threshold ring signature.

– OutOf ( 1, 'Org1.admin', 'Org2.admin' ) ⇒ OR ( 'Org1.admin', 'Org2.admin' ).

– OutOf ( 2, 'Org1.peer', 'Org2.peer' ) ⇒ AND ( 'Org1.peer', 'Org2.peer' ).

**Compound variant:**
OR('Org1.peer', AND('Org2.peer', 'Org3.peer')): Compound variant is the one where both the basic operators appear in the endorsement policy. Individual thresholds are useful for compound variant endorsement policy and are assigned by the clients. Total threshold $t$ is then the aggregate of individual thresholds. For the example given above, total threshold $t$ is set to 2 and $n$ is 3. Individual thresholds are assigned as $t^2$ for the policy principal Org1.peer and $t^1$ for both Org2.peer and Org3.peer. Aggregation is done as follows.

$$OR('Org1.peer', AND('Org2.peer',' Org3.peer')) \quad (1)$$

Substituting the individual thresholds in equation 1,

$$\begin{aligned}
&OR(t^2, AND(t^1, t^1)) \\
&OR(t^2, (t^1 * t^1)) && (\because AND \leftarrow *) \\
&OR(t^2, t^2) && (\because t^1 * t^1 => t^{1+1} => t^2) \quad (2) \\
&(t^2 + t^2) && (\because OR \leftarrow +) \\
&t^2 && (\because t^2 + t^2 => 2.t^2 => t^2)
\end{aligned}$$

Similarly for AND('Org1.peer', OR('Org2.peer', 'Org3.peer')): total threshold $t = 2$ and $n = 3$. Individual thresholds are assigned as $t^1$ for all three policy principals Org1.peer, Org2.peer and Org3.peer. Aggregate of individual thresholds are calculated as before.
$AND(t^1, OR(t^1, t^1)) => (t^1.(t^1 + t^1)) => (t^1.t^1) => t^2$.
The proposed privacy-preserving endorsement system can be used to implement chaincode or collection or key-level secure endorsement policy where the endorsers produce linkable threshold ring signature, pseudonymous organization identity and zero-knowledge membership proof using the technique described in Section 4.2.

### *3.3. Privacy-Preserving Endorsement Policy*

Endorsement policy is an expression over policy principals [5]. Policy principals are represented as `MSP.ROLE`, where MSP denotes the organization and ROLE denotes one of the four possible roles: (member,admin,client and peer). For example, a valid principal is of the form `Org1.peer`. Securing endorsement policy implies securing the organizations involved in the endorsement process.

#### *3.3.1. Our Approach*

Consider an endorsement policy is of the form $\{Org1.peer$ $AND\ Org2.peer\}$. This implies that an endorsing peer from Org1 and one endorsing peer from Org2 has to execute and sign the result of the transaction. As the endorsement policy clearly leaks the organizations involved in executing and signing the transactions, there are possible attacks like DDoS against the participating organizations. We propose a solution using cryptographic techniques to secure the endorsement policy.
The idea is as follows: Every policy creator first generates a commitment to the organization identities in the endorsement policy using Pedersen commitment scheme. Then it shares the randomness to generate the commitment to the respective endorsing peers of each organization through secure communication channels. Hyperledger Fabric allows for configuring secure communication using TLS. On receiving the commitment to their organization and the respective randomness, endorsers verify the validity of the commitment. Finally, endorsers create a proof to prove that they are the designated endorsers for the committed organization. The proof requires the knowledge of randomness to generate commitment, identity, credential on the and secret key of the endorser. A zero-knowledge proof of knowledge can be generated using Schnorr protocol [32] and applying Fiat-Shamir heuristic makes the protocol non-interactive. The classical Schnorr protocol suffers from a limitation that it has soundness error bounded by 1/2 under the standard model of security [33]. This soundness limitation is overcome in [34], by restricting the adversary to generic group model for groups of unknown order [35]. The proposed scheme employs a succinct non-interactive zero-knowledge argument of integer vector by Boneh et.al., [34]. Individual membership proofs are then aggregated using a proof of knowledge of co-prime roots technique to make the verification process simpler.

#### *3.3.2. Our Technique*

The proposed technique is described as follows.
**Generate Commitment to Organization Identity** Algorithm 10 depicts the generation of organization pseudonym through commitment to the organization identity.

**Algorithm 10:** Generate Commitment to Organization Identity

```
Setup($\lambda$): $\{\mathbb{G}, g, h \leftarrow NI-ZKPoKRep.Setup(\lambda)\}$
GenOrgPseudonym($Orgid$): Calculate $C \leftarrow g^{Orgid}.h^d$.
  Endorsement policy creators generate commitments
  to the organization identities in the endorsement
  policy. And, forward the organization pseudonym and
  the associated randomness through a secure channel.
VerifyOrgPseudonym($C, d$): The endorsers in the
  policy check the validity of the commitment as
  $C \overset{?}{=} g^{Orgid}.h^d$.
```

The parameters needed for the protocol are generated in the *Setup* phase by the committing peers (or verifiers) in the network. For a chaincode-based endorsement policy, a default

policy is set along with the chaincode. For state-based policy, policy consists of the state owners or subset of state owners. Any organization that owns a ownable state is said to be state-owners. An entity that creates the endorsement policy is called a policy creator. It is the policy creators which are aware of the organizations involved in the endorsement policy. Hence the pseudonyms are created for the organizations by the policy creators using the function *GenOrgPseudonym(Orgid)* as $C \leftarrow g^{Orgid}.h^d$ and forwarded to the endorsers. The endorsing peers receive the commitment to their organization $C$ and the randomness $d$ for generating the commitment. It then checks if the received commitment $C = g^{Orgid}.h^d$.
Given an endorsement policy, (Org1.peer AND Org2.peer) it will be expressed as pseudonyms of the organization identity as ($C_1.peer$ AND $C_2.peer$). Validating peers will now only possess this pseudonymous identity and hence the organization identity is no more leaked through endorsement policy.

