Constant Size Ring Signature Without Random Oracle

Priyanka Bose, Dipanjan Das, and C. Pandu Rangan

Indian Institute of Technology, Madras
{priyab,dipanjan,rangan}@cse.iitm.ac.in

Abstract. Ring signature enables an user to anonymously sign a message on behalf of a group of users termed as 'ring' formed in an 'ad-hoc' manner. A naive scheme produces a signature linear in the size of the ring, but this is extremely inefficient when ring size is large. Dodis *et al.* proposed a constant size scheme in *EUROCRYPT'13*, but provably secure in random oracle model. Best known result without random oracle is a sub-linear size construction by Chandran *et al.* in *ICALP'07* and a follow-up work by Essam Ghadafi in *IMACC'13*. Therefore, construction of a constant size ring signature scheme without random oracle meeting stringent security requirement still remains as an interesting open problem.
Our first contribution is a generic technique to convert a compatible signature scheme to a constant-sized ring signature scheme. The technique employs a constant size set membership check that may be of independent interest. Our construction is instantiated over asymmetric pairing of composite order and optimally efficient. The scheme meets strongest security requirements, *viz.* anonymity under full key exposure and unforgeability against insider-corruption without using random oracle under simple hardness assumptions. We also provide a concrete instantiation of the scheme based on Full Boneh-Boyen signatures.

Keywords: Ring Signature, Set Membership, Groth-Sahai protocol, Zero-Knowledge

1 Introduction

The idea of ring signature was introduced by Rivest, Shamir and Tauman [29]. The signature leaks no information more than the endorsement of the message by *some* ring member. Unlike a group in a group signatures [10], a ring is not administered by a manager. In practice, ring members may be completely unaware of each others' inclusion in the ring. To form a ring, the real signer arbitrarily chooses a set of *potential signers* including himself, thus concealing his identity. Since rings are formed on-the-fly, notions such as addition or deletion of users, revocation of signing rights or divulging the anonymity of the actual signer etc. are irrelevant.

Apart from regular properties, e.g. correctness and unforgeability that any signature scheme must have, ring signature [2] mandates anonymity. Correctness allows a ring member to sign on a message on behalf of the ring. Unforgeability is defined by the impossibility of a new signature to be generated by an adversary on behalf of the ring. Finally, anonymity demands a signature not being traceable to its signer. In other words, signatures produced on a message by any two members of the ring look alike.

The prime application of ring signature is in anonymous leaking of sensitive secrets as suggested in the original paper [29]. Another application is designated verifier signatures [22], where the verifier designated by confirmer/prover can obtain validity or invalidity of the proof. For more applications, refer to [12, 27]. Such protocol, often with additional blindness requirement, finds its relevance in e-voting or e-cash.

Table 1 gives a quick survey on the state-of-the-art of PKI based ring signatures.

1.1 Motivation

Most of the ring signature constructions [29, 1, 4, 20, 2, 11, 32, 6, 31, 7] are of linear size with respect to the size of the ring. First four constructions are in Random Oracle Model(ROM), remaining ones but last are without Random Oracle (RO) and the last one is in standard model. Often there are some limitations, e.g. [2] makes use of generic ZAP, which is a 2-round, public-coin, witness-indistinguishable proof system for any language in $\mathcal{NP}$, hence inefficient and far from being practical. Chow *et al.* introduced a new strong assumption in [11]. First sub-linear size ring signature scheme ($O(\sqrt{N})$) was proposed by Chandran, Groth and Sahai [8] followed by Ghadafi [14]. Both the schemes are provably secure without random oracle. To the best of our knowledge, only constant size ring signature scheme in PKI setting known so far is by Dodis *et al.* [12]. But, their approach uses Fiat-Shamir transformation and provably secure in random oracle model. So the idea of achieving constant-size ring signature without random oracle motivated our research.

Table 1. Survey of Ring Signatures in PKI Setting

Author	Reference	Size	Model	Remarks
Rivest, Shamir, Tauman	[29]	$O(N)$	ROM	Trapdoor permutation
Abe, Ohkubo, Suzuki	[1]	$O(N)$	ROM	RSA and DL based signatues
Boneh, Gentry, Lynn, Shacham	[4]	$O(N)$	ROM	Co-GDH based signature
Herranz, Sez	[20]	$O(N)$	ROM	Based on Schnorr Ring Signature
Bender, Katz, Morselli	[2]	$O(N)$	w/o ROM	ZAP based, Inefficient, Impractical
Chow, Wei, Liu, Yuen	[11]	$O(N)$	w/o ROM	$(q, n) - DsjSDH$ assumption
Shacham, Waters	[32]	$O(N)$	w/o ROM	All user keys belongs to same group
Boyen	[6]	$O(N)$	w/o ROM	-
Schage, Schwenk	[31]	$O(N)$	w/o ROM	Weak notion of unforgeability
Brakerski, Kalai	[7]	$O(N)$	Standard	Weak notion of unforgeability
Chandran, Groth, Sahai	[8]	$O(\sqrt{N})$	w/o ROM	-
Ghadafi	[14]	$O(\sqrt{N})$	w/o ROM	-
Dodis, Kiayias, Nicolosi, Shoup	[12]	$O(1)$	ROM	Fiat-Shamir transformation
Our construction	-	$O(1)$	w/o ROM	Composite order group

1.2 Our Contribution

Our major contribution is to present a generic technique to build a ring signature scheme on top of any compatible signature scheme, (such as Full-Boneh-Boyen [3]) having size independent of the cardinality of the ring. The scheme is instantiable in the most efficient Type-3 bilinear setting without any compromise in efficiency. We have attained the strongest possible security [2], e.g. anonymity under full key exposure and unforgeability against insider-corruption without using random oracle. Also we present a concrete instantiation of the technique to compare it with the existing schemes.

