Design of Anonymous Endorsement System in Hyperledger Fabric

Subhra Mazumdar and Sushmita Ruj

Cryptology and Security Research Unit
Indian Statistical Institute
203 Barrackpore Trunk Road, Kolkata 700 108, India

Email : subhra.mazumdar1993@gmail.com, sush@isical.ac.in

November 15, 2018

Abstract

Permissioned Blockchain has become quite popular with enterprises forming consortium since it prioritizes trust over privacy. One of the popular platforms for distributed ledger solution, *Hyperledger Fabric*, requires a transaction to be *endorsed* or approved by a group known as endorsers as per the specifications in the endorsement policy. To endorse a transaction, an endorser mentions its identity along with the signature so that it can be verified. However, for certain transactions, difference in opinion may exist among endorsers. Disclosure of identity of endorser may lead to conflict within the consortium. In such cases, an endorsement policy which allows an endorser to support a transaction discreetly but simultaneously takes into account the decision of the majority is preferred. As a solution, we propose an Anonymous Endorsement System which uses a threshold endorsement policy. For hiding the identity of endorsers, a new ring signature scheme, called *Fabric's Constant-Sized Linkable Ring Signature* (FCsLRS) with *Transaction-Oriented* linkability has been proposed. We have implemented the signature scheme in Golang and analyzed its security and performance by varying the RSA modulus size. Feasibility of implementation is supported by experimental analysis. Signature generation and verification is quite fast, with execution time remaining constant irrespective of change in message length or endorsement set size for a given RSA modulus value. Lastly, we also discuss the integration of the scheme on v1.2 Hyperledger Fabric.

1 Introduction

In permissionless blockchains, the miners and other participants of the network can retain pseudo anonymity. No central authority has total controls over the functioning of the system. But for political or business organizations, transparency is of utmost importance for fair governance. On the contrary, the permissioned counterpart is governed by the members of the blockchain business network. Economic incentives, code quality, code changes, and power allocation among peers are based on the business dynamics for which the network has been designed and built.

Hyperledger Fabric is one of the most popular open-source permission blockchain frameworks. It is quite scalable and robust, hence used mainly for enterprise purpose. It's architecture is illustrated via a diagram in Fig.1. A *Membership Service Provider (MSP)* is responsible for maintaining the identities of all

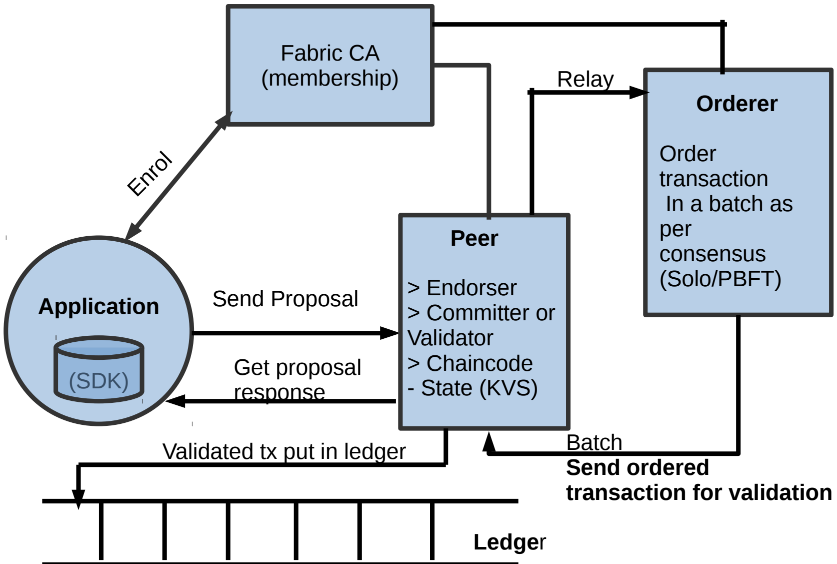


Figure 1: Hyperledger Fabric Architecture

nodes in the system - *clients*, *orderer* or *ordering service* and *validator or peer*. It issues credentials in the form of cryptographic certificates which is used for the purpose of authentication and authorization. *Fabric CA(Certification Authority)* acts as a private root CA provider capable of generating keys and certificate needed to configure an MSP. Shared and secure communication channel is provided to clients and orderers by the ordering service. It acts like private "subnet" of communication between two or more specific network members, for the purpose of conducting private and confidential transactions. Members (organizations), anchor peers per member, the shared ledger, chaincode application/s and the orderers define a particular channel. Any entity outside the channel cannot view any activity or status of any member inside a channel. A peer can be a member of multiple channels, and therefore maintain multiple ledgers. However, no ledger data can pass from one channel to another.

Any interaction among nodes occur through messages send via the channels and are authenticated via digital signatures. A client has to first connect to the channel, and then signs and submits a transaction proposal to the set of endorsers, specified by underlying endorsement policy. It is deemed as valid if and only if it collects sufficient number of endorsement for satisfying the endorsement policy. Endorsing peer after verifying the correctness of the transaction executes it against the current state database. Signature of the endorser along with endorser id is attached to *response*. Peer nodes, responsible for validation, verifies it by inspecting the identity and signature of each endorser. This ensures transparency in transaction flow where every details can be monitored. All transactions with required number of endorsement is passed on to the orderer, which establishes the total order of all transactions. The channel supports atomic delivery of all messages and outputs the transaction in the same logical order to all connected peers.

Revealing the identity of an endorser may not be suitable for sensitive transactions requiring support from majority of its members. Consider the following use case :

Example 1 *A referendum on allocation of funds by World Bank to a developing country, sharing hostile* *relation with major developed countries. The referendum succeeds if majority of them vote in favor of the motion. However the voting process is biased. Donor countries, though less in number, have more influence on any decision taken than any borrower country, force the judgement to be in their favor. Economically weaker countries have no role to play.*

If such a use case is deployed on Hyperledger Fabric with countries forming the set of endorsers, the voting process will be totally unfair. Over here, privacy is of utmost importance and hence we need an *Anonymous Endorsement System* where just the number of votes supporting the referendum is revealed and identity of endorsers remain hidden.

Blockchain was created to solve the specific problem of providing trust when all participants are anonymous. In permissionless setting, anyone can join the network without credentials. This is not true in the permissioned setting, where a participant's identity is know by everyone within the network. However, recently Hyperledger Fabric intends to integrate *Identity Mixer MSP (Membership Service Provider)*[9] in it's future releases where client and/or peer nodes can sign the transaction by remaining anonymous, generating unlinkable signature. With the aid of zero-knowledge proofs, a peer node can prove the correctness of the generated signature, without the verifier obtaining extra information. A privacy preserving distributed ledger, `zkLedger` [26] claims to support strong transaction as well as participants privacy (for hiding transaction value it uses Pedersen commitments) but at the same time provides fast and provably correct auditing using Schnorr-type non-interactive zero-knowledge proofs. But none of them address the concern of hiding the identity of endorser. This motivated us to design of a bias-free *endorsement system*, allowing provision of an endorsement policy which takes into account substantial amount of support by members of a pre-specified endorsement set without explicitly specifying who all must endorse.

A threshold endorsement policy serves the purpose which requires atleast t out of $n, (1 \leq t \leq n/2)$ endorsers to approve of the transaction without explicitly mentioning their identity. Any implementation of threshold cryptosystem might seem an obvious answer, but it does not ensure anonymity of endorser(explained later in Section3). Instead, we use *Ring Signature*, whose construction is based on *Signature of Knowledge*. Detachment of identity of signer from the endorsement leads to problem of double-signing whereby a member tries to endorse more than once. We prevent this by making multiple signatures generated by a particular signer for a particular session linkable. For accomplishment of threshold endorsement policy, verifiers resort to simple counting of each valid endorsement till it crosses the threshold value.

Our Contributions:

This paper makes the following contributions :

- We have proposed an *Anonymous Endorsement System* which implements a simple threshold endorsement policy which requires at least t out of n endorsers to approve of the transaction without explicitly mentioning their identity.
- We have given the construction of a constant sized linkable ring signature scheme, Fabric's Constant-Sized Linkable Ring Signature (*FCsLRS*), to hide the identity of an endorser.
- A new linking criterion, called as *"transaction-oriented"* linkability, is used, which prevents an endorser from signing the same transaction more than once.
- We have implemented this scheme in Golang (refer [1]) and analysed its performance.
- A detailed description of the integration of the scheme on v1.2 Hyperledger Fabric has been discussed.

1.1 Organization of the paper

`Section 2` discusses basic Fabric architecture and transaction flow. Few related works on various signature schemes and its application in blockchain system has been studied in `Section 3`. Mathematical notations and basic definitions has been discussed in `Section 4`. In `Section 5`, we describe our proposed anonymous endorsement system. Construction of Fabric's Constant-Sized Linkable Ring Signature (*FCsLRS*) scheme is stated in `Section 5.1`. `Section 6` describes the security model. Performance Analysis is given in `Section 7`. `Section 8` elucidates, in details, the integration of *FCsLRS* scheme with Fabric. Finally the paper is concluded in `Section 10`.

2 Background

2.1 Hyperledger Fabric

Up to version 0.6, Fabric used to follow the order-execute architecture which had several limitations. From version 1.0 onwards, it has been revamped to *execute-order-validate*, ensuring resiliency, flexibility, scalability and confidentiality. A distributed application for Fabric consists of two parts[2]:

- Set of logic encoding the rules for execution of transaction known as *smart contract* or *chaincode*. It needs to be installed across the channel's peer nodes and then instantiated in the channel itself by an authenticated member of the network.
- Conditions formed by basic operations of Boolean Algebra - "AND" or "OR" is used to define which endorser/s must endorse a transaction. This is called as *Endorsement Policy*. The policies can also generalize to logical expressions on sets, such as *t-out-of-n*.

2.1.1 Hyperledger Fabric Architecture.

Fig.1. gives a pictorial description of the network entities present in Hyperledger Fabric and their functioning. All the members are provided identity by a membership service provider (MSP). The transaction flow involves three main phases - execution , ordering and validation[2]. A client comes with transaction/s request and transmits it to the peer nodes which acts as endorsers. These peer nodes check the validity of the client's signature and checks the compliance of the transaction with the endorsement logic. If all these conditions gets satisfied, the peer *endorses* the transaction by executing it based on the version of keys for the values used in the transaction. This is called the *execution phase* as shown in step $\langle 1 \rangle$ & step $\langle 2 \rangle$ of Fig.2. The executed transaction along with the endorser's signature is sent back to the client node. If the endorsement policy is satisfied, it is broadcasted to the orderers nodes. In the *ordering phase* (step $\langle 3 \rangle$ of Fig.2), a pluggable consensus protocol is used to get a total ordering on the sequence of endorsed transactions grouped into blocks. These are broadcasted to all peer nodes for *validation* (step $\langle 4 \rangle$ of Fig.2). All peers deterministically validate the transactions in the same order by checking satisfaction of endorsement policy and version number of the keys present in the local key-value store.

3 Related Work

Even though hiding identity using ring signature is a quite well studied area [7],[8],[28],[11], it's applicability in cryptocurrencies and blockchain is being recently explored. Monero, an anonymous cryptocurrency, improves on its existing *Cryptonote* [33] protocol by using a new efficient Ring Confidential Transactions protocol -

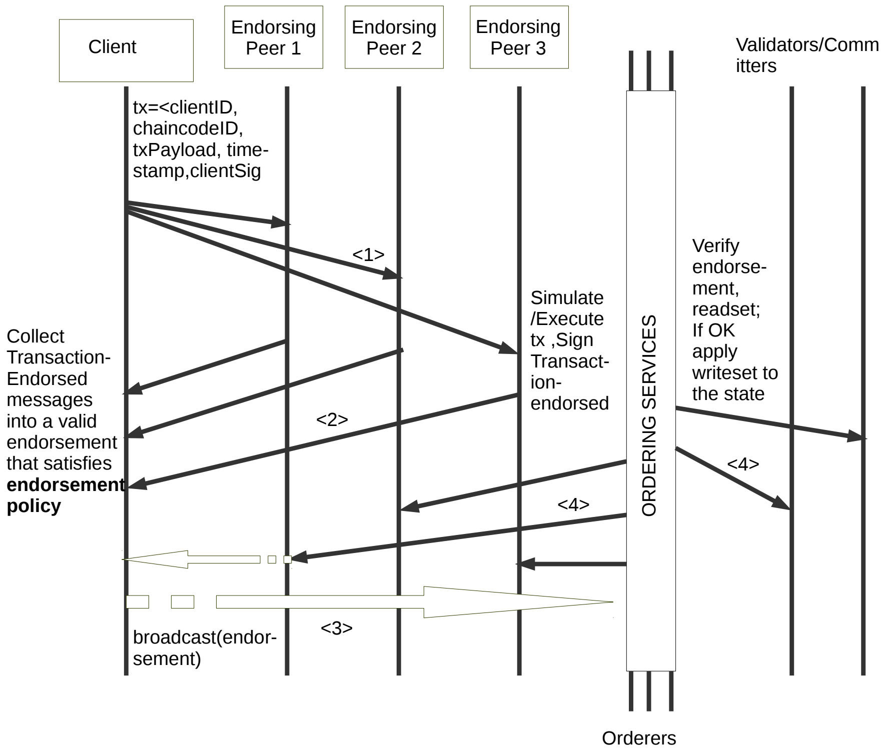


Figure 2: Transaction flow in Hyperledger Fabric v1.1

RingCT 2.0 [30]. It is based on the well-known Pedersen commitment, accumulator with one-way domain and signature of knowledge related to the accumulator. Over here, the size of signature is independent of the number of groups of input accounts in a transaction. Another cryptocurrency, *ZCash* requires a trusted set up stage, but after that the system is entirely anonymous, making use of zero-knowledge proof for verification [19]. Hardjono et al. [18] presented a new architecture called *ChainAnchor*, address the issue of retaining user anonymity, introducing the concept of *semi-permissioned* blockchains.