### Issue Credential

Every endorser before generating a membership proof requests the certificate authority to issue a credential. The certificate authority provides a certificate which serves as a link between the endorser signing key and the organization identity. Algorithm 11 depicts the issue credential procedure.

```
Algorithm 11: Issue Credential
  Request($id, VK, Orgid$): Endorser send a certificate
    request to the certificate authority. The request
    consists of endorser's identity $id$, verification key $VK$,
    and the organization identity.
  Issue($VK, Orgid$): Certificate authority validates the
    received information and issues a credential
    $\sigma_c \leftarrow Sign(sk_{CA}, m = (VK, Orgid))$. It signs on the
    pair: $(VK, Orgid)$ verification key and organization
    identity of the endorser to bind the endorser's signing
    key to the respective organization.
```

### Generate Proof of Membership

The peer then generates a non-interactive zero-knowledge proof of knowledge of integer vector that it is indeed an authorized entity to perform the endorsement process based on the NI-ZKPoKRep. To prove that an endorser is a member of an organization it needs to prove knowledge of the secret randomness $d$, its secret key $SK$, the technical identity *id* that describes the role within the organization, and the credential $\sigma_c$ issued by CA to link the endorser's secret key with the organization identity. The description of NI-ZKPoKRep is self-contained in the algorithm 12 and a separate description is also given in Appendix A.5. Prover chooses bases $g_i \in \mathbb{G}$ such that $g_i^{w_i}$ is also a group element for all $i \in [1, 4]$. The witness $w_1$ is the randomness $d$, to create the pseudonymous organization identity, witness $w_2$ is the secret key $SK$ of the endorser, witness $w_3$ is the endorser id *id* and $w_4$ is a signature on the verification key and organization identity of the endorser by the certification authority. The statement $x$ is the product of $g_i^{w_i}$. Every endorser invokes the $NI-ZKPoKRep.Prove(x, w_i)$ function to create a membership proof that its identity is bound to the corresponding pseudonymous organization identity. To construct a zero-knowledge proof, the prover first randomly samples four blinding factors $k_1, k_2, k_3, k_4$ from the range of integers $[-B, B]$ such that $|\mathbb{G}|/B$ should be negligible. The prime $l$ is computed by hashing the values $(g, x, A)$ to prime and the challenge $c$ is the hash of the computed prime $l$. By the division algorithm, any integer (say, $s$) and any positive integer (say, $l$) there exists unique integers $q$ and $r$ such that $s = l.q + r$, where $r \in \{0, ..., l-1\}$. $s_i$ is expressed as stated by the division algorithm and eventually the corresponding quotients and residues are derived. Finally proof $\pi$ is constructed as $\pi \leftarrow \{A, Q, r_i\}$. Policy creator validates the proof through $NI-ZKPoKRep.Verify(x, \pi)$ and accepts or rejects based on the correctness of the proof. Verifiers parses the proof $\pi = \{A, Q, r_i\}$. Computes $l$ and challenge $c$ as hash of prime $l$. It then checks if the residues $r_i$ are in the range $\{0, 1, ..., l-1\}$. It then checks if the verification equation $Q^l \prod_{i=1}^{4} g_i^{r_i} \overset{?}{=} Ax^c$ is satisfied. Equation 3 shows the correctness of the verification equation.

$$\begin{aligned} Q^l . \prod_{i=1}^{4} g_i^{r_i} &= \prod_{i=1}^{4} (g_i^{q_i})^l g_i^{r_i} \\ &= g_i^{q_i.l+r_i} \\ &= g_i^{s_i} \\ &= g_i^{k_i+c.w_i} \\ &= g^{k_i}.g_i^{w_i.c} \\ &= A.x^c \end{aligned} \tag{3}$$

### Aggregate Membership Proofs

Aggregating $t$ membership proofs into a single constant size proof is advantageous to verifiers. This is because, an aggregated proof can be verified faster than verifying $t$ different membership proofs individually.
The membership proof illustrates that the generated threshold ring signature is valid with respect to the commitments to the organization.
A succinct proof of knowledge of co-prime roots is used to aggregate the membership proofs as shown in algorithm 13.

One of the policy creators after receiving the individual membership proofs from the policy principals invokes PoKCR.AggProof() to aggregate them into a single batched proof by computing the product of the individual proofs. Pseudo-code for PoKCR is self-contained in the algorithms and a complete description can be found in Appendix A.6. Parameter $Q$ sent to the verifier in the NI-ZKPoKRep is the proof $\pi$ for an instance of PoKCR. $Q$ is the $l$th root of $\alpha$. Verifiers recompute the root $l$ which is the product of individual $l_i$'s and $y$ from the NI-ZKPoKRep proofs. It accepts if the validation $\pi^l \overset{?}{=} y$ passes. The verification process runs in $O(nlogn)$ time where $n$ is the number of proofs to be aggregated. The verification equations for $\pi^l \overset{?}{=} y$ is given in

```
Algorithm 12: Generate Proof of Membership
  ProveOrgMem($x, w_1 = d, w_2 = SK, w_3 = id, w_4 = \sigma_c$): Every policy principal computes a membership
    proof as: $\pi \leftarrow NI-ZKPoKRep.Prove(g_i, x \in \mathbb{G}; \mathbf{w} \in \mathbb{Z}^4 : x = \prod_i g_i^{w_i} \ \forall i \in [1,4])$
  Proof is calculated with NI-ZKPoKRep as follows:
      – Prover randomly samples $k_1, k_2, k_3, k_4 \leftarrow_{\$} [-B, B]$,
        where $B > 2^{2\lambda}|\mathbb{G}|$ and computes $A = \prod_{i=1}^{4} g_i^{k_i}$.
      – Computes $l \leftarrow H_{prime}(g_i, x, A)$ and $c \leftarrow H(l)$.
      – Computes $s_i = k_i + c.w_i \ \ \forall i \in [1,4]$.
      – Derives quotients $\mathbf{q} \in \mathbb{Z}^4$ and residues $\mathbf{r} \in [l]^4$
        such that $q_i.l + r_i = s_i \ \ \forall i \in [1,4]$.
      – Computes $Q = \prod_{i=1}^{4} g_i^{q_i}$.
      – Constructs the proof as $\pi \leftarrow \{A, Q, \mathbf{r}\}$.
  VerifyOrgMem($x, \pi$): Policy creator verifies the proof
    from every policy principal by computing
    $b \leftarrow NI-ZKPoKRep.Verify(x, \pi)$ and accepts the
    proof if the verification passes.
      – Parse proof $\pi$ as $\{A, Q, \mathbf{r}\}$
      – Computes $l \leftarrow H_{prime}(g_i, x, A)$ and $c \leftarrow H(l)$
      – Check if: $r_i \in \{0, ..., l-1\} \ \ \forall i \in [1,4]$;
        $Q^l \prod_{i=1}^{4} g_i^{r_i} \overset{?}{=} Ax^c$.
```