Lastly, our ring signature uses an $O(1)$ size membership checking protocol for an integer to be in a public set. It makes use of Groth-Sahai [16] commitment to realize witness indistinguishability and zero-knowledgeness. The protocol as well as its proof may be of independent interest.

1.3 Paper Organization

The paper is organized as follows: section 2 provides the necessary background pertaining to the ideas used in the paper. In section 3, we introduce a non-interactive, constant size membership proof to prove the knowledge of an element of a public set. Our main contribution, the construction of a constant-sized ring signature for PKI based cryptosystems is outlined in section 4. We have instantiated our construction in section 5. Conclusion and future directions are offered in section 6.

2 Preliminaries

2.1 Notations

By PPT, we mean a probabilistic polynomial time algorithm with respect to a security parameter κ. All adversaries defined here will be PPT except stated otherwise. Given a probability distribution $\mathcal{D}$ and an element y, $y \leftarrow \mathcal{D}$ denotes selecting an element y according to $\mathcal{D}$. Let $\mathcal{A}$ be a probabilistic algorithm , then $\mathcal{A}(x_1...x_n)$ describes the output distribution of $\mathcal{A}$ based on inputs $x_1, x_1, ..., x_n$. $\mathbb{Z}_n$ denotes set of all integers modulo n, where n is composite, product of two safe primes. The set of all polynomials in x with coefficients in $\mathbb{Z}_n$ is represented by $\mathbb{Z}_n[x]$. A function $\nu : \mathbb{N} \rightarrow \mathbb{R}^+$ is said to be negligible if $\forall c > 0$, $\exists k'$ such that $\nu(k) < k^{-c}$ for all $k' < k$.

2.2 Bilinear Groups

A bilinear pairing defined to be $\mathcal{G} = (n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ where we choose $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$ and $\mathbb{G}_T$ as multiplicative groups of order n. A bilinear pairing e is a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ having the following properties.

- ***Bilinearity***: For $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_n$ the following holds true: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- ***Non-degeneracy***: For any $\mathcal{X} \in \mathbb{G}_1$ and $\mathcal{Y} \in \mathbb{G}_2$, if $e(\mathcal{X}, \mathcal{Y}) = 1_T$, the identity element of $\mathbb{G}_T$, then either $\mathcal{X}$ is the identity of $\mathbb{G}_1$ or $\mathcal{Y}$ is the identity of $\mathbb{G}_2$.
- ***Efficiently Computable***: The map e should be efficiently computable.

Three main types of pairings exist in the literature[13, 33].

– **Type-1**. The groups $\mathbb{G}_1$ and $\mathbb{G}_2$ are the same.
– **Type-2**. $\mathbb{G}_1 \neq \mathbb{G}_2$, but an efficiently computable isomorphism $\zeta : \mathbb{G}_2 \to \mathbb{G}_1$ exists.
– **Type-3**. $\mathbb{G}_1 \neq \mathbb{G}_2$ and no efficiently computable isomorphism are known between $\mathbb{G}_1$ and $\mathbb{G}_2$.

We will use asymmetric pairing over groups of composite order which can be shown to be generated efficiently using the method in [24]. Meiklejohn *et.al* have shown that this setting has an advantage of the resulting curve having an embedding degree $k = 1$ and thus optimally efficient.

2.3 Hardness Assumptions

All the hardness assumptions are stated below:

- **Discrete logarithm Assumption (DL)**. Given a generator g of $\mathbb{G}$ and $a \in_R Z_n$ and for all PPT adversary $\mathcal{A}_{DL}$, the probability
$$|Pr[\mathcal{A}_{DL}(g, g^a) = a]| < \nu(\kappa)$$
- **Decisional Diffie-Hellman Assumption (DDH)**[28]. Given a cyclic group $\mathbb{G} = \langle g \rangle$, a tuple $(g, g^a, g^b, g^{ab}, g^c)$ where $a, b, c \in_R \mathbb{Z}_n$ and for all PPT adversaries $\mathcal{A}_{DDH}$, the probability
$$|Pr[\mathcal{A}_{DDH}(g, g^a, g^b, g^{ab}) = 1] - Pr[\mathcal{A}_{DDH}(g, g^a, g^b, g^c) = 1]| < \nu(\kappa)$$
- **Symmetric External Diffie-Hellman Assumption (SXDH)**. DDH holds in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$.
- **Decisional Linear Assumption (DLIN)**. For Type-1 bilinear groups where $\mathbb{G}_1 = \mathbb{G}_2 = \mathbb{G}$ and $\mathbb{G} = \langle g \rangle$, given $\langle g^a, g^b, g^{ra}, g^{sb}, g^t \rangle$ and $a, b, s, r, t \in Z_p$ being unknown, it is hard to tell whether $t = r + s$ or t is random.
- **Subgroup Hiding Assumption (SGH)**. Given a generation algorithm $\mathcal{G}$, which takes security parameter κ as input and gives output a tuple $\langle \mathcal{G}, sk \rangle$, where $\mathcal{G} = (n, \mathbb{G}, \mathbb{G}_T, e)$ and $sk = (p, q)$ such that $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and $\mathbb{G}$ and $\mathbb{G}_T$ are both groups of order $n = pq$, it is computationally in feasible to distinguish between an element of $\mathbb{G}$ and an element of $\mathbb{G}_p$. More formally all PPT adversaries $\mathcal{A}_{SGH}$, the probability
$$\begin{aligned}|Pr[(\mathcal{G}, sk) \leftarrow \mathcal{G}(1^\kappa); \mathcal{G} = (n, \mathbb{G}, \mathbb{G}_T, e, x); n = pq; sk = (p, q); x \leftarrow \mathbb{G}; \mathcal{A}_{SGH}(n, \mathbb{G}, \mathbb{G}_T, e, x) = 0] \\ - Pr[(\mathcal{G}, sk) \leftarrow \mathcal{G}(1^\kappa); \mathcal{G} = (n, \mathbb{G}, \mathbb{G}_T, e, x); n = pq; sk = (p, q); x \leftarrow \mathbb{G} : \mathcal{A}_{SGH}(n, \mathbb{G}, \mathbb{G}_T, e, x^q) = 0]| \\ < \nu(\kappa)\end{aligned}$$
where $\mathcal{A}_{SGH}$ outputs a 1 if it believes $x \in \mathbb{G}_p$ and 0 otherwise.
SGH to be hard in asymmetric pairing over composite order groups means, it is hard in both $\mathbb{G}_1$ and $\mathbb{G}_2$.
- **q-Strong Diffie-Hellman Assumption (q-SDH)**[3]. Let $\alpha \in_R \mathbb{Z}_p$. Given as input a $(q+1)$-tuple $\langle g, g^\alpha, g^{\alpha^2}, ..., g^{\alpha^q} \rangle \in \mathbb{G}^{q+1}$, for every adversary $\mathcal{A}_{q\text{-}SDH}$, the probability
$$Pr[\mathcal{A}_{q\text{-}SDH}(g, g^\alpha, g^{\alpha^2}, ..., g^{\alpha^q}) = \langle c, g^{\frac{1}{\alpha + c}} \rangle] < \nu(\kappa)$$
for any value of $c \in \mathbb{Z}_p \backslash \{-\alpha\}$ Though naturally q-type assumptions are defined on prime order groups, it has been shown in [9] that all q-type assumptions can also be proven to be secure in composite order groups provided subgroup hiding assumption holds.

- **Square Root Modulo Composite (SQROOT)[25]**. Given a composite integer n and $a \in Q_n$(the set of quadratic residues modulo n), it is computationally hard to find a square root of $a \mod n$; that is an integer x such that $x^2 \equiv a \mod n$, where $n = pq$, product of two safe primes.

  In all the above definitions $\nu(\kappa)$ is negligible in the security parameter κ.

2.4 Groth-Sahai Proofs

Groth-Sahai[16, 17] introduced a highly efficient and flexible proof system in common reference string(CRS) model that yields Non-Interactive Witness-Indistinguishable(NIWI) and Zero-Knowledge(NIZK) proofs. This system can be used for proving satisfiability of certain types of equations under various cryptographic assumptions. This proof system can be instantiated both in prime and composite order bilinear groups. The set of equations provable in this framework are as follows:

$\mathcal{X}_1, ..., \mathcal{X}_m \in \mathbb{G}_1, \mathcal{Y}_1, ..., \mathcal{Y}_n \in \mathbb{G}_1$, $x_1, ..., x_{m'} \in \mathbb{Z}_n$ and $y_1, ..., y_{n'} \in \mathbb{Z}_n$ are variables.

Pairing product equation:

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \underline{\mathcal{Y}}_i) \cdot \prod_{i=1}^{m} e(\underline{\mathcal{X}}_i, \mathcal{B}_i) \cdot \prod_{i=1}^{m}\prod_{j=1}^{n} e(\underline{\mathcal{X}}_i, \underline{\mathcal{Y}}_j)^{\gamma_{ij}} = t_T$$

For constants $\mathcal{A}_i \in \mathbb{G}_1, \mathcal{B}_i \in \mathbb{G}_2, t_T \in \mathbb{G}_T, \gamma_{ij} \in \mathbb{Z}_n$

Multi-scalar multiplication equation in $\mathbb{G}_1$:

$$\sum_{i=1}^{n'} \underline{y}_i \mathcal{A}_i + \sum_{i=1}^{m} b_i \underline{\mathcal{X}}_i + \sum_{i=1}^{m}\sum_{j=1}^{n'} \gamma_{ij} \underline{y}_j \underline{\mathcal{X}}_i = T_1$$