Various signature schemes mentioned in Table 1, do not ensure full anonymity. Some involve a third party which is responsible for signature generation (as in Bresson et al. [5]). Some are too complex to be implemented with signature size being dependent on the ring size (as in Yuen et al. [35]). Several threshold signature schemes for enhancing Bitcoin security has been proposed in Goldfeder et al. [16], Gennaro et al. [15] and Kogias et al. [24].

Till date, the only work which has focused on threshold signatures in permissioned blockchain - Hyperledger Fabric, is [29]. They have identified numerous potential application of threshold signature which can be used for group of Certificate Authorities, Byzantine Consensus protocols, chaincode applications and transaction validation. For this they have compared the performance of threshold signature schemes - threshold RSA signtaure/threshold BLS signature (ensures short signature size), against multisignature. In such schemes, a trusted third party is requested to generate the keys shares and distribute among signing parties (*Kate et al. [22]* proposed a scheme for distributed key generation scheme but it comes with fair amount of computation overhead). Any entity performing the task of signature combination in threshold signature scheme must know id value of the signer so that it can compute the Lagrange co-efficient. Verifier verifies just one signature instead of verifying each signature submitted by the endorsers. However, it leaks the identity of endorser to the entity who performs the task of recombining signature shares.

Since our problem statement demands anonymity on the identity of endorsers and not efficient transaction validation, we consider use of linkable ring signature for our anonymous endorsement system. It avoids all the complexities associated with the implementation of threshold cryptosystem.

Table 1: Comparison of Ring Signature Schemes

Scheme	Signature Size	Security Notions	Linking Complexity	Signing [1] Complexity	Verify [1] Complexity	Problem Encountered
1-out-of-n Ho. Au *et al.* [3]	$\mathcal{O}(1)$	Unforgeability, Linkable Anonymity, Linkability, Non slanderability - all wrt adversarially chosen keys	$\mathcal{O}(1)$: check linkability tag group oriented linkability	Uses Signature based on Proof of Knowledge, (n+2) exponentiation and 7 multibase exponentiation	7 multibase exponentiation	Adversary can corrupt the signer $\mathcal{S}$, overhead of certificate check, need event oriented linkability since ring members are fixed
t-out-of-n[5] Bresson *et al.*	$\mathcal{O}(l.2^t n \log n)$	Unforgeability, *t-CMA* secure Anonymity	unlinkable	$t.2^t \log n + t$ symmetric cipher operation, $n.2^t \log n + n$ exponentiation	$t.2^t \log n$ symmetric cipher op, $t.2^t \log n$ exponentiation	Prover may be malicious, t signers need to share their need secret keys, computationally expensive, double signing can't be prevented
t-out-of-n Tsang *et al.*[32]	$\mathcal{O}$(n)	Unforgeability, Linkable Anonymity, Event-oriented Linkability, Non slanderability.	$\mathcal{O}(n^2)$ Event-oriented Linkability	2(n+d) exponentiations and 2(n-d) multibase exponentiation	$\mathcal{O}$(n) multibase exponentiations	Prover/Signer may create 2 different event-id(double signing possible), Problem of CDS [11] scheme exist, sharing of secret key with prover $\mathcal{P}$,when $\mathcal{P}$ gets compromised, is not desired
1-out-of-n[35] Yuen *et al.*	$\mathcal{O}(\sqrt{n})$	Unforgeability, linkable anonymity event-oriented linkability, non-	$\mathcal{O}(1)$, event-oriented linkability	(8+4$\sqrt{n}$) exponentiation, (4+2$\sqrt{n}$) multibase exponentiation One-time signature	(8+8$\sqrt{n}$) pairing, 2 exponentiation, one-time verification	Dependency on event-id, ,complex verification mechanism, signature complexity is high
t-out-of-n[35] Yuen *et al.*	$\mathcal{O}(t.\sqrt{n})$	Unforgeability, linkable anonymity event-oriented linkability, non-slanderability	$\mathcal{O}(t \log t)$,event-oriented linkability	(8t+4t$\sqrt{n}$) exponentiation, (4t+2t$\sqrt{n}$) multibase exponentiation	(8t+8t$\sqrt{n}$) pairing, 2t exponentiation, t one-time verification	More complex than *1-out-of-n* signature scheme
URS Franklin,Zhang[14], [25]	$\mathcal{O}$(n)	Unforgeability, secure linkability, and restricted anonymity	$\mathcal{O}(1)$ - tag is hash of message, ring members and private key of signer	2n-1 multibase exponentiation and 1 exponentiation	2n multibase exponentiation	Computationally expensive scheme, not yet extended to *t-out-of-n* scheme.
Our proposed signature scheme *FCsLRS*	$\mathcal{O}(1)$, constant size	Unforgeability Linkability, Linkable Anonymity, Non-slanderability, -all wrt adversarially chosen keys	$\mathcal{O}(1)$, transaction oriented linkability	11 exponentiations and 5 multibase exponentiations	10 multibase exponentiations and 6 exponentiation	Extension to *t-out-of-n* signature scheme not efficient.

[1]Exponentiation operation is of the form g^a for base g, multibase exponentiation operation is of the form $g^a.h^b$ for base g, h.

4 Preliminaries

4.1 Mathematical Notations

We define the mathematical notations which will be used in our construction of FCsLRS scheme :

- $\lambda, l, \mu \in \mathbb{N} : \lambda > l - 2, l/2 > \mu + 1$ be the security parameters.
- $\mathtt{RSA}_\lambda$ be the set of RSA integers of size λ.
- A number p is a *safe prime* if $p = 2p' + 1$ and both p and p' are odd primes.
- A number N is an *RSA integer* if $N = pq$ for distinct safe primes p and q where $p = 2p' + 1$ and $q = 2q' + 1$. It is termed as a rigid integer $|p| = |q|$.
- Set of λ-bit rigid integers are denoted by $\mathtt{Rig}_\lambda$.
- $QR(N)$ denotes the group of quadratic residues modulo N of order $p'q'$.

4.2 Hardness Assumptions

- **Decisional Diffie-Hellman (DDH) Assumption.** [3] *"Consider a group G of order q, q is prime and let g be generator of G. Given $a, b, c \in_R \mathbb{Z}_q$, there exists no probabilistic polynomial time (PPT) algorithm that can distinguish two distributions $\langle g, g^a, g^b, g^{ab} \rangle$ and $\langle g, g^a, g^b, g^c \rangle$ with non-negligible probability over 1/2 in time polynomial in q."*
- **Strong RSA (SRSA) Assumption.** [3] *"Given input a random RSA integer N and a value $z \in_R QR(N)$, there exists no probabilistic polynomial time (PPT) algorithm which can return $u \in \mathbb{Z}_N^*$ and $e \in \mathbb{N}$ such that $e > 1$ and $u^e = z(mod\ N)$, with non-negligible probability and in time polynomial in λ."*
- **Link Decisional RSA (LD-RSA) Assumption.**[3] *"Given input a random RSA integer N, $\hat{g} \in_R QR(N), n_0 = p_0q_0$ and $n_1 = p_1q_1$ where p_0, q_0, p_1, q_1 are sufficiently large random primes of size polynomial in $\lambda, \hat{g}^{p_b+q_b}$ where $b \in_R \{0, 1\}$, there exists no PPT algorithm which returns $b' = b$ with probability non-negligibly over 1/2 and in time polynomial in λ."*

4.3 Building Blocks

Definition 1 *(Σ-**Protocols**.) [12] It is an efficient 3-round two-party protocol defined over an **NP**-relation R. For every input $(x, secret)$ given to prover P and x given to verifier V, the first round, initiated by prover P, yields a commitment message `COM`. In the second round, verifier V replies with a random challenge message `CH`. The last round by P concludes by sending response message `RES`. Finally, an honest verfier will output a 0 or 1, provided the transcript π =(`COM,CH,RES`) is valid and prover P possesses the secret.*

A Σ-protocol satisfies - *Special Soundness* and *Special Honest-Verifier Zero-Knowledge* property. This protocol can be efficiently constructed under the assumption that one way functions are easy to compute but hard to invert.

Definition 2 ***(Knowledge extractor)****. [17] On input x, auxiliary input $\tilde{z}$ and random input $r : x \in L_R$, and $\tilde{z}, r \in \{0,1\}^*$, let V outputs 1, after interacting with prover specified by $P_{x,\tilde{z},r}$ with probability $p(x,\tilde{z},r)$ and $\kappa(.)$ be error, $\kappa : \mathbb{N} \to [0,1]$. A probabilistic oracle machine K is called a* ***(universal) knowledge extractor****, if on input x (same as that given to V) and access to oracle $P_{x,\tilde{z},r}$, it outputs a solution $y \in R(x)$ within an expected number of steps bounded by*

$$\frac{q(|x|)}{p(x,\tilde{z},r) - \kappa(|x|)}$$

where $q(.)$ is a positive polynomial, provided $p(x,\tilde{z},r) > \kappa(|x|)$.

Definition 3 ***(Sphere Truncations of Quadratic Residues).****[12] Given that N is a RSA integer where $N = pq$, we define a sphere denoted by $S(2^l, 2^\mu) = \{2^l - 2^\mu + 1, \ldots, 2^l + 2^\mu - 1\}$ for two parameters $l, \mu \in N$ where $|S(2^l, 2^\mu)| = 2^{\mu+1} - 1$. Any random variable a^x with $x \in_R S(2^l, 2^\mu)$ is indistinguishable from the uniform distribution over $QR(N)$ provided factoring is hard and sphere $S(2^l, 2^\mu)$ is sufficiently large but not of the order of $QR(N)$. In simple terms, this means that a probabilistic polynomial-time observer cannot distinguish whether a value is selected from $S(2^l, 2^\mu)$ or $QR(N)$.*

Definition 4 ***(Discrete-log Relation Sets).*** *[12] For a group G of unknown order, a discrete-log relation set R with z relations over r free variables, $\alpha_1, \ldots, \alpha_r$, and m objects is a set of relations defined over the objects $A_1, \ldots, A_m \in G$, such that*

- *each free variable α_j is assumed to take value in a finite integer range $S(2^l_j, 2^\mu_j)$ where $l_j, \mu_j \geq 0$.*
- *i^{th} relation in the set R is specified by a tuple $\langle a^i_1, \ldots, a^i_m \rangle$ so that each a^i_j is selected to be one of the free variables $\{\alpha_1, \ldots, \alpha_r\}$ or an element of $\mathbb{Z}$. The relation is to be interpreted as $\Pi^m_{j=1} A_j^{a^i_j} = 1$.*

Such sets are quite useful in planning complex proofs of knowledge for protocols operating over groups of unknown order in general like for group $QR(N)$. A *discrete-log* relation set R is said to be *triangular*, if for each relation i involving the free variables $\alpha_1, \alpha_2, \ldots \alpha_k$, it holds that the free-variables $\alpha_1, \alpha_2, \ldots \alpha_k$ are contained in relations $1, \ldots, i-1$.