equation 4 as follows:

$$\begin{aligned}\pi^l &= \prod_{i=1}^{n} Q_i^l \\ &= \prod_{i=1}^{n} (\alpha^{1/l_i})^l \\ &= y\end{aligned} \tag{4}$$

The endorsement policy will be presented as pseudonymous organization identity with existing policy operators thus concealing the participating organizations. The membership proofs are verified by the policy creators and aggregated. The committing nodes in the Fabric network verify the aggregated proofs. Individual proofs can also be extracted out from the aggregated proof by the validating peers. An aggregated proof allows a verifier to verify a single proof instead of verifying it separately for every endorser. The final endorsement will now consist of a single threshold signature and an aggregated membership proof that the signers belong to the committed organization.

## 4. Threat Model and Security Model

### 4.1. Threat Model

The framework for threat model is adapted from [4] which captures the power of adversary. As in any permissioned blockchain

```
Algorithm 13: Aggregate Membership Proofs
  AggregateProof($\pi_i \forall i \in [1,n]$): Policy creator invokes
    PoKCR.AggProof($\alpha_i, l_1, ..., l_n \in \mathbb{Z}, \forall i \in [1,n]$).
      – $\pi_i := (Q_i)$
      – $Q_i := (\alpha)^{1/l}$ where, $\alpha = (A \prod_{i=1}^{3} (g_i^{r_i})^{-1} x^c)$.
      – Computes a single aggregated proof as the
        product of all the received membership proofs:
        $\pi \leftarrow \prod_{i=1}^{n} Q_i$.
  VerifyAggregateProof($\pi$): Every committing peer
    invokes PoKCR.VerifyAggProof($n, \alpha^i, l$)
      – Computes $l \leftarrow \prod_{i=1}^{n} l_i$ and $y \leftarrow \prod_{i=1}^{n} (\alpha_i^{l/l_i})$.
      – Accepts the proof, if $\pi^l \overset{?}{=} y$.
```

network, Hyperledger Fabric allows the ledger to be accessible only to the members of the network. Channels in Hyperledger Fabric allows only a subset of the members to access the respective ledger. Assume an adversary $A$ to be a member of a channel in Hyperledger Fabric. Following are the assumptions with respect to adversary $A$:

- $A$ can see all the network traffic within the channel.
- It is possible for an adversary to change an unconfirmed transaction sent from client to endorser during transaction proposal phase.
- It can endorse (provided it is a policy principal) and/ or validate the transactions within a channel.
- Any adversary $A$ acting as a validator may hold back the block of transactions without verifying its validity.
- Any adversary $A$ acting as a validator may incorrectly mark a transaction as valid or invalid without honestly running the validation process.

### 4.2. Security Model for Privacy-preserving Endorsement System

The security goals for the anonymity and unlinkability of endorsement system are adapted from [5].

#### 4.2.1. Endorsement Policy Anonymity

This security model captures the idea that the real identity of the policy principals expressed in the endorsement policy is concealed. It is defined using a game that is played between a challenger and a *PPT* adversary $A$. The game proceeds as follows.

- The challenger runs $GenerateKeys(1^{\lambda})$ algorithm and gives the ring of verification keys $R$ to the adversary $A$.
- Adversary $A$ chooses a pair of equal length policy principal identities $id_0, id_1$ and sends these identities to the challenger.

- The challenger picks a random bit $b \in \{0, 1\}$ and calculates the challenge commitment $c_b = Commit(id_b, r)$, and sends $c_b$ to the adversary $A$.
- The adversary outputs a guess $b' \in \{0, 1\}$, it succeeds if $b' = b$.

We define the advantage of the adversary $A$ in attacking the endorsement policy anonymity of endorsement system $\xi$ as $Adv_{\xi,A}(\lambda) = |Pr[b = b'] - 1/2|$.

**Definition 4.1.** We say that the endorsement policy is anonymous if for all $PPT$ adversaries $A$ we have $Adv_{\xi,A}(\lambda)$ is a negligible function.

### *4.2.2. Policy Principal Unlinkability*

This property demands that when the same policy principal is used in two different endorsement policies the principals should remain unlinkable. It is defined using a game that is played between a challenger and a $PPT$ adversary $A$. The game proceeds as follows.

- The challenger runs $GenerateKeys(1^{\lambda})$ algorithm and gives the ring of verification keys $R$ to the adversary $A$.
- Adversary $A$ chooses an identity $id$ and sends it to the challenger.
- The challenger picks a random bit $b \in \{0, 1\}$ and for $b = 0$ calculates commitments $c_0, c_1$ with $id$ else calculates commitment $c_0$ with $id$ and $c_1$ with random and sends $c_0, c_1$ to the adversary $A$.
- The adversary outputs a guess $b' \in \{0, 1\}$, it succeeds if $b' = b$.