For constants $\mathcal{A}_i, T_1 \in \mathbb{G}_1$ and $b_i, \gamma_{ij} \in \mathbb{Z}_n$

Multi-scalar multiplication equation in $\mathbb{G}_2$:

$$\sum_{i=1}^{n} a_i \underline{\mathcal{Y}}_i + \sum_{i=1}^{m'} \underline{x}_i \mathcal{B}_i + \sum_{i=1}^{m'}\sum_{j=1}^{n} \gamma_{ij} \underline{x}_i \underline{\mathcal{Y}}_j = T_2$$

For constants $\mathcal{B}_i, T_2 \in \mathbb{G}_2$ and $a_i, \gamma_{ij} \in \mathbb{Z}_n$

Quadratic equation in $\mathbb{Z}_n$:

$$\sum_{i=1}^{n'} a_i \underline{y}_i + \sum_{i=1}^{m'} \underline{x}_i b_i + \sum_{i=1}^{m'}\sum_{j=1}^{n'} \gamma_{ij} \underline{x}_i \underline{y}_j = t$$

For constants $a_i, b_i, \gamma_{ij}, t \in \mathbb{Z}_n$ For clarity we will underline the elements of the witness in the description of equations.

With multiplicative notation, Multi-scalar multiplication equations will be Multi-scalar multi-exponential equations.

The Groth-Sahai proof systems consists of following PPT algorithms as defined in [14]

`(GSSetup,GSCRSGen,GSCommit,GSProve,GSVerify)`

- **GSSetup**(1^κ): It takes security parameter κ as input and produces description of bilinear group $\mathcal{G}$ and secret key sk as output.

- **GSCRSGen**$(\mathcal{G}, sk)$: Given group description $\mathcal{G}$ as input, it outputs common reference string(CRS) crs and an extraction key xk.

- **GSCommit**(w, τ_w): Given a witness $w \in \mathbb{G}$ and randomness τ_w, it produces a commitment to w with randomness τ_w, We denote the commitment to a witness w as Υ_w in this paper.

- **GSProve**$(\mathcal{G}, crs, x, w)$: It uses `GSCommit(.,.)` internally and outputs proof $\phi = \langle \boldsymbol{\Upsilon}, \boldsymbol{\Gamma} \rangle$, where $\boldsymbol{\Upsilon} = \langle \boldsymbol{c}, \boldsymbol{d} \rangle$ and $\boldsymbol{\Gamma} = \langle \boldsymbol{\pi}, \boldsymbol{\theta} \rangle$ as defined in [17, p. 12]. In symmetric setting $\boldsymbol{\Upsilon} = \langle \boldsymbol{c} \rangle$ and $\boldsymbol{\Gamma} = \langle \boldsymbol{\pi} \rangle$. In asymmetric setting for linear equation $\boldsymbol{\Upsilon} = \langle \boldsymbol{c} \rangle$. Here, $\boldsymbol{\Upsilon}$ is called commitment to witnesses.

- **GSVerify**$(\mathcal{G}, crs, x, \phi)$: It takes the tuple $(\mathcal{G}, crs, x, \phi)$ as input and outputs 1 if proof ϕ is accepted or 0 if rejected.

In addition to the above algorithms we also define the following ones:

`(GSExtract,GSSimSetup,GSSimProve)`

- **GSExtract**$(\mathcal{G}, crs, xk, \phi)$: It takes the tuple $(\mathcal{G}, crs, xk, \phi)$ as input and outputs the witness w used in the proof ϕ.

- **GSCRSSimGen**$(\mathcal{G})$: It takes group description $\mathcal{G}$ and outputs simulated CRS crs_{sim} and a trapdoor key td.

- **GSSimProve**$(\mathcal{G}, crs_{sim}, td, x)$: It takes $\mathcal{G}$, simulated CRS crs_{sim} and a trapdoor key td and generates simulated proofs ϕ_{sim}.

The system works by first committing to the elements of the witness and then producing the proof of satisfiability of all equations. If one witness component is involved in more than one equation, then same commitment is used during verification, thereby making the proofs correlated. First let R be a some efficiently computable ternary relation having elements of the form $(\mathcal{G}, x, w)$. let L be the set of all statements in R for a fixed $\mathcal{G}$. The proof system will have the following properties as defined in [17, 16]:

1. **Perfect Completeness:** We say `(GSSetup, GSCRSGen, GSProve, GSVerify)` is perfectly complete if for all adversaries $\mathcal{A}$ we have

$$Pr[(\mathcal{G}, xk) \leftarrow \texttt{GSSetup}(1^\kappa); crs \leftarrow \texttt{GSCRSGen}(\mathcal{G}, sk); (x, w) \leftarrow \mathcal{A}(\mathcal{G}, crs); \\ \phi \leftarrow \texttt{GSProve}(\mathcal{G}, crs, x, w) : \texttt{GSVerify}(\mathcal{G}, crs, x, \phi) = 1 \text{ if } (\mathcal{G}, x, w) \in R] = 1$$