Definition 5 ***(Signature of Knowledge).*** *Instead of Challenger supplying the challenge value to Prover in three-round Σ-protocols or Honest-Verifier-Zero-Knowledge (HVZK) Proof of Knowledge (PoK) protocols, setting the challenge to the hash value of the commitment concatenated with the message to be signed [13] by the Prover transforms it into a signature scheme known as Signature based on Proof of Knowledge or simply 'Signature of Knowledge (SoK)'[8],[10]. Security of this scheme in the random oracle model is defined in[27],[4].*

Definition 6 ***(Accumulators with One-Way Domain).*** *[12][31] An accumulator family is a pair $(\{F_\lambda\}_{\lambda\in\mathbb{N}}, \{X_\lambda\}_{\lambda\in\mathbb{N}})$, where $(\{F_\lambda\}_{\lambda\in\mathbb{N}}$ is a sequence of families of functions such that each $f \in F_\lambda$ is defined as $f : U_f \times X^{ext}_f \leftarrow U_f$ for some $X^{ext}_f \subseteq X_\lambda$. It also satisfies the properties - efficient generation (in polynomial time in λ) and efficient evaluation (in polynomial time in λ). For all $\lambda \in \mathbb{N}$, $f \in F_\lambda$, $u \in U_f, x_1, x_2 \in X_\lambda$,*

$$f(f(u, x_1), x_2) = f(f(u, x_2), x_1) \tag{1}$$

[1]Exponentiation operation is of the form g^a for base g, multibase exponentiation operation is of the form $g^a.h^b$ for base g, h.

$\{X_\lambda\}_{\lambda\in\mathbb{N}}$ is referred to as the value domain of the accumulator. Due to the property of quasi-commutativity, such value is independent of the order of the x_i's and will be denoted by $f(u,X)$. For any $\lambda \in \mathbb{N}, f \in F_\lambda$ and $X = \{x_1,\ldots,x_s\} \subset X_\lambda$, $f(\ldots f(u,x_1),\ldots,x_s)$ is the accumulated value of the set X over u.

Based on the hardness assumption of Strong RSA, an accumulator with one-way domain[3] is a quadruple $(\{F_\lambda\}_{\lambda\in\mathbb{N}}, \{X_\lambda\}_{\lambda\in\mathbb{N}}, \{Z_\lambda\}_{\lambda\in\mathbb{N}}, \{R_\lambda\}_{\lambda\in\mathbb{N}})$, such that the pair $(\{F_\lambda\}_{\lambda\in\mathbb{N}}, \{X_\lambda\}_{\lambda\in\mathbb{N}})$ is a collision-resistant accumulator, each R_λ is a relation over $X_\lambda \times Z_\lambda$ with the following properties: (efficient verification). There exists an efficient algorithm D that on input $(x,z) \in X_\lambda \times Z_\lambda$, returns 1 if and only if $(x,z) \in R_\lambda$. (efficient sampling). There exists a probabilistic algorithm W that on input 1^λ returns a pair $(x,z) \in X_\lambda \times Z_\lambda$ such that $(x,z) \in R_\lambda$, z is the pre-image of x. (one-wayness). It is computationally hard to compute any pre-image z' of an x that was sampled with W. Formally, given a negligible value $\nu(\lambda)$, for any adversary $\mathcal{A}$:

$$Pr[(x,z) \xleftarrow{R} W(1^\lambda); z' \xleftarrow{R} \mathcal{A}(1^\lambda, x) \mid (x,z') \in R_\lambda] = \nu(\lambda) \tag{2}$$

For $\lambda \in \mathbb{N}$, the family F_λ consists of the exponentiation functions modulo λ-bit rigid integers :

$$\begin{gathered} f : QR(N) \times \mathbb{Z}_{N/4} \to QR(N) \\ f : (u,x) \to u^x \mod N \end{gathered} \tag{3}$$

where $N \in \texttt{Rig}_\lambda$.

The accumulator domain $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ is defined by:

$$X_\lambda = \{e \ prime \mid (\frac{e-1}{2} \in \texttt{RSA}_l) \wedge (e \in S(2^l, 2^\mu))\} \tag{4}$$

where $S(2^l, 2^\mu)$ is the integer range $(2^l - 2^\mu, 2^l + 2^\mu)$ that is embedded within $(0, 2^\lambda)$ with $\lambda - 2 > l$ and $l/2 > \mu + 1$. The pre-image domain $\{Z_\lambda\}_{\lambda\in\mathbb{N}}$ and the one-way relation $\{R_\lambda\}_{\lambda\in\mathbb{N}}$ are defined as follows:

$$Z_\lambda = \left\{ \begin{gathered} (e_1, e_2) \mid e_1, e_2 \ are\ distinct\ l/2 - bit \\ primes\ and\ e_2 \in S(2^{\frac{l}{2}}, 2^\mu) \\ R_\lambda = \{(x,(e_1,e_2)) \in X_\lambda \times Z_\lambda \mid (x = 2e_1e_2 + 1)\} \end{gathered} \right\} \tag{5}$$

5 Our Proposed Anonymous Endorsement System

To address the problem of biased endorsement policy as well as ensuring privacy of endorsers, we have designed an anonymous endorsement system. To ensure privacy of the system, we have proposed a new ring signature scheme which is discussed in Section 5.1.

In Hyperledger Fabric, membership service provider (MSP) identifies the parties, who are the members of a given organization in the blockchain network. The endorsement set for a particular chaincode is presumed to be predefined and remains fixed for a long time, unless any of the members get revoked. The right measure of "signature size" constructed for each transaction must not involve explicit description of the ring members (endorsers for this case). A one-time computation of accumulation of public keys proportional to the size of the ring needs to be performed by the *Fabric CA* and communicated to all the verifiers present in the network. This constant-sized information allows signers to generate or verifiers to verify many subsequent signatures in constant time.

Entities present in the network

Fabric CA(Certification Authority) Server [1] issuing enrolment certificates to all the peer nodes (endorser and validators). Setup mentioned in [20].
Client : An entity lying outside the blockchain network, having a transaction request. A peer node, within the network, acts as a proxy for the client node.
Endorser set $\mathcal{E}$: A pre-defined set to be specified before instantiation of chaincode. Members of this set, based on a given endorsement policy, decides on whether to endorse a transaction.
Signer S : A member of the endorsement set $\mathcal{E}$ which executes the ring signature algorithm on the transaction response packet for the endorsed transaction.
Verifier/Validator set $\mathcal{V}$: Validator nodes verify whether the signature was generated by a valid member of the endorsement set.

Requirement of the signature scheme

Signature and tag generated must be short. The signature generation and verification must be computationally efficient. None of the entities must get access to any secret of the signer. Apart from this, the following two properties must be ensured for the scheme :

- With negligible probability, valid signatures generated according to specification fails to get **accepted** during verification.

- With negligible probability, two signatures signed according to specification, generated by the same signer on the same transaction for a given set of endorsers fails to get **linked**.

5.1 Proposed construction of Fabric's Constant-Sized Linkable Ring Signature (FCsLRS)

In this section, we propose a new ring signature scheme called as *Fabric's Constant-Sized Linkable Ring Signature*(FCsLRS) for a fixed set of endorsers and discuss the construction details. Our construction is inspired by the signature scheme obtained by applying *Fiat-Shamir transformation* to the `Identification Protocol` suggested in Dodis et al.[12]. Previously, this identification protocol has been used as a short signature scheme by Tsang, et al.[31] and Ho Au, et al.[3] for e-Cash, e-voting and attestation. But none of them could have been used directly for our endorsement system as values used for construction of *Signature of Knowledge* allows an adversary to easily identify the endorser.

Considering n to be the number of members in the endorsement set and t ($1 \leq t < n$) to be the threshold value. FCsLRS is represented as a tuple (`Init, KeyGen, AccumulatePubKey, GeneratePubKeyWitness, Sign, Verify, Link`) of seven polynomial time algorithms, which are described below.

- `Init`. On input of security parameters, *Fabric CA (Certificate Authority)* prepares a collision-resistant accumulator with one-way domain. A generator $u \in QR(N)$ is picked up uniformly at random, where $N \in \texttt{Rig}_\lambda$. Public parameters $g, h, y, t, s, \zeta \in QR(N)$, is also generated. These parameters remain same across all the transactions.

[1] It is a private root CA provider capable of managing digital identities of Fabric participants that have the form of X.509 certificates

- `KeyGen`. On input the system's parameters generated in `Init` phase, each endorser $E_i \in \mathcal{E}, 1 \leq i \leq n$ generates key pairs $(pk_i, sk_i) = (y_i, (p_i, q_i))$, $y_i = 2p_iq_i + 1$, p_i and q_i being safe primes, by executing the probabilistic sampling algorithm W of their accumulator[2]. The range of $q_i \in S(2^{l/2}, 2^{\mu})$. Upon obtaining the key pair, endorser E_i submits its public key y_i and *verifiable credentials* to Fabric CA. The CA first checks whether such credentials matches with any of those present in *Certificate Revocation List (CRL)*. If yes, then its *enrolment certificate* was previously revoked and hence cannot be added as a network entity. Else, E_i proves in zero-knowledge to CA the correctness of the value y_i[6], [7]. If endorser is able to prove, then CA issues an *enrolment certificate* to it. The identity of the endorser, public keys $y_j, 1 \leq j \leq n$ along with enrolment certificate gets added to the public database $\mathcal{DB}$ (any valid entity in the network has access to this database).

- `AccumulatePubKey`. Fabric CA executes this algorithm for combining all the public keys in public database $\mathcal{DB}$. The accumulated value v calculated by using data from $\mathcal{DB}$, is :

$$\begin{aligned} v &= f(u, \{y_j | 1 \leq j \leq n\}) \\ &= f(f(\dots f(u, y_1), y_2), y_3) \dots, y_n) \\ &\dots\dots \\ &= (\dots(((u^{y_1} \mod N)^{y_2} \mod N)\dots)^{y_n} \mod N) \end{aligned} \tag{6}$$

  This value is generated and used for long time unless the endorsement set $\mathcal{E}$ changes. Hence the computation can be said to be performed one time before instantiation of chaincode in all the peer nodes of the network.

- `GeneratePubKeyWitness`. Each member e of set $\mathcal{E}$ computes witness $w_e \leftarrow f(u, \{y_i | 1 \leq i \leq n, i \neq e\}), \langle u \rangle = QR(N)$ for public key y_e, where accumulated value v can be generated by computing $v \leftarrow f(w_e, y_e)$. When the endorser is willing to endorse or sign a transaction, it uses this value w_e for construction of Signature based on Proof of Knowledge. As we have considered endorsement set to be fixed, even this value can be pre-computed.

- `Sign`. Endorser $E_\pi \in \mathcal{E}$ who wants to endorse a transaction is the Signer $\mathcal{S}$. It obtains the public key set $\mathcal{DB} = \{y_1, y_2, \dots, y_n\}$, possessing a valid *enrolment certificate* and has not been revoked (CA performs the check and informs if any endorser has been put in *CRL*).

  A new linking criterion called *Transaction-Oriented linkability* has been used in which one can tell if two signatures are linked if and only if they are signed by a common signer for a given transaction (similar to the concept of *Event-oriented* linkability in [32]). For this purpose we use a public parameter g_{tid} instead of simply using $g \in QR(N)$. To construct g_{tid}, we consider $g \in QR(N)$ and a function $\tilde{H} : \mathbb{N} \rightarrow \mathbb{G}, \mathbb{G} \subset \mathbb{Z}_{N/4}$ which generates $\tilde{tx} = \tilde{H}$(transaction-id), where *transaction-id* is unique for each transaction, which is again the hash of the transaction payload `txPayload`. Thus, $g_{tid} = f(g, \tilde{tx}) = g^{\tilde{tx}} \mod N$, where f is the function defined as in Eq. 3.

  For a given message $m \in \mathcal{M}$ (which is the *transaction-response*) which has a transaction id *transaction-id*, a private key $sk_\pi = (p_\pi, q_\pi)$ that corresponds to original public key, y_π, accumulated value v and secret value w_π, signer $\mathcal{S}$ does the following (notations used as per [3]):

$$SPK\left\{ \begin{pmatrix} w_\pi, y_\pi \ , \\ p_\pi, q_\pi \end{pmatrix} : \begin{array}{c} w_\pi^{y_\pi} = v \mod N \wedge \\ y_\pi = 2p_\pi q_\pi + 1 \ \wedge \ y_\pi \in S(2^l, 2^\mu) \\ \wedge \ q_\pi \in S(2^{\frac{l}{2}}, 2^\mu) \ \wedge \\ \tilde{y} = \theta_d(p_\pi, q_\pi) \end{array} \right\}(m) \tag{7}$$

[2] All endorsers run the sampling algorithm of the accumulator in parallel

where θ_d defined as $\theta_d(p_\pi, q_\pi) = g_{tid}^{p_\pi + q_\pi} \mod N$, is a one-way bijective mapping and $\tilde{y}$ is the tag generated corresponding to the signature.

*"Signature based on Proof of Knowledge is basically a signature scheme in which a signer can speak on behalf of any **NP** statement (as stated in 7) to which he knows a witness/es without revealing all the irrelevant information [10]."* Here the witness values are w_π, y_π, p_π and q_π. Any person who knows a satisfying assignment (that means posses the knowledge of witness) to the statements (in 7) has signed the message.