We define the advantage of the adversary $A$ in attacking the policy principal unlinkability of endorsement system $\xi$ as $Adv_{\xi,A}(\lambda) = |Pr[b = b'] - 1/2|$.

**Definition 4.2.** We say that the policy principals in the endorsement policy is unlinkable if for all $PPT$ adversaries $A$ we have $Adv_{\xi,A}(\lambda)$ is a negligible function.

### *4.2.3. Endorser Anonymity*

This security model captures the idea that it should be infeasible for an adversary to trace back the true identity of the endorser from the endorser signature. It is defined using a game that is played between a challenger and a $PPT$ adversary $A$. The game proceeds as follows.

- The challenger runs $GenerateKeys(1^{\lambda})$ algorithm and gives the ring of verification keys $R$ to the adversary $A$.
- Adversary $A$ chooses a pair of equal length endorser identities $id_0, id_1$ and sends these identities to the challenger.
- The challenger picks a random bit $b \in \{0, 1\}$ and generates the challenge endorsement $\sigma_b = GenerateSignature(m, SK, R, t)$. Challenge endorsement is given to adversary $A$.
- The adversary outputs a guess $b' \in \{0, 1\}$, it succeeds if $b' = b$.

We define the advantage of the adversary $A$ in attacking the anonymity of endorsement system $\xi$ as $Adv_{\xi,A}(\lambda) = |Pr[b = b'] - 1/2|$.

**Definition 4.3.** We say that the endorsers in endorsement system are anonymous if for all $PPT$ adversaries $A$ we have $Adv_{\xi,A}(\lambda)$ is a negligible function.

### *4.2.4. Endorser Unlinkability*

This property requires that when an endorser signs/endorses two different transactions, it should be infeasible to track that the endorsements were made by the same endorser. It is an extension to the endorser anonymity property. It is defined using a game that is played between a challenger and a $PPT$ adversary $A$. The game proceeds as follows.

- The challenger runs $GenerateKeys(1^{\lambda})$ algorithm and gives the ring of verification keys $R$ to the adversary $A$.
- Adversary $A$ chooses an endorser identity $id$ and sends it to the challenger.
- The challenger picks a random bit $b \in \{0, 1\}$ and generates two challenge endorsements $\sigma_0$ and $\sigma_1$. If $b = 0$, both $\sigma_0 = GenerateSignature(m_0, SK_0, R, t)$ and $\sigma_1 = GenerateSignature(m_1, SK_0, R, t)$ are generated for the same $id$ itself else the endorsements are generated separately as one for $id$ and another for some random. Then, the endorsements are given to adversary $A$.
- The adversary outputs a guess $b' \in \{0, 1\}$, it succeeds if $b' = b$.

We define the advantage of the adversary $A$ in attacking the unlinkability of endorsement system $\xi$ as $Adv_{\xi,A}(\lambda) = |Pr[b = b'] - 1/2|$.

**Definition 4.4.** We say that the endorsers in endorsement system are unlinkable if for all $PPT$ adversaries $A$ we have $Adv_{\xi,A}(\lambda)$ which is a negligible function.

## 5. Security Analysis of privacy-preserving endorsement system

The work assumes the following corresponding to the Hyperledger Fabric framework: Fabric's Certificate Authority (CA) is assumed to be honest; more than half of the peer nodes and endorsers are honest for the smooth execution of endorsement system. In the following we prove that the security goals are achieved by giving proofs by reduction. The security proofs are based on [36].

### 5.1. Endorsement Policy Anonymity and Unlinkability

**Theorem 5.1.** Our endorsement system provides anonymity and unlinkability to the endorsement policy.

*Proof.* We will show that an adversary $A$ can reveal the endorsement policy in our endorsement scheme $\xi$ with non-negligible probability then there exist another adversary $A'$ which can break the underlying commitment scheme and zero-knowledge proof with non-negligible probability.

1. Adversary $A$ chooses two organization identities $Orgid_0$ and $Orgid_1$ and forwards it to adversary $A'$.
2. $A'$ assigns $m_0 = Orgid_0$ and $m_1 = Orgid_1$ and forwards the messages $\{m_0, m_1\}$ to be committed to challenger $C$.
3. Challenger chooses a bit $b \in \{0,1\}$ and computes the challenge message $c_b = Commit(m_b; r)$ and a zero-knowledge proof of its correctness as $\pi_b = NI{-}ZKPoKE.Prove(x = c_b, w = r)$ and forwards the challenge (message, proof) pair $(c_b, \pi_b)$ to adversary $A'$.
4. Adversary $A'$ forwards the pair to $A$.
5. Adversary $A$ outputs a guess $b'$ to $A$ which is then forwarded to the challenger $C$.

Adversary $A$ can compute the policy principal $id$ $Orgid$ corresponding to the committed policy principals $c_{id}$ with probability significantly better than $\frac{1}{2}$. This is given by:

$$Pr[A(c_{id}) = Orgid_i] - \frac{1}{2} > non - neglgbl(\lambda) \tag{5}$$

The probability of $A$ breaking the endorsement policy anonymity and unlinkability is exactly equal to the probability of $A'$ breaking the commitment scheme and zero-knowledge proof system given by equation 5. By the unconditional hiding property of commitment scheme and statistical zero-knowledge property of the NIZK scheme it is computationally infeasible to find the policy principal and link a policy principal across different endorsement policies. Since underlying schemes are secure so is our scheme. □

### 5.2. Endorser Anonymity and Unlinkability

**Theorem 5.2.** Our endorsement system provides anonymity and unlinkability to the endorser across different endorsements.