2. **Perfect Soundness:** A non-interactive proof is sound if it is impossible to prove a false statement. More formally, for all adversaries $\mathcal{A}$ we have:

$$Pr[(\mathcal{G}, sk) \leftarrow \texttt{GSSetup}(1^\kappa); crs \leftarrow \texttt{GSCRSGen}(\mathcal{G}, sk); (x, \phi) \leftarrow \mathcal{A}(\mathcal{G}, crs) \\ : \texttt{GSVerify}(\mathcal{G}, crs, x, \phi) = 0 \text{ if } x \notin L] = 1$$

3. **Perfect L_{co}-Soundness:** We can consider L_{co} is language that depends upon $\mathcal{G}$ and crs. Standard soundness is a special case of L_{co} soundness where $\bar{L} = L_{co}$ and an adversary tries to create a valid proof for statement in $\bar{L}$. In the literature[19, 18] it has been shown that it is impossible to create valid proof for statements in L_{co}. So we say `(GSSetup, GSCRSGen, GSProve, GSVerify)` is L_{co}-sound if for all adversaries $\mathcal{A}$ we have:

$$Pr[(\mathcal{G}, sk) \leftarrow \texttt{GSSetup}(1^\kappa); crs \leftarrow \texttt{GSCRSGen}(\mathcal{G}, sk); (x, \phi) \leftarrow \mathcal{A}(\mathcal{G}, crs) \\ : \texttt{GSVerify}(\mathcal{G}, crs, x, \phi) = 0 \text{ if } x \in L_{co}] = 1$$

4. **Composable Witness Indistinguishability:** Composable witness indistinguishability introduces the notion of simulated CRS. An adversary can not distinguish between a real CRS and simulated CRS. It is also required that on a simulated CRS different witnesses used to construct the proof are also indistinguishable. Formally for all adversaries $\mathcal{A}$ we have:

$$Pr[(\mathcal{G}, sk) \leftarrow \texttt{GSSetup}(1^\kappa); crs \leftarrow \texttt{GSCRSGen}(\mathcal{G}, sk) : \mathcal{A}(\mathcal{G}, crs) = 1] \\ - Pr[(\mathcal{G}, sk) \leftarrow \texttt{GSSetup}(1^\kappa); crs_{sim} \leftarrow \texttt{GSCRSSimGen}(\mathcal{G}, sk) : \mathcal{A}(\mathcal{G}, crs_{sim}) = 1] < v(\kappa)$$

and for all adversaries $\mathcal{A}$ we have

$$Pr[(\mathcal{G}, sk) \leftarrow \texttt{GSSetup}(1^\kappa); crs_{sim} \leftarrow \texttt{GSCRSSimGen}(\mathcal{G}, sk); (x, w_0, w_1) \leftarrow \mathcal{A}(\mathcal{G}, crs_{sim}); \phi \leftarrow \texttt{GSProve}(\mathcal{G}, crs_{sim}, x, w_0) : \mathcal{A}(\phi) = 1] = Pr[(\mathcal{G}, sk) \leftarrow \texttt{GSSetup}(1^\kappa); crs_{sim} \leftarrow \texttt{GSCRSSimGen}(\mathcal{G}, sk); (x, w_0, w_1) \leftarrow \mathcal{A}(\mathcal{G}, crs_{sim}); \phi \leftarrow \texttt{GSProve}(\mathcal{G}, crs_{sim}, x, w_1) : \mathcal{A}(\phi) = 1]$$

$(\mathcal{G}, x, w_0), (\mathcal{G}, x, w_1) \in R$

5. **Composable Zero-Knowledge:** Composable Zero-Knowledge states that an adversary can not distinguish between a real CRS and simulated CRS. Even if it is given access to some secret trapdoor information td, it can not distinguish between real proofs and simulated proofs on simulated CRS. Therefore, `(GSSetup, GSCRSGen, GSProve, GSVerify)` is composable zero-knowledge if for all adversaries $\mathcal{A}$ we have:

$$Pr[(\mathcal{G}, sk) \leftarrow \texttt{GSSetup}(1^\kappa); (crs_{sim}, td) \leftarrow \texttt{GSCRSSimGen}(\mathcal{G}, sk); (x, w) \leftarrow \mathcal{A}(\mathcal{G}, crs_{sim}, td); \phi \leftarrow \texttt{GSProve}(\mathcal{G}, crs_{sim}, x, w) : \mathcal{A}(\phi) = 1] = Pr[(\mathcal{G}, sk) \leftarrow \texttt{GSSetup}(1^\kappa); (crs_{sim}, td) \leftarrow \texttt{GSCRSSimGen}(\mathcal{G}, sk); (x, w) \leftarrow \mathcal{A}(\mathcal{G}, crs_{sim}, td); \phi_{sim} \leftarrow \texttt{GSCRSSimProve}(\mathcal{G}, crs_{sim}, td, x) : \mathcal{A}(\phi_{sim}) = 1]$$

2.5 Ring Signatures - Definitions

Definition 1 (Ring signature): A ring signature scheme is a quadruple of PPT algorithms `RSig := (RSetup, RKeyGen, RSign, RVerify)` which generates public parameters, keys for users, signs message and verifies the validity of signature on a message produced by user.