A practical Σ-protocol for relation stated in Eq. 7 is constructed using the framework of discrete logarithm sets over group $QR(N)$. The public parameters $g_{tid}, h, y, t, s, \zeta \in QR(N)$ with unknown relative discrete logarithms alongwith the sequence of public values T_1, T_2, T_3, T_4, T_5 such that

$$T_1 = g_{tid}^r \;\; , T_2 = h^r \zeta^{x+r} \;\; , T_3 = s^r g_{tid}^{e_2} \;\; , T_4 = wy^r , \;\; T_5 = t^r g_{tid}^{2e_1}$$

where $r \xleftarrow{R} [0, \lfloor N/4 \rfloor - 1]$ is used for the construction of proof.

The public values T_1 is for the free variable r, T_2 is for the free variable x, T_3 is for the free variable e_2, T_4 is for the free variable w and T_5 is for the free variable e_1. Note that all the construction from T_2 to T_5 satisfy the property of triangularity with respect to first relation T_1. The construction of T_2 cannot be chosen of the form $h^r g_{tid}^x \mod N$ since x belongs to the set $\mathcal{DB}$ whose size is negligible compared to the size $S(2^{\frac{l}{2}}, 2^\mu)$, exponential order of the security parameter l. If prover P sends this value of T_2 to verifier $\mathcal{V}$, it can figure out, in polynomial time, the public key of the endorser/signer during verification phase.

The **NP** statements used for generating Signature based on Proof of Knowledge is given below:

$$\begin{gathered}
T_1 = g_{tid}^r, \text{ (witness of } r) \\
T_2 = h^r.\zeta^r.\zeta^x = h^r.\zeta^{r+x}, \text{ (witness of } x \in S(2^l, 2^\mu)) \\
(T_1)^x = g_{tid}^{a_1}, \text{ (witness of } a_1) \\
(T1)^{e_2} = g_{tid}^{a_2}, \text{ (witness of } a_2) \\
T_3 = s^r g_{tid}^{e_2}, \text{ (witness of } e_2 \in S(2^{l/2}, 2^\mu)) \\
(T_4)^x = vy^{a_1}, \text{ (witness of } x) \\
(T_5)^{e_2} g_{tid} = t^{a_2}.g_{tid}^x \text{ (witness of } e_2 \text{ - a non-trivial factor of } x) \\
(T_3)^2 T_5 = s^{2r}.t^r.\tilde{y}^2 \text{ (correctness of } \tilde{y})
\end{gathered} \tag{8}$$

for the free variables r, x, e_2, a_1, a_2 such that $x \in \mathcal{S}(2^l, 2^\mu), e_2 \in S(2^l, 2^\mu), a_1 = rx$ and $a_2 = re_2$. The signer $\mathcal{S}$ gives a proof of knowledge for witness w, y_π, p_π and q_π by satisfying the above eight equations corresponding to the given accumulated value v. The variables x, e_1 and e_2 is assigned value y_π (public key of $\mathcal{S}$), p_π and q_π respectively. It also proves that $(x-1)/2$ can be factorized into two prime values, one of which belongs to $S(2^{l/2}, 2^\mu)$ and each of them are non trivial factors[12].

Public Parameters : $g_{tid}, h, t, s, y, \zeta, v \in QR(N)$, T_1, T_2, T_3, T_4, T_5, $\mathbb{Z}_{N/4} \subset S(2^l, 2^\mu)$.

1. `Signer` $\mathcal{S}$ computes

$$\begin{aligned}
&\alpha_1 \xleftarrow{R} \mathbb{Z}_{N/4} && \alpha_2 \xleftarrow{R} \mathbb{Z}_{N/4} \\
&\alpha_3 \xleftarrow{R} \mathbb{Z}_{N/4}, && u_1 \leftarrow g_{tid}^{\alpha_1} \ mod \ N \\
&u_2 \leftarrow \zeta^{\alpha_1+\alpha_2} \ mod \ N, && u_3 \leftarrow h^{\alpha_1} \ mod \ N, \\
&u_4 \leftarrow g_{tid}^{\alpha_1} \ mod \ N && u_5 \leftarrow g_{tid}^{\alpha_3} \ mod \ N \\
&u_6 \leftarrow w^{\alpha_2} \ mod \ N, && u_7 \leftarrow g_{tid}^{2e_1.\alpha_3} \ mod \ N \\
&u_8 \leftarrow t^{\alpha_1} \ mod \ N, && u_9 \leftarrow g_{tid}^{\alpha_2} \ mod \ N
\end{aligned} \tag{9}$$

2. $\mathcal{S}$ computes $c = H_1(m||u_1||u_2||u_3||u_4||u_5||u_6||u_7|| u_8||u_9), H_1 : \mathcal{M} \times QR(N)^8 \rightarrow \mathcal{C}$, $\mathcal{C} \subseteq QR(N)$ and uses it to compute

$$\begin{aligned}
&\tilde{\alpha_1} \leftarrow \alpha_1 + c.r, && \tilde{\alpha_2} \leftarrow \alpha_2 + c.x \\
&\tilde{\alpha_3} \leftarrow r.\alpha_2 + r.c.x, && \tilde{\alpha_4} \leftarrow \alpha_3 + c.e_2 \\
&\tilde{\alpha_5} \leftarrow r.\alpha_3 + r.c.e_2
\end{aligned} \tag{10}$$

$\mathcal{S}$ sends the signature $\sigma' = (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, \tilde{\alpha_1}, \tilde{\alpha_2}, \tilde{\alpha_3}, \tilde{\alpha_4}, \tilde{\alpha_5}, \tilde{y})$ where $\tilde{y} = g_{tid}^{p_\pi+q_\pi}$ to all the validators in set $\mathcal{V}$.

`Signature size` : Values to communicated to the *Verifier* as Signature based on Proof of Knowledge are $\sigma = (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, \tilde{\alpha_1}, \tilde{\alpha_2}, \tilde{\alpha_3}, \tilde{\alpha_4}, \tilde{\alpha_5}, \tilde{y})$, $u_1, u_2, \ldots, u_9, \tilde{y}, \tilde{\alpha_1}, \tilde{\alpha_2}$ and $\tilde{\alpha_4}$, each approximately being λ bits in size and $\tilde{\alpha_3}, \tilde{\alpha_5}$ each approximately being 2λ bits in size. Hence the signature generated is of constant size, being $\mathcal{O}(\lambda)$ where λ is the security parameter.

- `Verify`. To verify the signature $\sigma' = (u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8, u_9, \tilde{\alpha_1}, \tilde{\alpha_2}, \tilde{\alpha_3}, \tilde{\alpha_4}, \tilde{\alpha_5}, \tilde{y})$ on message $m \in \mathcal{M}$, all the validator nodes in $\mathcal{V}$ computes $c = H_1(m||u_1||u_2||u_3||u_4||u_5||u_6||u_7||u_8||u_9)$, $H_1 : \mathcal{M} \times QR(N)^8 \rightarrow \mathcal{C}, \mathcal{C} \subseteq QR(N)$ and checks if all the statements in Eq. 11 is valid or not. For *1-out-of-n* endorsement policy, if all check passes, then the verifier outputs `accept`; otherwise it outputs `reject` and aborts. For *t-out-of-n*, $t > 1$, we need to perform the test for signature linkability (`Link`) as well for final acceptance.

- **Link**. In [3], the tag generated is $\theta_d = ((e1, e2)) = g^{e_1+e_2}$. θ_d being PK-bijective, it prevented double signing on the same message. However as the endorsement set remains fixed, tag constructed must be function of the secret key as well as the transaction payload. For (*Transaction-Oriented linkability*), tag construction is modified by introducing transaction-id, a unique value associated with each transaction payload. Since $g_{tid} \in QR(N)$, modified $\theta_d = g_{tid}^{e_1+e_2}$ remains PK-bijective. Given two valid signatures σ_1' and σ_2' for a given transaction, validator node checks if $\tilde{y_1} = \tilde{y_2}$. If yes, output **linked**. Otherwise, output **unlinked**.

5.2 Extending to threshold endorsement policy

Given a threshold value t, if a validator node receives at least t out of n *transaction-response* with a valid, pairwise unlinked signatures (after $\binom{t}{2}$ tests of linkability) whose responses (read set and write set) are the same, then the endorsement policy is said to be satisfied. If each of at least $\frac{|\mathcal{V}|}{2}$ validator nodes in $\mathcal{V}$ reach a consensus on receipt of at least t signatures for the given transaction, then one of the honest validator node "broadcast" the *transaction-response* within a *transaction message* to the *ordering service* so that the transactions can be ordered chronologically by the channel.

$$
\begin{gathered}
g_{tid}^{\tilde{\alpha_1}} \stackrel{?}{=} u_1.T_1^c, \\
g_{tid}^{\tilde{\alpha_1}} \stackrel{?}{=} g_{tid}^{\alpha_1}.{g_{tid}^r}^c \mod N \ (\because Eq.\ 8,9,10), \\
g_{tid}^{\tilde{\alpha_1}} \stackrel{?}{=} g_{tid}^{\alpha_1+r.c} \mod N \\
\\
\zeta^{\tilde{\alpha_2}+\tilde{\alpha_1}} h^{\tilde{\alpha_1}} \stackrel{?}{=} u_2.u_3.T_2^c, \\
\zeta^{\tilde{\alpha_1}+\tilde{\alpha_2}} h^{\tilde{\alpha_1}} \stackrel{?}{=} \zeta^{\alpha_1+\alpha_2}.h^{\alpha_1}.(h^r.\zeta^{x+r})^c \mod N,\ (\because Eq.\ 8,9,10), \\
\zeta^{\tilde{\alpha_1}+\tilde{\alpha_2}} h^{\tilde{\alpha_1}} \stackrel{?}{=} \zeta^{\alpha_2+\alpha_1+c.(x+r)}.h^{\alpha_1+r.c} \mod N \\
\\
g_{tid}^{\tilde{\alpha_3}} \stackrel{?}{=} T_1^{\tilde{\alpha_2}}, \\
g_{tid}^{\tilde{\alpha_3}} \stackrel{?}{=} (g_{tid}^r)^{\alpha_2+c.x} \mod N,\ (\because Eq.\ 8,10), \\
g_{tid}^{\tilde{\alpha_3}} \stackrel{?}{=} g_{tid}^{r.\alpha_2+r.c.x} \mod N. \\
\\
g_{tid}^{\tilde{\alpha_5}} \stackrel{?}{=} T_1^{\tilde{\alpha_4}}, \\
g_{tid}^{\tilde{\alpha_5}} \stackrel{?}{=} (g_{tid}^r)^{\alpha_3+c.e_2} \mod N \ (\because of Eq. 8,10), \\
g_{tid}^{\tilde{\alpha_5}} \stackrel{?}{=} g_{tid}^{r.\alpha_3+r.c.e_2} \mod N. \\
\\
g_{tid}^{\tilde{\alpha_4}}.s^{\tilde{\alpha_1}} \stackrel{?}{=} T_3^c.u_4.u_5, \\
g_{tid}^{\tilde{\alpha_4}}.s^{\tilde{\alpha_1}} \stackrel{?}{=} (s^r g_{tid}^{e_2})^c.s^{\alpha_1}.g_{tid}^{\alpha_3} \mod N \ (\because Eq.\ 8,9,10), \\
g_{tid}^{\tilde{\alpha_4}}.s^{\tilde{\alpha_1}} \stackrel{?}{=} g_{tid}^{\alpha_3+c.e_2}.s^{\alpha_1+c.r} \mod N \\
\\
u_6.v^c.y^{\tilde{\alpha_3}} \stackrel{?}{=} T_4^{\tilde{\alpha_2}}, \\
w^{\alpha_2}.(w^x)^c.y^{\tilde{\alpha_3}} \stackrel{?}{=} (w.y^r)^{\tilde{\alpha_2}} \mod N \ (\because Eq.\ 8,9,10), \\
w^{\alpha_2+c.x}.y^{\tilde{\alpha_3}} \stackrel{?}{=} w^{\tilde{\alpha_2}}.y^{r.\tilde{\alpha_2}} \mod N \\
\\
t^{\tilde{\alpha_5}}.g_{tid}^{\tilde{\alpha_2}}.u_7 \stackrel{?}{=} T_5^{\tilde{\alpha_4}}.u_9.g_{tid}^c \\
t^{\tilde{\alpha_5}}.g_{tid}^{\tilde{\alpha_2}}.g_{tid}^{2.e_1.\alpha_3} \stackrel{?}{=} (t^r.g_{tid}^{2.e_1})^{\tilde{\alpha_4}}.g_{tid}^{\alpha_2}.g_{tid}^c \mod N \ (\because Eq.\ 8,9,10), \\
t^{\tilde{\alpha_5}}.g_{tid}^{\tilde{\alpha_2}+2.e_1.\alpha_3} \stackrel{?}{=} t^{r.\tilde{\alpha_4}} g_{tid}^{2.e_1.(\alpha_3+c.e_2)+\alpha_2+c} \mod N \\
t^{\tilde{\alpha_5}}.g_{tid}^{\tilde{\alpha_2}+2.e_1.\alpha_3} \stackrel{?}{=} t^{r.\tilde{\alpha_4}} g_{tid}^{2.e_1.\alpha_3+2.c.e_1.e_2+\alpha_2+c} \mod N \\
t^{\tilde{\alpha_5}}.g_{tid}^{\tilde{\alpha_2}+2.e_1.\alpha_3} \stackrel{?}{=} t^{r.\tilde{\alpha_4}} g_{tid}^{2.e_1.\alpha_3+c.(x-1)+\alpha_2+c} \mod N \\
\\
\tilde{y}^{2c}.s^{2\tilde{\alpha_1}}.t^{\tilde{\alpha_1}} \stackrel{?}{=} (T_3^2.T_5)^c.u_4^2.u_8 \\
\tilde{y}^{2c}.s^{2\tilde{\alpha_1}}.t^{\tilde{\alpha_1}} \stackrel{?}{=} ((s^r.g_{tid}^{e_2})^2.t^r.g_{tid}^{2.e_1})^c.(s^{\alpha_1})^2.t^{\alpha_1} \mod N(\because Eq.\ 8,9,10), \\
\tilde{y}^{2c}.s^{2\tilde{\alpha_1}}.t^{\tilde{\alpha_1}} \stackrel{?}{=} s^{2.r.c}.g_{tid}^{2.c.e_2}.t^{r.c}.g_{tid}^{2.e_1.c}.s^{2.\alpha_1}.t^{\alpha_1} \mod N \\
\tilde{y}^{2c}.s^{2\tilde{\alpha_1}}.t^{\tilde{\alpha_1}} \stackrel{?}{=} g_{tid}^{2.c.e_1+2.c.e_2}.s^{2.r.c+2.\alpha_1}.t^{r.c+\alpha_1} \mod N
\end{gathered} \tag{11}
$$