*Proof.* We will show that an adversary $A$ can reveal endorser in our endorsement scheme $\xi$ with non-negligible probability then there exist another adversary $A'$ which can break underlying linkable threshold ring signature with non-negligible probability.

1. Adversary $A$ chooses two endorser identities $eid_0$ and $eid_1$ and forwards it to adversary $A'$.
2. $A'$ forwards the identities $\{eid_0, eid_1\}$ to the challenger $C$.
3. Challenger chooses a bit $b \in \{0,1\}$ and computes the challenge signature $\sigma_b = LTRS.Sign$(m,SK,R,t,tid) and forwards the challenge signature $\sigma_b$ to adversary $A'$.
4. Adversary $A'$ forwards it to $A$.
5. Adversary $A$ outputs a guess $b'$ to $A$ which is then forwarded to the challenger $C$.

Adversary $A$ can compute the endorser $id$ corresponding to the endorsement $\sigma$ with probability significantly better than $\frac{1}{2}$. This is given by:

$$Pr[A(\sigma_i) = id_i] - \frac{1}{2} > non - neglgbl(\lambda) \tag{6}$$

The probability of $A$ breaking the endorser anonymity is exactly equal to the probability of $A'$ breaking the linkable threshold ring signature scheme given by equation 6. By the linkable anonymity property of linkable threshold ring signature it is computationally infeasible to find the actual signer and link an honest signer across different transactions. Since underlying linkable threshold ring signature is a secure scheme so is our scheme. □

**Theorem 1.** If the linkable threshold ring signature scheme possesses unforgeablity, linkable anonymity, transaction-level linkability and non-slanderability then the proposed privacy-preserving endorsement system is also secure under the standard model of security.
The detailed security proofs can be found in [25].

## 6. Implementation

Hyperledger Fabric is implemented in `Ubuntu 16.04 64 bit OS`, `Intel Core I5 quad core processor`. Fabric and the chaincode use Go 1.12 [37]. Secure communication is established between the nodes through `TLS` (Transport Layer Security). TLS is configured on peer nodes by setting the environment variable in peer/tls/enabled = true and providing a fully qualified path for server certificate, server private key and CA chain file. TLS was similarly configured at the clients as well by setting the appropriate environment variables. To integrate the privacy-preserving endorsement system in Fabric a new implementation of the Membership Provider is needed. The MSP implementation for the proposed work is given below:

### 6.1. MSP Implementation with Privacy-Preserving Endorsement System

MSP Implementation of the privacy-preserving endorsement system is supported for all the roles in the Fabric network:

- $(sk_{CA}, pk_{CA}, params) \leftarrow Setup(1^\lambda)$, the algorithm takes as input the security parameter $\lambda$ and outputs the public and private key pair for the Certificate Authority (CA).
- $(sk_U, pk_U) \leftarrow Gen(params)$, the algorithm takes as input the system parameters and outputs the public and private keys for the users.
- $id \leftarrow Enroll\{CA(sk_{CA}, pk_{CA}) \leftrightarrow U(sk_U, pk_U)\}$, The algorithm generates an enrollment identifier for the users based on their role - (member,client,peer,admin) within the organization. Enroll is a two party protocol between the $CA$ and the user $U$. $CA$ provides as input its key-pair and the user $U$ provides its key-pair along with an identification information. Finally, the $CA$ and user $U$ gets an identity $id$ that matches the secret key $sk_U$.

- $CRL' \leftarrow Revoke(CA(sk_{CA}, CRL, id))$, the algorithm allows a CA to revoke a particular identity *id* from the network and generate an updated certificate revocation list of revoked users.

- $\sigma \leftarrow FEAT.GenerateSignature(m, SK, R, t, tid)$ generates a t-out-of-n threshold ring signature with the secret key $SK$ for a message $m$ of a transaction *tid*, and for a designated set of verification keys from the endorser set, where $E$. By setting threshold $t = 1$, linkable threshold ring signature becomes a linkable ring signature scheme.

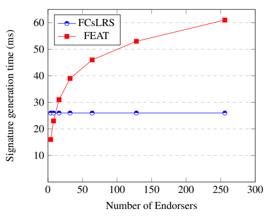
- $0/1 \leftarrow FEAT.VerifySignature(m, R, \sigma, t, tid)$, the algorithm on a single shot validates the threshold signature with only information that the signature came from a designated set and nothing beyond that.

## 7. Performance Analysis

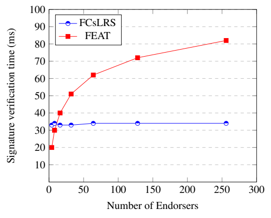
Table 1 compares FCsLRS, which is the previous work on endorser anonymization, with the proposed work based on signature size and security model. It is to be noted that the comparison is made for the endorsers' individual signatures. It is clear from the analysis that FCsLRS scheme has a constant signature size and hence less signature generation and verification times whereas secure only under random oracle model. But the proposed FEAT scheme is a logarithmic function of the number of signers which is still efficient. Moreover the existing scheme does not consider securing the endorsement policy. Hence, comparing only the signature schemes the proposed scheme is secure under standard assumptions. While considering the complete FEAT framework (securing endorser signature and endorsement policy), the proposed scheme is secure under generic group model of security.

Table 1: Comparative Analysis

| Performance Metric | FCsLRS [27] | FEAT |
|---|---|---|
| Signature Size | $O(1)$ | $O(log(l))$ |
| Security Assumption | ROM | Standard Model |



(a) Signature generation time



(b) Signature verification time

Figure 2: Signature generation and verification time w.r.t number of endorsers

Figure 2 shows a comparative analysis in terms of signature generation and verification time between the existing FCsLRS scheme and the proposed FEAT scheme for $\lambda = 1024$ bits.