- **RSetup**(1^κ): It takes security parameter κ as input and produces public parameters $rParam$ as output.
- **RKeyGen**$(rParam)$: Given $rParam$ as input, it produces a secret key SK and a public key PK.
- **RSign**$(m, SK_s, \mathcal{R})$: Given a private signing key SK_s as input, it outputs a signature Σ on message m with respect to a ring $\mathcal{R} := (PK_1, PK_2, ..., PK_n)$. We require that $PK_s \in \mathcal{R}$ for $s \in [1, n]$ where $n \in \mathbb{N}$ and (PK_s, SK_s) is a valid key pair.
- **RVerify**$(m, \Sigma, \mathcal{R})$: It verifies a signature Σ on message m with respect to a set of public keys in $\mathcal{R}$ and outputs 1 if succeeds, otherwise 0.

The quadruple `(RSetup, RKeyGen, RSign, RVerify)` is a secure ring signature if it satisfies the following properties from the literature [2, 8].

Definition 2 (Correctness)[2]: A ring signature `(RSetup, RKeyGen, RSign, RVerify)` has correctness if for any polynomial $p(.)$, set of secret-public key pairs $\{(PK_i, SK_i)\}_{i=1}^{p(\kappa)}$ generated by `RKeyGen(.)`, any message m and any index $j \in [1, p(\kappa)]$, the signature Σ produced by `RSign`$(m, SK_j, \mathcal{R})$ will be accepted by `RVerify`$(m, \Sigma, \mathcal{R})$. Here $\mathcal{R} = (PK_1, PK_2...PK_{p(\kappa)})$

Definition 3 (Anonymity against full key exposure)[2]: A ring signature scheme `(RSetup, RKeyGen, RSign, RVerify)` achieves anonymity (with respect to full key exposure) if for any adversary $\mathcal{A}$ and for any polynomial $p(.)$, the probability that $\mathcal{A}$ succeeds in the following game is negligibly close to $1/2$:

1. Key pairs $(PK_i, SK_i)_{i=1}^{p(\kappa)}$ are generated by the challenger using $rParam \leftarrow$ `RSetup`(1^κ) and `RKeyGen`$(rParam, \omega_i)$ for randomly chosen ω_i. $\mathcal{A}$ is given $S := (PK_i)_{i=1}^{p(\kappa)}$.
2. Throughout the game, $\mathcal{A}$ has access to a signing Oracle `Osign(.,.,.)`, where `Osign`$(s,m,\mathcal{R})$ outputs `RSign`$(m, SK_s, \mathcal{R})$ and we require $PK_s \in \mathcal{R}$ and $s \in [1, p(\kappa)]$.
3. $\mathcal{A}$ is also given access to the corrupt oracle `Corrupt(.)` , where `Corrupt`(i) outputs ω_i.
4. $\mathcal{A}$ outputs a message m, two distinct indices i_0 and i_1 and a ring $\mathcal{R}$ with the only condition that $PK_{i_0}, PK_{i_1} \in \mathcal{R}$. It interacts with the challenger to get a signature $\Sigma \leftarrow$ `RSign`$(m, SK_{i_b}, \mathcal{R})$ where $b \leftarrow \{0, 1\}$.

5. $\mathcal{A}$ outputs a bit b' and succeeds if $b = b'$.

Definition 4 (Unforgeability with respect to insider corruption)[2]: A ring signature scheme `(RSetup, RKeyGen, RSign, RVerify)` is unforgeable (with respect to insider corruption) if for any adversary $\mathcal{A}$ and for any polynomial $p(.)$, the probability that $\mathcal{A}$ succeeds in the following game is negligible:

1. Key pairs $(PK_i, SK_i)_{i=1}^{p(\kappa)}$ are generated by the challenger using `RKeyGen(`$rParam$`)`. $\mathcal{A}$ is given $S := (PK_i)_{i=1}^{p(\kappa)}$
2. Throughout the game, $\mathcal{A}$ has access to a signing Oracle `Osign(.,.,.)`, where `Osign`$(s,m,\mathcal{R})$ outputs `RSign`$(m, SK_s, \mathcal{R})$ and we require $PK_s \in \mathcal{R}$.
3. $\mathcal{A}$ is also given access to the corrupt oracle `Corrupt(.)` , where `Corrupt`(i) outputs SK_i.
4. $\mathcal{A}$ outputs $(\mathcal{R}^\star, m^\star, \Sigma^\star)$ and succeeds if `RVerify`$(m^\star, \Sigma^\star, \mathcal{R}^\star) = 1$, $\mathcal{A}$ never queried $(\star, m^\star, \mathcal{R}^\star)$ and $\mathcal{R}^\star \subseteq S - C$, where C is the set of corrupted users.