6 Security Model

Assumptions made

Some assumptions made regarding the entities in Hyperledger Fabric are - all communication channels are secure, Fabric CA is honest, members of endorsement set is fixed , all the peer nodes have their local copy of database consistent with the world state, signature generation algorithm follows a *Random Oracle Model.* In order to satisfy the threshold endorsement policy when at least t signers are willing to endorse, we assume that at least half of the members in the verifier set and more than half (at least $n/2+1$) endorsers in an endorsement set is honest. The security model defined here is similar to the one defined in [31],[3].

Syntax

A *Linkable Ring Signature* scheme is a tuple (**Init, KeyGen, AccumulatePubKey, GeneratePubKey-Witness, Sign, Verify, Link**) of seven polynomial time algorithms. Instead of a single entity generating keys for all the participants in the permissioned blockchain, we define **KeyGen** as an algorithm executed by each individual user for the generation of the public and private key pair. Syntax is as follows :

- **param** $\leftarrow$ **Init**(1^λ), the poly-time *initialization* algorithm which, on input a security parameter $\lambda \in \mathbb{N}$, outputs the system parameters containing, among other things, 1^λ. All other algorithms implicitly use λ as one of their inputs.

- $(sk_i, pk_i) \leftarrow$ **KeyGen()**, the PPT (*probabilistic polynomial time*) *key generation* algorithm which outputs a secret/public key pair (sk_i, pk_i). $\mathcal{SK}$ and $\mathcal{PK}$ denote the domains of possible secret keys and public keys respectively. All the generated $pk_i, 1 \leq i \leq n$ for n participants is made publicly available along with system parameters.

- $(v) \leftarrow$ **AccumulatePubKey()**, the deterministic poly-time algorithm which, on input a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$, where $n \in \mathbb{N}$ is of size polynomial in λ, produces the value v.

- $(w) \leftarrow$ **GeneratePubKeyWitness()**, the deterministic poly-time algorithm which, on input a set $\mathcal{Y}' = \mathcal{Y} \setminus \{pk_e\}$ i.e. all public keys except that of the entity e who executes it ($n \in \mathbb{N}$ is of size polynomial in λ), produces the value w_e such that $f(w_e, pk_e) = w_e^{pk_e} = v$.

- For a *Signatures based on Proofs of Knowledge*, the Σ-protocol between signer and verifier for the **NP**-relation stated in Eq.7 has been converted into a signature scheme. It comprises the **(Sign,Verify)** algorithm pair, executed on the signer and verifier side respectively. Execution of this protocol is time independent from the number of public keys that gets aggregated in **AccumulatePubKey** or **GeneratePubKeyWitness**.

  - $\sigma \leftarrow$ **Sign**$(\mathcal{Y}, M, x)$, the PPT *signing* algorithm which, on input a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$, where $n \in \mathbb{N}$ is of size polynomial in λ, a message $M \in \{0,1\}^*$, and a private key $x \in \mathcal{SK}$ whose corresponding public key is contained in $\mathcal{Y}$, produces a signature σ. We denote by Σ the domain of possible signatures.
  - $1/0 \leftarrow$ **Verify**$(\mathcal{Y}, M, \sigma)$, the poly-time *verification* algorithm which, on input a set $\mathcal{Y}$ of n public keys in $\mathcal{PK}$, where $n \in \mathbb{N}$ is of size polynomial in λ, a message $M \in \{0,1\}^*$ and a signature $\sigma \in \Sigma$, returns 1 or 0 meaning **accept** or **reject** respectively. If the algorithm returns **accept**, the message-signature pair (M, σ) is said to be *valid.* The signature scheme must satisfy *Verification*

Correctness, i.e. signatures signed by honest signer as per the specification must be accepted by an honest verifier with overwhelming probability.

- $1/0 \leftarrow$ **Link**(σ_0, σ_1), the poly-time linking algorithm which, on input two valid signatures, checks their corresponding tag and outputs 1 (if tags are same - signatures are linked) or 0 (if tags are different - unlinked signature) meaning **linked** or **unlinked** respectively. The signature scheme must satisfy *Linking Correctness*, i.e. any two signatures signed by a common honest signer on the same message are **linked** with overwhelming probability. On the other hand, any two signatures signed by two different honest signer must be **unlinked** with overwhelming probability.

Security Notions

Before defining the security notions, let us define the adversarial model and the possible attacks :

6.1 Adversarial Model

- Any corrupt endorser may launch insider attack (threat or use of influence) on the rest of the endorsers, acquiring their private keys.
- Members (excluding the signer) belonging to the endorsement set may collude and reveal their secret keys on receipt of signature.
- Validator may hold back the packets without verifying.
- Validator can act maliciously by randomly mark a transaction as valid/invalid without actually verifying.

Any adversary is assumed to have the following oracle access:

- The *Corruption Oracle,* which outputs the corresponding secret key given a public key as input.
- The *Signing Oracle,* which returns a valid signature, on input a designated signer s, message M and subring R (comprising subset of public keys). The signature is computationally indistinguishable from one produced by $\texttt{Sign}(\mathcal{Y}, M, x)$ using the real secret key x of signer s on message M, $\mathcal{Y}$ being the set of all public keys.

Note, that if endorsement logic gets corrupted by adversary then it needs a mechanism of formal verification to check whether desired output is achieved or not. This is beyond our scope of work. Based on the last two points, we discus the correctness and soundness property of the Σ-protocol as well as security of signature scheme in the permissioned blockchain framework.

6.2 Correctness

For correctness, any execution of the Σ-protocol for the NP-relation given in Eq.7 will terminate with the verifier outputting 1, with overwhelming probability, if and only if a prover or an endorser possess the correct witness values $(y_\pi, w_\pi, p_\pi, q_\pi)$ for the corresponding accumulated public value v.

6.3 Soundness

The Honest-Verifier Zero-Knowledge property of the Σ-protocol for **NP**-relations stated in Eq.7 guarantees that the transcript generated out of the interaction between signer and verifier does not leak any information to the adversary $\mathcal{A}$ that has no knowledge of the secret. The soundness property is formalized in terms of the game played between Fabric CA and adversary $\mathcal{A}$, assuming all endorsers, participating in ring formation, are honest.

- Fabric CA runs the `Init` algorithm for security parameter λ and generates system parameters. All endorsers executes `KeyGen` algorithm to generate the public key and private key pair and stored in public database $\mathcal{DB}$.
- $\mathcal{A}$ receives system parameters from Fabric CA and gets the transcript of prior runs of the protocol between an honest signer and verifier. Given $\mathcal{A}$ has access to the corruption oracle, it can query for the secret key of some but not all endorsers, who has put their public keys in database $\mathcal{DB}$.
- $\mathcal{A}$ now select a set of endorsers E' for which it has not queried their secret keys. It generates a value v' by accumulation of public keys of E'.
- $\mathcal{A}$ starts executing the Σ-protocol in the role of the signer and the probability of winning the game is negligible. Following the correctness property, an honest verifier with output 1(accept) with overwhelming probability if and only if the $\mathcal{A}$ can produce the correct secret value $(y_\pi, w_\pi, p_\pi, q_\pi)$ corresponding to accumulated value v'.

Note that if $\mathcal{A}$ is not given access to a correct tuple $(y_\pi, w_\pi, p_\pi, q_\pi)$ but still it wins the game, then it must have generated it by himself/herself. This contradicts the one-wayness of accumulator's domain.

6.4 Security Analysis of Signature Scheme

Since the architecture of *Fabric* is modular, the proposed signature scheme can be plugged-in as a feature. Given that Fabric is secure (Security model discussed in [21],[2]), we need to argue on the security of the proposed anonymous endorsement system based on the security of *FCsLRS* scheme.

Theorem 1 *If FCsLRS scheme is unforgeable, linkable anonymous, linkable and non-slanderable, then the scheme is secure and hence the proposed Anonymous Endorsement System also remains secure in the random oracle model.*

We have defined the security notions in details :

Unforgeability

The following construction of *constant-sized linkable ring signature* is unforgeable against "chosen endorser" attacks (means a subset of set E is selected and only the public keys of those endorsers are taken into consideration). The adversary is further allowed to corrupt endorsers and acquire their private keys using *corruption oracle.*

Definition 7 ***(Unforgeability)****.[3] Any corrupt endorser acting as adversary $\mathcal{A}$ is given access to the public keys of all the members belonging to endorsement set E as well as the signing oracle and corruption oracle. $\mathcal{A}$ is allowed to query the signing oracle for signature on message of its choice for a given subset of endorsers and use corruption oracle to get secret key for a set of corrupt endorsers denoted by $C : C \subset E$. A linkable* *ring signature scheme is unforgeable if for any PPT adversary $\mathcal{A}$ and for any polynomial n(.), the probability that $\mathcal{A}$ succeeds in forging signature on a message which it has not queried before, for a set of endorsers including at least more than one honest member, is negligibly close to* $1/2$.

Linkable-Anonymity

If all the endorsers except one honest signer gets corrupted and reveals their secret key in order to frame the signer ([3]), the scheme does not ensure anonymity anymore. But in FCsLRS, for *t-out-of-n* threshold endorsement policy, since more than half of the endorsers in $\mathcal{E}$ is assumed to be honest, possibility of such attacks is negligible. At least $n/2+1$ members in $\mathcal{E}$ will not reveal their secret keys .

Definition 8 ***(Linkable-anonymity)****.[3] Assuming that any corrupt endorser acting as adversary $\mathcal{A}$ has the same advantage of obtaining signature on message of its choice and querying for secret keys as stated in Definition 7. $\mathcal{A}$ now selects two public keys PK_{i_0}, PK_{i_1} of its choice (which has not been used for querying the signing oracle or corruption oracle), a message M and subset of E denoted by E_{sub}. The challenger now selects any public key at random out of PK_{i_0}, PK_{i_1} and generates a signature of M over set E_{sub}. After this step, $\mathcal{A}$ is allowed to query the signing oracle and corruption oracle for any public key except PK_{i_0} and PK_{i_1}. A linkable ring signature is linkably anonymous, if for any PPT adversary $\mathcal{A}$ and for any polynomial n(.), the probability that $\mathcal{A}$ succeeds in guessing the correct public key (either PK_{i_0} or PK_{i_1}), for which the signature was generated by challenger, is negligibly close to* $1/2$.

Adversarially-chosen keys defines the power of $\mathcal{A}$ which allows it to select public keys outside the set E for constructing set E_{sub} (but $PK_{i_0}, PK_{i_1} \in E$) and allowing use of such externally chosen public keys for querying the signing oracle. A linkable ring signature is linkably anonymous w.r.t adversarially-chosen keys, if for any PPT adversary $\mathcal{A}$ and for any polynomial n(.), the probability that $\mathcal{A}$ succeeds in guessing the correct public key (either PK_{i_0} or PK_{i_1}), for which the signature was generated by challenger, is negligibly close to $1/2$.