## 8. Conclusion and Future Work

The proposed privacy-preserving endorsement system for Hyperledger Fabric conceals both the endorser identity and the endorsement policy. The privacy-preserving endorsement system achieves the security requirements of anonymity and unlinkability among endorsers and endorsement policy.
In this work, the employed threshold ring signature scheme is secure under the standard security assumptions but lacks security against quantum attacks. A possible future work would be to design a secure endorsement system which has efficient threshold ring signature that is post-quantum secure.

## References

[1] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., Hyperledger fabric: a distributed operating system for permissioned blockchains, in: Proceedings of the Thirteenth EuroSys Conference, 2018, pp. 1–15.
[2] J. Camenisch, A. Lysyanskaya, Signature schemes and anonymous credentials from bilinear maps, in: Annual International Cryptology Conference, Springer, 2004, pp. 56–72.
[3] Current limitations of idemix (mar 2020).
URL `https://hyperledger-fabric.readthedocs.io/en/release-2.0/idemix.html`
[4] N. Andola, M. Gogoi, S. Venkatesan, S. Verma, et al., Vulnerabilities on hyperledger fabric, Pervasive and Mobile Computing 59 (2019) 101050.
[5] E. Androulaki, A. De Caro, M. Neugschwandtner, A. Sorniotti, Endorsement in hyperledger fabric, in: 2019 IEEE International Conference on Blockchain (Blockchain), IEEE, 2019, pp. 510–519.
[6] R. L. Rivest, A. Shamir, Y. Tauman, How to leak a secret, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2001, pp. 552–565.
[7] P. P. Tsang, V. K. Wei, Short linkable ring signatures for e-voting, e-cash and attestation, in: International Conference on Information Security Practice and Experience, Springer, 2005, pp. 48–60.
[8] M. H. Au, S. S. Chow, W. Susilo, P. P. Tsang, Short linkable ring signatures revisited, in: European Public Key Infrastructure Workshop, Springer, 2006, pp. 101–115.
[9] S. S. Chow, V. K. Wei, J. K. Liu, T. H. Yuen, Ring signatures without random oracles, in: Proceedings of the 2006 ACM Symposium on Information, computer and communications security, 2006, pp. 297–302.
[10] A. Bender, J. Katz, R. Morselli, Ring signatures: Stronger definitions, and constructions without random oracles, in: Theory of Cryptography Conference, Springer, 2006, pp. 60–79.
[11] J. K. Liu, V. K. Wei, D. S. Wong, Linkable spontaneous anonymous group signature for ad hoc groups, in: Australasian Conference on Information Security and Privacy, Springer, 2004, pp. 325–335.
[12] M. Backes, N. Döttling, L. Hanzlik, K. Kluczniak, J. Schneider, Ring signatures: Logarithmic-size, no setup—from standard assumptions, in: Annual International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2019, pp. 281–311.
[13] E. Bresson, J. Stern, M. Szydlo, Threshold ring signatures and applications to ad-hoc groups, in: Annual International Cryptology Conference, Springer, 2002, pp. 465–480.
[14] M. Abe, M. Ohkubo, K. Suzuki, 1-out-of-n signatures from a variety of keys, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2002, pp. 415–432.
[15] Y.-F. Chang, C.-C. Chang, P.-Y. Lin, A concealed t-out-of-n signer ambiguous signature scheme with variety of keys, Informatica 18 (4) (2007) 535–546.
[16] A. Munch-Hansen, C. Orlandi, S. Yakoubov, Stronger notions and a more efficient construction of threshold ring signatures, Cryptology ePrint Archive, Report 2020/678, `https://eprint.iacr.org/2020/678` (2020).

[17] J. K. Liu, D. S. Wong, On the security models of (threshold) ring signature schemes, in: International Conference on Information Security and Cryptology, Springer, 2004, pp. 204–217.
[18] T. Okamoto, R. Tso, M. Yamaguchi, E. Okamoto, A k-out-of-n ring signature with flexible participation for signers., IACR Cryptol. ePrint Arch. 2018 (2018) 728.
[19] P. P. Tsang, V. K. Wei, T. K. Chan, M. H. Au, J. K. Liu, D. S. Wong, Separable linkable threshold ring signatures, in: International Conference on Cryptology in India, Springer, 2004, pp. 384–398.
[20] T. H. Yuen, J. K. Liu, M. H. Au, W. Susilo, J. Zhou, Threshold ring signature without random oracles, in: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, 2011, pp. 261–267.
[21] P.-L. Cayrel, R. Lindner, M. Rückert, R. Silva, A lattice-based threshold ring signature scheme, in: International Conference on Cryptology and Information Security in Latin America, Springer, 2010, pp. 255–272.
[22] C. A. Melchor, P.-L. Cayrel, P. Gaborit, F. Laguillaumie, A new efficient threshold ring signature scheme based on coding theory, IEEE Transactions on Information Theory 57 (7) (2011) 4833–4842.
[23] S. Bettaieb, J. Schrek, Improved lattice-based threshold ring signature scheme, in: International Workshop on Post-Quantum Cryptography, Springer, 2013, pp. 34–51.
[24] A. Haque, A. Scafuro, Threshold ring signatures: New definitions and post-quantum security, in: IACR International Conference on Public-Key Cryptography, Springer, 2020, pp. 423–452.
[25] A. Haque, S. Krenn, D. Slamanig, C. Striecks, Logarithmic-size (linkable) threshold ring signatures in the plain model, Cryptology ePrint Archive, Report 2020/683, https://eprint.iacr.org/2020/683 (2020).
[26] C. Stathakopoulous, C. Cachin, Threshold signatures for blockchain systems, Swiss Federal Institute of Technology (2017).
[27] S. Mazumdar, S. Ruj, Design of anonymous endorsement system in hyperledger fabric, IEEE Transactions on Emerging Topics in Computing (2019).
[28] J. Dharani, K. Sundarakantham, K. Singh, S. M. Shalinie, Design of anonymous endorsers in hyperledger fabric with linkable threshold ring signature, ReBICTE 6 (2020).
[29] J. Dharani, K. Sundarakantham, K. Singh, et al., A privacy-preserving framework for endorsement process in hyperledger fabric, Computers & Security 116 (2022) 102637.
[30] Y. Dodis, A. Yampolskiy, A verifiable random function with short proofs and keys, in: International Workshop on Public Key Cryptography, Springer, 2005, pp. 416–431.
[31] T. Okamoto, K. Pietrzak, B. Waters, D. Wichs, New realizations of somewhere statistically binding hashing and positional accumulators, in: International Conference on the Theory and Application of Cryptology and Information Security, Springer, 2015, pp. 121–145.
[32] C.-P. Schnorr, Efficient identification and signatures for smart cards, in: Conference on the Theory and Application of Cryptology, Springer, 1989, pp. 239–252.
[33] E. Bangerter, J. Camenisch, S. Krenn, Efficiency limitations for $\sigma$-protocols for group homomorphisms, in: Theory of Cryptography Conference, Springer, 2010, pp. 553–571.
[34] D. Boneh, B. Bünz, B. Fisch, Batching techniques for accumulators with applications to iops and stateless blockchains, in: Annual International Cryptology Conference, Springer, 2019, pp. 561–586.
[35] I. Damgård, M. Koprowski, Generic lower bounds for root extraction and signature schemes in general groups, in: International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 2002, pp. 256–271.
[36] N. Andola, V. K. Yadav, S. Venkatesan, S. Verma, et al., Spychain: A lightweight blockchain for authentication and anonymous authorization in iod, Wireless Personal Communications (2021) 1–20.
[37] A. Rakoczy, Go 1.14 release notes (mar 2020).
URL https://golang.org/doc/go1.14
[38] R. Cramer, I. Damgård, B. Schoenmakers, Proofs of partial knowledge and simplified design of witness hiding protocols, in: Annual International Cryptology Conference, Springer, 1994, pp. 174–187.