2.6 Polynomial Commitments

Polynomial commitment means committing to a polynomial with a short string used by some verifier to confirm the claimed evaluations of the committed polynomial. Let us consider committing a polynomial $F(x) \in \mathbb{Z}_p[x]$ with degree t and coefficients $(f_0, f_1, ..., f_t)$. One way of committing it is to commit to the string $(f_0||f_1||...||f_t)$ or any other equivalent representation of $F(x)$. However, this is not suitable for many cryptographic applications, since opening commitment will reveal the entire polynomial. Another solution could be to commit to the coefficients e.g., $C = (g^{f_1}, g^{f_2}, ..., g^{f_t})$. In that case commitment size becomes $t+1$ group elements. All of the above problems can be overcome by an efficient polynomial commitment scheme found in the literature [23]. The main idea of the paper [23] was to efficiently commit the polynomial $F(x)$ over a bilinear pairing group with two different types of commitment schemes $\text{PolyCommit}_{\text{DL}}$ and $\text{PolyCommit}_{\text{Ped}}$ based on discrete log and Pedersen commitments. It ensures the size of the commitment to be constant, a single group element.

An immediate application of the above polynomial commitment scheme is Zero Knowledge Sets(ZKS) [26]. In short, a ZKS allows a committer to create a short commitment to the set of values contained in S and later prove statements like $s_j \in S$ or $s_j \notin S$ without revealing S or the upper bound of $|S|$. Another relaxation of ZKS is nearly ZKS where the information about the size of the set can be revealed and this is more suitable for practical applications. Kate *et al.*[23] have given an efficient application of polynomial commitments in nearly ZKS and have shown that the membership or non-membership of an element in a particular set is Zero-Knowledge. It also achieves constant communication complexity. Our primary idea of a constant size membership proof of an element in a public set is derived from the idea above.

3 Constant Size Set Membership Proof

We provide a non-interactive, constant-sized set membership proof technique based on the application of polynomial commitment scheme in ZKS [23] and Groth-Sahai NIZK proof system [16, 17]. The technique allows a prover to prove the containment of an element α_δ in a set $S = \{\alpha_1, \alpha_2, ..., \alpha_\delta, ...\alpha_N\} \in \mathbb{Z}_n^N$.

The sub-linear size membership proof in [14] arranges the set elements in the form of a square matrix. Each element of a vertical and horizontal bit vector is committed using Groth-Sahai (GS) scheme resulting in an expensive sub-linear blowup both in number of proofs and commitment components. GS proofs for $O(\sqrt{|S|})$ number of MSME and QE equations as well as GS commitments contribute to the final size of the proof.

Our formulation constructs a polynomial $F(x)$ having only $\alpha_i \in S, \forall i \in [1, |S|]$ as roots. The prover aims to demonstrate the existence of *some* secret value $\alpha_\delta \in S$ to the verifier in a non-interactive manner without revealing its value. While correctness of the proof system follows from the construction itself, soundness relies on the hardness of q-SDH assumption. Verification equations are in the form of Pairing Product Equation (PPE) and Multi-Scalar Multiplication Equation (MSME) respectively as defined by GS framework. Variables which could potentially leak out the secret value α_δ are committed using GS commitment scheme to provide required zero-knowledgeness. Our proof works both in prime and composite order groups. However, to be able to fit it as-is in our ring signature scheme, we use asymmetric pairing of composite order in the presentation.

Our membership proof consists of following four algorithms:

– **MemSetup**$(1^\kappa, q)$: This algorithm is run by a trusted authority (possibly a distributed one to enhance security) to generate required q-SDH tuple and initialize GS protocol.

- Initialize GS protocol $\langle \mathcal{G}, sk \rangle \leftarrow \texttt{GSSetup}(1^\kappa)$ in symmetric setting of composite order $n = p.q$ where $sk = \langle p, q \rangle$, $\mathcal{G} = \langle n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2 \rangle$ and q-SDH assumption holds in $\mathbb{G}_1$. p and q are large prime numbers.
- Generate common reference string $(crs, xk) \leftarrow \texttt{GSCRSGen}(\mathcal{G}, sk)$
- Choose a secret key $\beta \in_R \mathbb{Z}_n^*$
- Generate a $(q+1)$ tuple $qSDH = \langle g_1, g_1^{\beta}, g_1^{\beta^2}, ..., g_1^{\beta^q} \rangle \in \mathbb{G}_1^{q+1}$ to accommodate a set of cardinality $\leq q$. Note that the secret key is not needed any more and can be discarded.
- Publish public parameters $mParam = \langle \mathcal{G}, crs, qSDH, g_2^{\beta} \rangle$

– **MemWitness**$(mParam, \alpha_\delta, S)$: This algorithm is run by the prover to generate witness W testifying the presence of $\alpha_\delta \in S$.

- Compute the polynomial $F(x) = \prod_{i=1}^{|S|}(x - \alpha_i) = \sum_{j=0}^{|S|} F_j x^j, \alpha_i \in S$
- Compute the polynomial $\psi(x) = \frac{F(x)}{(x-\alpha_\delta)} = \sum_{i=0}^{|S|-1} \psi_i x^i$
- Compute $w = g_1^{\psi(\beta)} = \prod_{i=0}^{|S|-1}(g_1^{\beta^i})^{\psi_i}$. Note that the components of the form $g_1^{\beta^i}$ are available to the prover as part of $qSDH$ tuple.
- Compute $D = g_2^{\alpha_\delta}$
- Output the tuple $W = \langle \alpha_\delta, w, D \rangle$ as witness.