Linkability

Under the SRSA assumption of *Accumulators with one-way domain*, it is hard to generate the secret keys (e_1, e_2) for a given public key value x of an endorser. Thus the probability of producing a valid signature for a given transaction(message) and secret key pair is negligible. Security is ensured even in presence of adversarially-chosen keys.

Definition 9 ***(Linkability)****.[3] Assuming that any corrupt endorser acting as adversary $\mathcal{A}$ has the same advantage of obtaining signature on message of its choice and querying for secret keys as stated in Definition 7. A linkable ring signature is linkable if for any PPT adversary $\mathcal{A}$ and for any polynomial n(.), the probability that $\mathcal{A}$ succeeds in generating two different signature σ_1, σ_2 for the same message over same set of endorsers defined by E without getting linked (by returning $Link(\sigma_1, \sigma_2) = 0$), is negligibly close to* $1/2$.

The signature scheme is also linkable w.r.t to adversarially-chosen keys.

Non-slanderability

It ensures that any corrupt endorser cannot produce a linkable signature on behalf of or frame an honest endorser([34]). For our construction of FCsLRS, we consider a tag generation which provides *Transaction-Oriented* linkability. Since each tag generated is dependent on the transaction id and secret key pair and transaction id being unique for each transaction[3] and assuming the function $\tilde{H}$ is a random oracle , it is hard to produce two linked signatures for same transaction. Security is ensured even in presence of adversarially-chosen keys.

Definition 10 *(**Non-slanderability**).[3] Assuming that any corrupt endorser acting as adversary $\mathcal{A}$ has the same advantage of obtaining signature on message of its choice and querying for secret keys as stated in Definition 7. Given that an honest member of E with public key PK can generate a valid signature σ on a message M for subset of E, a linkable ring signature is non-slanderable if for any PPT adversary $\mathcal{A}$ and for any polynomial n(.), the probability that $\mathcal{A}$ succeeds in the generating a valid signature σ^* corresponding to the public key PK (provided this public key was not used while querying corruption oracle or signing oracle) such that $Link(\sigma^*, \sigma) = 0$, is negligibly close to $1/2$.*

The signature scheme is also non-slanderable w.r.t to adversarially-chosen keys.

Now we define the theorems that justifies the security of FCsLRS and provide the security proofs as well :

Theorem 2 *"If the DDH in QR(N) problem, the LD-RSA problem, the Strong-RSA problem are hard and the function $\tilde{H}$ is random oracle, our construction is unforgeable. [3]"*

Proof. If the signature scheme is *Non-Slanderable* and *Linkable* then it is *Unforgeable*. That is, if a corrupt endorser or malicious peer node can forge a signature, then he can even frame an honest endorser and sign on his behalf or collude with other corrupted endorsers to break the linkability of the signature. Therefore, theorems 4 and 5 together imply that the scheme is unforgeable.

Theorem 3 *"Under the assumption that endorsement set has at least two honest members, if the DDH in QR(N) problem, the LD-RSA problem, the Strong-RSA problem are hard and the function $\tilde{H}$ is random oracle , then our construction is linkably-anonymous w.r.t. adversarially-chosen keys.[3]"*

Proof Sketch. Here we proof the contrapositive of the theorem, i.e. if we can construct a simulator S from a corrupt endorser $\mathcal{A}$ which succeeds in guessing the correct public key (either PK_{i_0} or PK_{i_1}) corresponding to the signature generated by challenger (as defined in *Definition* 8), then it can also solve the LD-RSA problem under the DDH assumption.

Given the values $n_0 = p_0.q_0, n_1 = p_1.q_1$, a bit $b \in_R \{0, 1\}$ is selected randomly and the value $Tag = g_{tid}^{p_b+q_b}$ is generated by challenger. The values (n_0, n_1, Tag) is transmitted to S. Using the system parameters, it randomly generate a set of key pairs $J = \{(PK_i, SK_i)\}$ $1 \leq i \leq |E|$ in order to simulate the actual situation. Now S selects a bit $b' = 0$ or 1 randomly and sets PK^* to $2n_{b'} + 1$. The set of public keys $J^* = J \cup \{PK^*\}$ is then given to $\mathcal{A}$.

The adversary A may query the signing oracle for any of the public keys present in J^*. If the public key belongs to set J, then simulator can straight away generate the signature, given that it posseses the secret key. If the request comes from a public key not present in J (but not PK^*), S first asks the adversary to submit a proof showing that it has correctly generated the key pair. Here, S can extract the secret key during the proof of validity of public key. Now the signature generation can proceed as was stated before. If the request comes for public key PK^*, then S sets tag $\tilde{y} = Tag$ and computes the signature of knowledge. The simulated signature of knowledge (constructed as in Eq. 7.) is indistinguishable from the actual one under the DDH assumption in $QR(N)$, provided Tag is correctly formed. This is possible if and only if the bit b selected by challenger is same as bit b'. Since the advantage of breaking linkable anonymity is non-negligibly more than 1/2, advantage of S in generating a valid signature of knowledge will also be non-negligibly more than 1/2, thereby solving the LD-RSA problem.

[3]transaction-id is the hash of the transaction payload

Theorem 4 *"If the DDH in QR(N) problem, the LD-RSA problem, the Strong-RSA problem are hard and the function $\tilde{H}$ is random oracle, then our construction is linkable w.r.t. adversarially-chosen public keys of endorsers.[3]"*

Proof Sketch. In order to break the linkability property, a corrupt endorser $\mathcal{A}$ has to convince a verifier to accept a invalid tag $\tilde{y}$ with non-negligible probability. This is possible if $\mathcal{A}$ is successful in forging a signature for a given message over a given set of endorsers or it can generate an incorrect proof of construction for the invalid tag [12]. Either of the case is not possible, since forging a signature is hard under the Strong-RSA assumption and generating an incorrect proof is hard under LD-RSA assumption.

Theorem 5 *"If the DDH in QR(N) problem, the LD-RSA problem, the Strong-RSA problem are hard and the function $\tilde{H}$ is random oracle, then our construction is non-slanderable w.r.t. adversarially-chosen public keys of endorsers.[3]"*

Proof Sketch. If a corrupt endorser $\mathcal{A}$ is able to output a signature which slanders an honest endorser with public key PK^* and the soundness property of *Signature based on Proof of Knowledge* (Eq. 7) holds, then there exists a *knowledge extractor (def. 2)* which can extract the secret key (p', q') corresponding to the public key PK^* such that $PK^* = 2p'q' + 1$. Hence simulator S solves the LD-RSA problem.

7 Performance Analysis of FCsLRS [4]

The (**Sign,Verify**) algorithm pair just involves the execution of the Σ-protocol which is time independent from the number of public keys that were aggregated when constructing the accumulated value v and generation of witness value w. At the end of each protocol run, verifier $\mathcal{V}$ outputs a 0/1.

7.1 Theoretical Analysis

Table 2: Asymptotic complexity analysis of FCsLRS

Algorithm	Operations performed	Asymptotic complexity
Tag generation	$2E$	$\mathcal{O}(\lambda)$
Signature	$11E + 5M$	$\mathcal{O}(\lambda)$
Verification	$6E + 10M$	$\mathcal{O}(\lambda)$

E is an exponentiation (single base of the form g^a) operation, M is multibase exponentiation (of the form $g^a.h^b$) operation and λ is the security parameter.

7.2 Experimental Analysis

The performance of the signature scheme was measured on **Intel Core i5-4200U CPU, quad core processor, frequency 1.60 GHz**, OS : *Ubuntu-16.04 LTS* (64 bit). The programming language used is **Go 1.10**, packages used is *crypto, golang.org/x/ crypto/sha3, rand* and *math.* The code for *Cyclic Group Generator* is based on the one given under **Project-iris**[5]. For the ananlysis of the signature generation and verification time, the RSA modulus size, also the security parameter λ, has been varied as 1024 bits, 2048 bits and 3072

[4]Over here we analyze the performance for generation and verification of one signature

[5]https://github.com/project-iris/iris/blob/v0.3.2/crypto/cyclic/cyclic.go

bits. The endorsement set size (number of participants, n) was varied as $4, 8, \ldots, 256$, in ascending powers of 2 and message length was varied as 2KB,4KB and 8KB respectively. The functions H_1 and $\tilde{H}$ used is *SHA-3* producing a message digest of size 128 bits. The range of both the hash functions must be a subset of $QR(N), N$ is the RSA-modulus. Also as per Eq (4), any element in the group $QR(N)$ when raised to the power of a number in the group $\mathbb{Z}_{N/4}$, again returns an element which belongs to $QR(N)$. Since N is varied between 1024 bits to 3072 bits and least value of $N/4$ is 256 bits, so a message digest of 128 bit is definitely falling in the range $\mathbb{Z}_{N/4}$ for any case. We had to do this approximation since selection of an element from $QR(N)$ when $\phi(N)$ is unknown is hard. 10 instances for each message length was generated and in total, the FCsLRS algorithm was executed for 630 instances. However the execution time remains invariant of change in message length for a given number of participants and RSA modulus size . Thus for ease of tabulation, average over all 30 instances of such values was taken. Time taken by the signature generation and verification algorithm with respect to number of endorsers has been recorded in Table 3 and Table 4. Corresponding graphs has been plotted in Fig.3 and Fig. 4 respectively. The signature generation time and verification time remains constant for a fixed value of RSA modulus size. On varying the λ value, increase in computation time has been observed for both the algorithms. The Golang code for signature generation of *1-out-of-n* endorsement policy is available in [1].

Table 3: Signature generation time for FCsLRS vs number of participants

Endorsement set size	Signature generation time(ms) for 1024 bit RSA modulus	Signature generation time(ms) for 2048 bit RSA modulus	Signature generation time(ms) for 3072 bit RSA modulus
4	26.3810697	153.494131	460.1620130
8	25.9974438	156.6600916	460.5884873
16	25.9047032	153.3631163	461.7230385
32	26.0976248	153.1091731	463.5039483
64	25.9170626	153.0620514	462.9505392
128	25.187382	152.0527724	464.6665291
256	25.6127477	152.1151231	461.6722804

Table 4: Verification time for FCsLRS vs number of participants

Endorsement set size	Verification time(ms) for 1024 bit RSA modulus	Verification time(ms) for 2048 bit RSA modulus	Verification time(ms) for 3072 bit RSA modulus
4	33.8444186	189.9202852	563.9633690
8	34.1522705	194.7279413	564.7610533
16	33.7478959	190.1404563	561.8273189
32	33.8859284	190.7751803	565.7962644
64	34.4397107	190.8586968	564.2824740
128	34.1952231	191.0990971	564.6969238
256	34.3507846	191.6187186	565.3521524