## Appendix A. Supplementary Preliminaries

### *Appendix A.1. Anonymous Endorsement System in Hyperledger Fabric*

The framework to achieve anonymous endorsement system in Hyperledger Fabric using threshold ring signature is described below:

- GenerateKeys($1^\lambda$): It takes as input the security parameter and generate the keys for the endorsers.
- GenerateSignature($m, SK, R, t$): This is run by every endorser to generate their share of the threshold signature $\sigma_i$. It takes as input the message $m$ to be signed, secret key of the signer $SK$, ring of verification keys $R$ and the threshold $t$ that indicates the required number of endorsements on the message $m$ for the transaction to be valid.
- AssembleSignature($\sigma = \{\sigma^i\}_{i=1}^t$): All endorsers forward their individual signature or signature share $\sigma^i$ to the client. On receiving $t$ number of signatures, the client then aggregates them into a single endorsement as $\sigma = \{\sigma^i\}_{i=1}^t$ and appends it with the transaction.
- VerifySignature($m, R, \sigma, t$): Validators parse the individual signatures/signature shares to check if each of them are valid. It then checks if the individual contribution add up to the threshold $t$ and returns true if it is valid.

### *Appendix A.2. Commitment Scheme*

A commitment scheme is a triplet of PPT algorithms $(Gen, Commit, Open)$.

- $Gen(1^\lambda)$ generates a public commitment key $PK$ for the user taking the security parameter $\lambda$ as input.
- $Commit(PK, m)$ generates a commitment $C$ and an opening or decommitment information $d$ as output, taking the public key $PK$ and message $m$ to be committed.
- $Open(C, m, d)$ outputs a single bit as output indicating validity or invalidity of the commitment, taking as input the commitment $C$, message $m$ and the decommitment information $d$.

A commitment scheme should satisfy hiding and binding properties. Hiding ensures that a commitment should reveal no information about the message that is committed. Binding property guarantees that no malicious committer should be able to commit a message $m$ and later decommit to a different message $m'$.

### *Appendix A.3. Ring Signatures*

A ring signature scheme is a triplet of PPT algorithms $(KeyGen, Sign, Verify)$ to generate keys for the signer, sign a message and verify the signature on a message respectively.

- $KeyGen(1^\lambda)$ The key generation algorithm takes as input the security parameter $\lambda$ and outputs a secret key $SK$ and a verification key $VK$.

- $Sign(m, SK, R)$ The signing algorithm takes as input the message to be signed, secret key of the signer $SK$ and a ring of verification keys $R = \{VK_1, VK_2, ..., VK_n\}$ and outputs a signature $\sigma$.

- $Verify(m, \sigma, R)$ The verification algorithm takes as input the message $M$, signature $\sigma$ and the ring of verification keys $R$ and generates a single bit response to indicate the validity or invalidity of the signature.

A linkable ring signature additionally has a PPT algorithm *link* that avoids a signer from signing a message more than once with respect to a ring $R$. This linkability function is useful in e-voting applications to prevent a user from voting twice.

- $Link(\sigma_1, \sigma_2)$ This algorithm checks if the signatures $\sigma_1$ and $\sigma_2$ were produced by the same signer and outputs a single bit to indicate linked or unlinked.

A t-out-of-n threshold ring signature for $t \leq n$, allows $t$ users to generate a single threshold ring signature without revealing the identity of signers who contributed to the threshold signature. The signing and verification algorithms additionally take as input a threshold parameter $t$ that indicates t parties are involved in the signature generation process. The advantage of a threshold ring signature is that verification involves validating only a single threshold signature for the t-signers. Linkable threshold ring signatures should satisfy the following properties:

- Unforgeability: It guarantees that no PPT adversary can successfully generate a valid signature $\sigma_A$ on a message $m$ with respect to a ring $R$ such that $Verify(m, \sigma_A, R) = 1$ for $VK_A \notin R$.