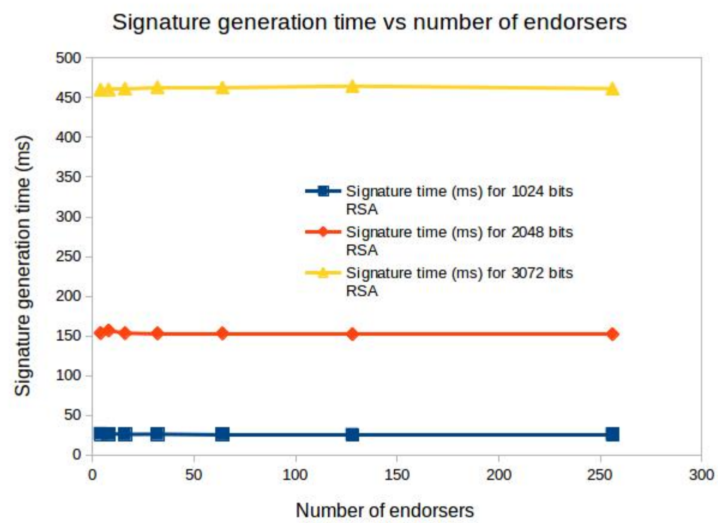


Figure 3: Signature Run time vs endorsement set size plot

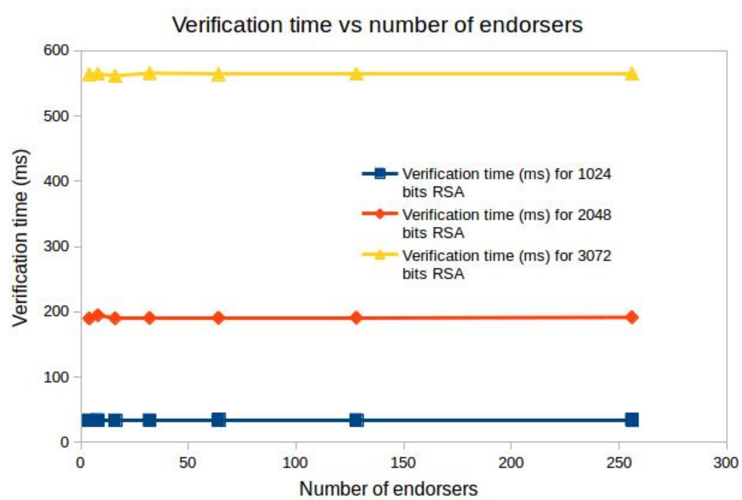
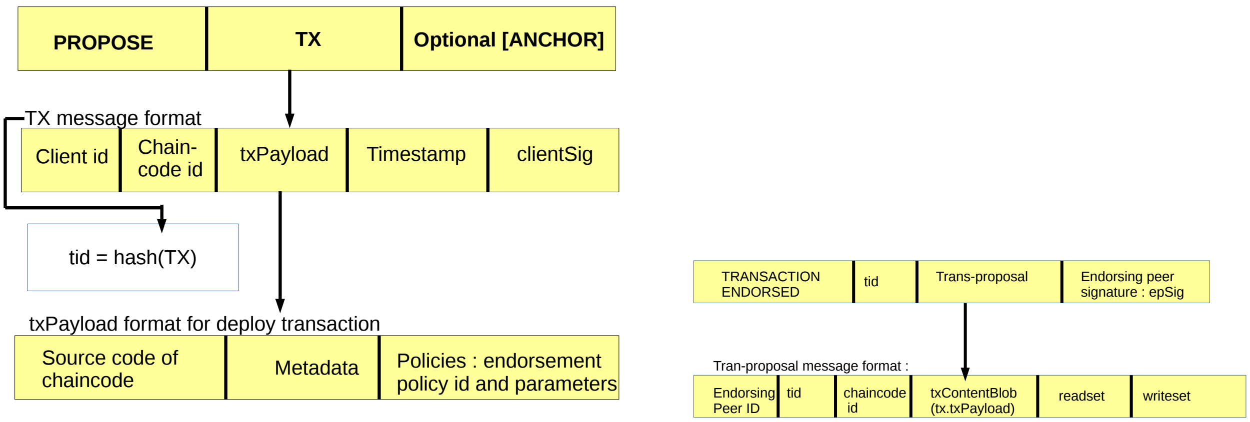


Figure 4: Verification Run time vs endorsement set size plot

8 Integration of Constant-Sized Linkable Ring Signature module in Hyperledger Fabric

In current workflow of transaction ([21]), to invoke a transaction, the client sends a "`PROPOSE` message" to the endorsers mentioned in the endorsement policy, provided endorsers are also part of the given *chaincodeID*.



(a) PROPOSE message format

(b) PROPOSE RESPONSE message format

Figure 5

The format of a `PROPOSE message` is ⟨*PROPOSE,tx,[anchor]*⟩ as shown in Fig. 5a.

The endorser with identity *epID*, on receipt of the `PROPOSE message` message,verifies the client's signature denoted by *clientSig*. It then executes the transaction (txPayload) by forwarding internally the `tran-proposal` to the part of its logic that endorses a transaction. Currently, the endorsing logic by default accepts the `tran-proposal` and signs the `tran-proposal`. However, one can change the endorsing logic as per the requirement to reach a decision whether to endorse a transaction or not. After endorsing the transaction, the peer sends a `PROPOSE-RESPONSE` packet containing the message - ⟨*TRANSACTION-ENDORSED, tid,* `tran-proposal`*,epSig*⟩ (as shown in Fig. 5b) to the submitting client, where: `tran-proposal` := *(epID,tid,chaincodeID,txContentBlob, readset,writeset)*. txContentBlob denotes chaincode/transaction specific information and *epSig* denotes the endorsing peer's signature on `tran-proposal`. If in case the endorsing logic refrains from endorsing the transaction, an endorser may send a negative acknowledgment to the submitting client stating its decision of rejecting the transaction[21].

To integrate the *Constant-Sized linkable ring signature* module, we first have to change the **PROPOSAL RESPONSE** format (as shown in Fig. 6). In the source code, under *hyperledger/fabric/protos/peer/ proposal_response.proto* in the structure **ProposalResponse**, add a field called as **Tag** which will enable *Transaction-oriented* linkability. The structure **Endorsement** must be changed by deletion of the field **endorser** (data type bytes[ ]) which reveals the endorsing peer ID. Create a **FCsLRS**[6] package under *hyperledger/fabric/bccsp* which can be used by the signer to sign the message. In the file *hyperledger/fabric/msp/identities.go*, delete the field **identity** in the structure **signingidentity**. The **Verify** function will just check the validity of the signature corresponding to a message. Instead of checking the identity of signer, it will verify whether signer can give a *Signature based on Proof of Knowledge* of the secret to prove its membership to the *Endorsement set* $\mathcal{E}$.

The *Transaction flow diagram* for *1-out-of-N'* endorsement policy is given in Fig. 7. It is clear that when *endorsing peer ep2* decides to endorse the transaction, it just forms a ring with rest of endorser in the *Endorsement set* to construct the ring signature and the **PROPOSAL RESPONSE** format being same as that shown in Fig.6. This is broadcasted to all the *Peer* nodes (⟨2⟩ of Fig. 7). If majority of the nodes reach a consensus on receipt of a valid endorsement, then one of the *validator* node forwards it to the *Ordering*

[6]Fabric's Constant-Sized Linkable Ring Signature

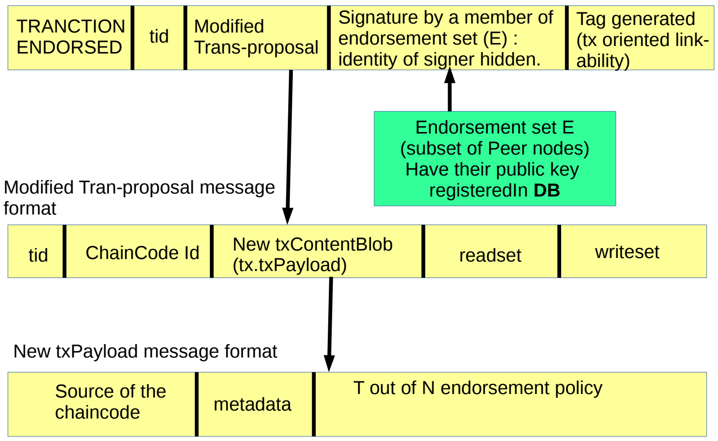


Figure 6: Modified PROPOSE message format

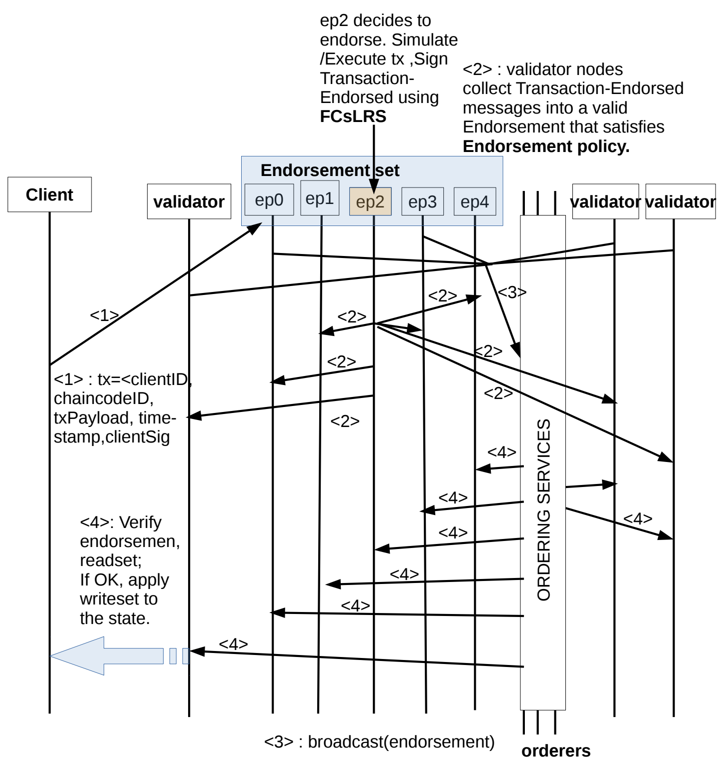


Figure 7: Modified transaction flow diagram : 1-out-of-N endorsement policy

Service(⟨3⟩ of Fig. 7). From this step onwards, the transaction flow is same as that shown in Fig. 2.

9 Description of the Implementation

In this section, we give a high level description of the main methods for *1-out-of-n* endorsement policy. Assuming that a signer S wants to endorse a transaction with transaction id as $tid = hash(transaction\ payload)$ and transaction payload denoted by $m \in \mathcal{M}$, where $\mathcal{M} \in 0,1^*, S \in \mathcal{E}$, $\mathcal{E}$ is the endorsement set, secret key is $sk_S = (p_\pi, q_\pi)$ and public key of S is $pk_S = 2p_\pi.q_\pi + 1$. Since we consider the case of just one endorser, we eliminate the code for check of linkability match as of now. But when it is integrated, check for linkability must be added for each transaction.

1. **Initialization.** This step involves generation of the RSA Modulus integer N of size λ bits. This step is executed by Fabric CA which generates the values by taking the security parameters as its input. To find a generator of $QR(N)$, we use the following lemma ([36]) :

   Lemma 1 *Let $N = p.q$ be the product of two distinct safe primes, and $u \in QR(N)$ a quadratic residue. Then u is a generator for $QR(N)$ if and only if $gcd(u-1, N) = 1$.*

Procedure 1: Initializations

Input : λ
Output: Public parameters : N, g, h, t, y, s, ζ

1. Generate 2 safe primes $p, q : p = 2p' + 1, q = 2q' + 1, |p| = |q| = \frac{\lambda}{2}$.
2. Find $N = p.q$.
3. Find a generator of the group $QR(N)$ using Lemma 1. Let that be u.
4. u generates g, h, t, y, s, ζ using some random discrete logarithm value $rd_i, 1 \le i \le 6, 2 \le rd_i \le |QR(N)| - 1$ where $|QR(N)| = p'.q'$.

---

2. **Key Generation.** Given an input n, which is the number of endorsers, each of the endorsers generate their own public key and private key pairs independently. (It only proves using zero-knowledge to Fabric CA about the correctness of the public key generated [7]). Upon key generation , these values are made available in the public database $\mathcal{DB}$. The procedure mentioned below must be run parallelly for each endorser present in endorsement set $\mathcal{E}$.

Procedure 2: Key Generation for endorser E_i

Input : $\lambda, l, \mu : \lambda > l - 2, \frac{l}{2} > \mu + 1$, where $\mathcal{E}$ is the endorsement set
Output: Public Key : pk_i, Secret Key : sk_i

1. Generate 2 prime $p, q, p \neq q : q \in (2^{\frac{l}{2}} - 2^\mu + 1, 2^{\frac{l}{2}} + 2^\mu - 1)$. $sk_i = (p, q)$.
2. Generate $pk_i : pk_i = 2p.q + 1$.
3. Send pk_i to database $\mathcal{DB}$.

---

[7] In our implementation, since we have developed the signature scheme as an independent module without considering any Public Key Infrastructure, so for the ease of implementation we have assumed the public keys generated by each endorser is correct.

3. **Public Key accumulation**. Fabric CA uses its accumulator with one-way domain to generate an accumulated value of all the public keys in $\mathcal{DB}$, each having valid enrolment certificate.

Procedure 3: Accumulated value computation

Input : all pk's in database $\mathcal{DB}$, generator $u, \langle u \rangle = QR(N)$
Output: Accumulated value : v

```
v ← u
for pk_i ∈ DB do
    v ← v^{pk_i} mod N
end
```

---

4. **Witness Generation for Signer S**. Signer $\mathcal{S}$ can generate the witness w using values of all public keys forming the ring except its own public key.

Procedure 4: Witness value for signer S

Input : all pk's in database $\mathcal{DB}$, generator $u, \langle u \rangle = QR(N)$, Signer S public key : pk_S
Output: witness value : w

```
w ← u
for pk_i ∈ DB : pk_i ≠ pk_S do
    w ← w^{pk_i} mod N
end
```

---

5. **Tag Generation**. To generate the tag, the signer S needs to compute g_tid from g given the transaction id *tid*.

Procedure 5: Tag generation for signer S

Input : Signer S secret key : $sk_S = (p_\pi, q_\pi)$, transaction id : tid
Output: tag value : $\tilde{y}$

```
x ← H̃(tid)
g_tid ← g^x mod N
ỹ ← g_tid^{p_π + q_π} mod N
```

---

6. **Computation of public values for Signature based on Proof of Knowledge Construction**. Signer S computes public values T_1, T_2, T_3, T_4, T_5 where

   $T_1 = g_{tid}^r \mod N, T_2 = (h^r \zeta^{pk_S + r}) \mod N, T_3 = (s^r g_{tid}^{q_\pi}) \mod N, T_4 = (w.y^r) \mod N, T_5 = (t^r g_{tid}^{2p_\pi})$ mod N.

7. **Signature Generation**. Signer generates the challenge value which can be generated again at the verifier side as well. This is the standard *Fiat-Shamir Transformation* which has been used. Send all these value (mentioned in the output of *Signature Algorithm* along with tag $\tilde{y}$ to verifier $v \in \mathcal{V}$.

Procedure 6: Public value generation by signer S

Input : Signer S secret key : $sk_S = (p_\pi, q_\pi)$, public key pk_S, $g_t id$, witness value : w

Output: Public values : T_1, T_2, T_3, T_4, T_5

$r \xleftarrow{R} \mathbb{Z}_{N/4}$

Compute T_1, T_2, T_3, T_4, T_5 as per equations mentioned above.

Procedure 7: Signature

Input : r,Signer S secret key : $sk_S = (p_\pi, q_\pi)$, public key pk_S, message $m \in \mathcal{M}$

Output: $u_1, u_2, \ldots, u_9, \tilde{\alpha_1}, \tilde{\alpha_2}, \ldots, \tilde{\alpha_5}$

1. Generate $\alpha_i, 1 \leq i \leq 3 : 0 < \alpha_i < N/4 - 1$.
2. Compute $u_1, u_2, \ldots, u_9$ as per Eq. 9.
3. Computes the challenge value $c = H_1(m, u_1, u_2, \ldots, u_9)$, where H_1 is a random oracle.
4. Using c, compute $\tilde{\alpha_1}, \ldots, \tilde{\alpha_5}$ as per Eq. 10.

8. **Verification Algorithm**. Verifier v computes c using the values sent to it by signer S. Using equations under Eq. 11 it's going to check whether the *Signature based on Proof of Knowledge* construction is correct or not.

10 Conclusion and Future Work

We have discussed about the design of an anonymous endorsement system for Hyperledger Fabric and gave the construction of a new constant sized linkable ring signature scheme, FCsLRS. We have integrated the construction into the existing framework and evaluated the performance. In our future work, we aim to provide a construction of short ring signature scheme, probably using pairing based cryptography. Currently, verifiers are required to count individual valid ring signature and check if the aggregate is above the threshold in order to implement threshold endorsement policy. We would like to replace it with threshold signature scheme which can guarantee the same or better level of anonymity as it is offered now by the proposed system. Since the endorsement policy is specific to Hyperledger Fabric, we would like to explore other permissioned blockchain systems and check whether the proposed scheme can be extended for other use cases.

11 Acknowledgement

This work is partially supported by *Cisco University Research Program Fund*, CyberGrants ID: #698039 and *Silicon Valley Community Foundation*. The authors would like to thank Chris Shenefiel, Samir Saklikar and Anoop Nannra for their comments and suggestions.

References

[1] Golang code : Fcslrs. **https://www.dropbox.com/sh/ 8zr0yvzuimtnyuu/AACUk0z8qRHC wecxXEIfePJ6a?dl=0** (2018)

The main methods described in section 9 is illustrated here.

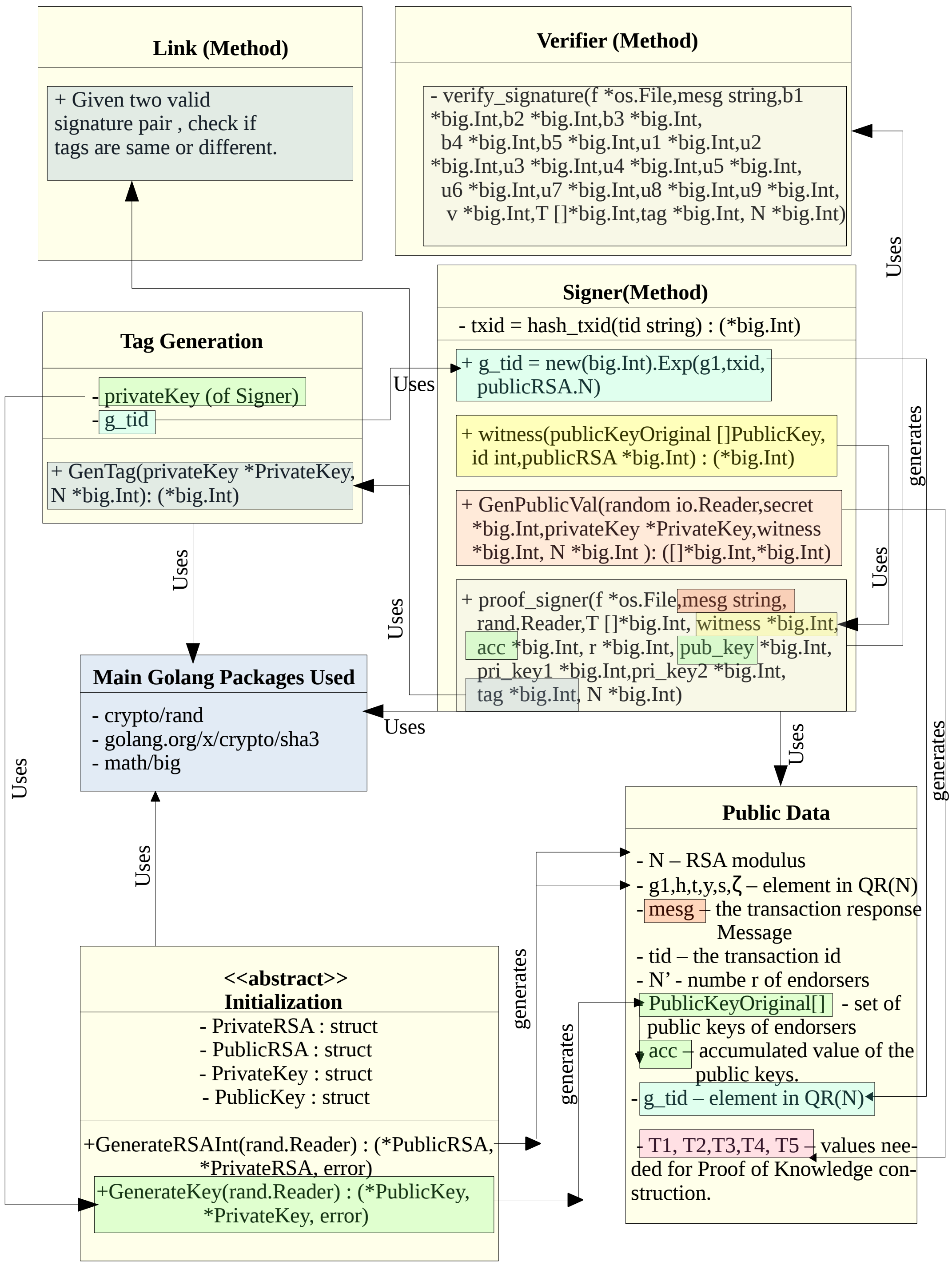


Figure 8: Class Diagram

[2] Androulaki, E., Barger, A., Bortnikov, V., Cachin, C., Christidis, K., De Caro, A., Enyeart, D., Ferris, C., Laventman, G., Manevich, Y., et al.: Hyperledger fabric: A distributed operating system for permissioned blockchains. arXiv preprint arXiv:1801.10228 (2018)

[3] Au, M.H., Chow, S.S., Susilo, W., Tsang, P.P.: Short linkable ring signatures revisited. In: European Public Key Infrastructure Workshop. pp. 101–115. Springer (2006)

[4] Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM conference on Computer and communications security. pp. 62–73. ACM (1993)

[5] Bresson, E., Stern, J., Szydlo, M.: Threshold ring signatures and applications to ad-hoc groups. In: Proceedings of the 22Nd Annual International Cryptology Conference on Advances in Cryptology. pp. 465–480. CRYPTO '02, Springer-Verlag, London, UK, UK (2002)

[6] Camenisch, J., Michels, M.: Proving in zero-knowledge that a number is the product of two safe primes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 107–122. Springer (1999)

[7] Camenisch, J., Michels, M., et al.: Separability and efficiency for generic group signature schemes. In: Annual International Cryptology Conference. pp. 413–430. Springer (1999)

[8] Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Annual International Cryptology Conference. pp. 410–424. Springer (1997)

[9] Camenisch, J., Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system. In: Proceedings of the 9th ACM conference on Computer and communications security. pp. 21–30. ACM (2002)

[10] Chase, M., Lysyanskaya, A.: On signatures of knowledge. In: Annual International Cryptology Conference. pp. 78–96. Springer (2006)

[11] Cramer, R., Damgård, I., Schoenmakers, B.: Proofs of partial knowledge and simplified design of witness hiding protocols. In: Advances in Cryptology—CRYPTO'94. pp. 174–187. Springer (1994)

[12] Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 609–626. Springer (2004)

[13] Feige, U., Fiat, A., Shamir, A.: Zero-knowledge proofs of identity. Journal of cryptology **1**(2), 77–94 (1988)

[14] Franklin, M.K., Zhang, H.: A framework for unique ring signatures. IACR Cryptology ePrint Archive **2012**, 577 (2012)

[15] Gennaro, R., Goldfeder, S., Narayanan, A.: Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In: International Conference on Applied Cryptography and Network Security. pp. 156–174. Springer (2016)

[16] Goldfeder, S., Gennaro, R., Kalodner, H., Bonneau, J., Kroll, J.A., Felten, E.W., Narayanan, A.: Securing bitcoin wallets via a new dsa/ecdsa threshold signature scheme (2015)

[17] Goldreich, O.: Foundations of Cryptography: Volume 1. Cambridge University Press, New York, NY, USA (2006)

[18] Hardjono, T., Smith, N., Pentland, A.: Anonymous identities for permissioned blockchains. Technical report (2014)

[19] Hopwood, D., Bowe, S., Hornby, T., Wilcox, N.: Zcash protocol specification. Tech. rep., Tech. rep. 2016-1.10. Zerocoin Electric Coin Company (2016)

[20] IBM, L.F..: hyperledger-fabric-ca Documentation , Release master. Read the Docs (2018)

[21] IBM, L.F..: hyperledger-fabricdocs Documentation , Release master. Read the Docs (2018)

[22] Kate, A., Goldberg, I.: Distributed private-key generators for identity-based cryptography. In: International Conference on Security and Cryptography for Networks. pp. 436–453. Springer (2010)

[23] Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 571–589. Springer (2004)

[24] Kogias, E.K., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., Ford, B.: Enhancing bitcoin security and performance with strong consistency via collective signing. In: 25th USENIX Security Symposium (USENIX Security 16). pp. 279–296 (2016)

[25] Mercer, R.: Privacy on the blockchain: Unique ring signatures. arXiv preprint arXiv:1612.01188 (2016)

[26] Narula, N., Vasquez, W., Virza, M.: zkledger: Privacy-preserving auditing for distributed ledgers. auditing **17**(34), 42 (2017)

[27] Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: International Conference on the Theory and Applications of Cryptographic Techniques. pp. 387–398. Springer (1996)

[28] Rivest, R., Shamir, A., Tauman, Y.: How to leak a secret. Advances in Cryptology—ASIACRYPT 2001 pp. 552–565 (2001)

[29] Stathakopoulou, C., Cachin, C.: Threshold signatures for blockchain systems. `https://domino.research.ibm.com/library/\cyberdig.nsf/papers/CA80E201DE9C8A0A852580FA\004D412F/File/rz3910.pdf` (2017)

[30] Sun, S.F., Au, M.H., Liu, J.K., Yuen, T.H.: Ringct 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency monero. In: European Symposium on Research in Computer Security. pp. 456–474. Springer (2017)

[31] Tsang, P.P., Wei, V.K.: Short linkable ring signatures for e-voting, e-cash and attestation. In: ISPEC. vol. 3439, pp. 48–60. Springer (2005)

[32] Tsang, P.P., Wei, V.K., Chan, T.K., Au, M.H., Liu, J.K., Wong, D.S.: Separable linkable threshold ring signatures. In: Indocrypt. vol. 3348, pp. 384–398. Springer (2004)

[33] Van Saberhagen, N.: Cryptonote v 2. 0 (2013)

[34] Wei, V.K.: Tracing-by-linking group signatures. In: International Conference on Information Security. pp. 149–163. Springer (2005)

[35] Yuen, T.H., Liu, J.K., Au, M.H., Susilo, W., Zhou, J.: Efficient linkable and/or threshold ring signature without random oracles. The Computer Journal **56**(4), 407–421 (2012)

[36] Micciancio, Daniele : The RSA group is pseudo-free. Annual International Conference on the Theory and Applications of Cryptographic Techniques. pp. 387–403, Springer (2005)