- Linkable Anonymity: Anonymity of ring signatures guarantees that the signature does not reveal the signer's identity. Linkable anonymity guarantees signer anonymity but allows anyone to check if two signatures were generated by the same ring member.

- Transaction-oriented linkability: It ensures no two signatures were generated by the same endorser for same transaction.

- Non-Slanderability: It ensures that corrupt signers acting as adversary $A$ cannot succeed in forging a signature such that it is linkable with signature created by another honest endorser.

- Claimability and Repudiability: It allows a signer to claim a signature that he produced by revealing its identity. Repudiability allows the signer to prove that he did not generate a particular signature.

The proposed Endorser Anonymization Technique adapts to a modified linkable threshold ring signature scheme from [25] which in turn takes the framework from [12].

### *Appendix A.4. Sigma-Protocol*

A $\Sigma$-protocol [38] is a public-coin honest-verifier zero-knowledge proof of knowledge protocol between a prover $P$ and a verifier $V$ with a 3-move form:

1. Commit Phase: $P$ sends a commitment $I$ to a random value $r$ to $V$.
2. Challenge Phase: $V$ sends a random challenge t-bit string $e$ to $P$.
3. Response Phase: $P$ sends a response $z$ which is a function of witness $w$, $r$, $e$ to $V$.

$\Sigma$-protocol is said to be secure if it satisfies the following properties: i)Perfect Completeness: For honestly generated keys and $x$ belonging to a language $L$ the verification should always accept the proof. ii)Honest-Verifier Zero-Knowledge: Zero-knowledge property guarantees a prover that a malicious verifier learns nothing beyond the truth of the statement. Honest-verifier indicates that the verifier follows the protocol and chooses the challenge uniformly at random. iii) Knowledge-Extractable: It not only ensures that a witness $w$ exists for the statement $x$ but also that the prover knows the witness. This is modeled by a program interacting with the prover and extracts out the witness held by the prover.

### *Appendix A.5. Non-Interactive Zero-Knowledge Proof of Knowledge Representation (NI-ZKPoKRep)*

The protocol NI-ZKPoKRep is an honest-verifier statistically zero-knowledge argument of knowledge of integer vectors in the generic group model for the relation:

$$R_{NI-ZKPoKRep} = \{(((g_i, w) \in \mathbb{G}); \mathbf{x} \in \mathbb{Z}^n) :$$

$$w = \prod_{i=1}^{n} g_i^{x_i} \forall i \in [1, n]\}$$

The protocol allows a prover with an integer vector $\mathbf{x}= (x_1, ..., x_n)$ to prove that $w = \prod_{i=1}^{n} g_i^{x_i}$ without disclosing the witness $x = (x_1, ..., x_n)$.

- $Setup(\lambda)$ : The setup algorithm takes as input the security parameter and generates the parameters needed for the protocol. It samples a group $\mathbb{G}$ using a group generator function $GGen(\lambda)$, then samples generators $(g_1, ..., g_n)$ group $\mathbb{G}$.

- $Prove(\mathbf{x}, w)$ :
  - Prover randomly samples $k_1, ..., k_n \leftarrow_\$ [-B, B]$, where $B > 2^{2\lambda}|\mathbb{G}|$ and computes $A = \prod_{i=1}^{n} g_i^{k_i}$.
  - Computes $l \leftarrow H_{prime}(g_i, x_i, A)$ and $c \leftarrow H(l)$.
  - Computes $s_i = k_i + c.w_i \;\; \forall i \in [1, 3]$.
  - Derives quotients $\mathbf{q} \in \mathbb{Z}^3$ and residues $\mathbf{r} \in [l]^n$ such that $q_i.l + r_i = s_i \;\; \forall i \in [1, n]$.
  - Computes $Q = \prod_{i=1}^{n} g_i^{q_i}$.
  - Constructs the proof as $\pi \leftarrow \{A, Q, \mathbf{r}\}$.

- $Verify(\pi, w)$ :
  - Parse proof $\pi$ as $\{A, Q, \mathbf{r}\}$
  - Computes $l \leftarrow H_{prime}(g_i, x_i, A)$ and $c \leftarrow H(l)$
  - Check if: $r_i \in \{0, ..., l-1\}\ \ \forall i \in [1, 3]$; $Q^l \prod_{i=1}^{3} g_i^{r_i} \overset{?}{=} Ax^c$.

*Appendix A.6. Proof of Knowledge of Co-prime Roots (PoKCR)*

PoKCR is a technique to aggregate several NI-ZKPoKE proofs. It generates a succinct proof of knowledge for multiple co-prime roots $x_1, ..., x_n$ (i.e.,$gcd(x_i, x_j) = 1 : i \neq j$) for the relation:

$$R_{PoKCR} = \{(\alpha \in \mathbb{G}^n; x \in \mathbb{Z}^n) : \pi = f(x) \in \mathbb{G}\}$$

- AggProof($\alpha_i \in \mathbb{G}, x_i \in \mathbb{Z} \forall i \in [1, n]$): Aggregates the individual proofs as: $\pi \leftarrow \prod_{i=1}^{n} \pi_i$ such that $\pi_i^{x_i} = \alpha_i$ and forwards it to the verifier.
- VerifyAggProof($n, \alpha, x_i$): takes as input the total number of proofs to be aggregated $n$, element $\alpha$ for which the root needs to be calculated and the individual roots $x_i$ $\forall i \in [1, n]$. It recomputes $x \leftarrow \prod_{i=1}^{n} x_i$, and computes $y \leftarrow \prod_{i=1}^{n} \alpha_i^{x/x_i}$. Accepts, if $\pi^x \overset{?}{=} y$.