– **MemProve**$(mParam, S, W)$: This algorithm is run by the prover to generate commitments for variables and GS proofs for verification equations.

- Compute $C = g_1^{F(\beta)} = \prod_{i=0}^{|S|}(g_1^{\beta^i})^{F_i}$. Note that the components of the form $g_1^{\beta^i}$ are available to the prover as part of $qSDH$ tuple.
- Compute $t = e(C, g_2)$
- Compute the membership proof $\phi_{mem} = \langle \{\Upsilon_w, \Upsilon_{\alpha_\delta}, \Upsilon_D\}, \boldsymbol{\Gamma}_{mem} \rangle$

$$\phi_{mem} \leftarrow \texttt{GSProve}\{\mathcal{G}, crs, \{e(\underline{w}, g_2^{\beta}/\underline{D}) = t \wedge \underline{D} = g_2^{\alpha_\delta}\}, (\alpha_\delta, w, D)\}$$

- Send the proof ϕ_{mem} to the verifier.

– **MemVerify**$(mParam, S, \phi_{mem})$: This algorithm is run by the verifier to verify the presence of the element α_δ chosen by the prover in the set S.

- Compute $F(x), C$ and t in the same way as the prover did from publicly available information.
- Compute $c \leftarrow \texttt{GSVerify}\{\mathcal{G}, crs, \{e(\underline{w}, g_2^{\beta}/\underline{D}) = t \wedge \underline{D} = g_2^{\alpha_\delta}\}, \phi_{mem}\}$
- Announce 'Success' if $c = 1$, 'Failure' otherwise

Theorem 1. *The set membership proof technique is correct, perfectly sound and zero-knowledge.*

Proof: Detailed proof of the theorem above is presented in Appendix C.

Complexity of the membership proof: We present the cost of membership proof and associated commitments of our construction in terms of group elements of $\mathbb{G}_1$ and $\mathbb{G}_2$ in Table 2. Detailed calculation of the size of the membership proofs are given in Appendix A.

Table 2. Cost of our membership proofs

Components | Instantiations	$\text{DLIN}_{\mathbb{G}}$	$\text{DDH}_{\mathbb{G}_1}$ + $\text{DLIN}_{\mathbb{G}_2}$		$\text{SXDH}_{\mathbb{G}_1,\mathbb{G}_2}$	
	$\mathbb{G}$	$\mathbb{G}_1$	$\mathbb{G}_2$	$\mathbb{G}_1$	$\mathbb{G}_2$
GS Commitments	9	4	3	4	2
GS Proofs	18	8	6	8	6
Membership Proofs	27	21		20	

4 Generic Construction of Ring Signature

We now present a construction of constant size ring signature scheme based on our membership proof outlined in section 3. Signature scheme is a four algorithm protocol `Sig := (SSetup, SKeyGen, SSign, SVerify)`. Let, $\mathcal{G}$ be a bilinear group, $\mathcal{M}$ be the message space, $\langle SK_i = \{sk_{i1}, sk_{i2}, ..., sk_{iM}\}, PK_i = \{pk_{i1}, pk_{i2}, ..., pk_{iN}\}\rangle$ be the secret and public keys of the signer i with respect to the signature scheme `Sig`. $\Delta = \langle \Delta_1, \Delta_2, ..., \Delta_n\rangle$ be the signature on message $m \in \mathcal{M}$ and $\mathcal{R}$ be the ring. The signature scheme must be a 'compatible' one satisfying the following characteristics:

- Our construction requires $\mathcal{G} = \langle n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2\rangle$ to be a bilinear group of composite order in asymmetric setting [24] to be able to instantiate GS commitment scheme under SXDH assumption. Apparently for composite order groups, SGH instantiation of GS scheme in symmetric setting could have been the most obvious choice. But, the reason such a construction wouldn't work in our case is GS commitments are not fully extractable in this setting as we require in the unforgeability game. Moreover, proofs are L_{co} sound rather than being perfectly sound in this case.
- q-SDH assumption must be hard in $\mathbb{G}_1$ for our constant size set membership proof to be plugged-in.
- Secret key $SK_i \in \mathbb{Z}_n^M$ and public key $PK_i \in \mathbb{G}_k^N,\ k \in [1, 2]$
- Verification equations of the scheme must be MSME or PPE committable in GS framework.
- We commit signature components in GS framework. Signature component Δ_i that depends on SK_i must be a group element of $\mathbb{G}_1$ or $\mathbb{G}_2$ committable in GS framework. Unless committed, an adversary may trivially break the signature anonymity by test verifying Δ_i with $\forall PK_j \in \mathcal{R}$. Extractability is important to demonstrate the impossibility of forgery by the challenger in the unforgeability game.

Our ring signature construction is as follows:

- **RSetup**$(1^\kappa, q)$: This algorithm is run by a trusted authority (possibly a distributed one to enhance security). GS proof system is instantiated and a suitable hash function is chosen to associate message to ring information.
  - $mParam \leftarrow$ `MemSetup`$(1^\kappa, q)$. Parse $mParam$ as public parameters $\langle \mathcal{G}, crs, qSDH, g_2^\beta\rangle$ of the constant size membership proof technique as outlined in section 3. Parse $\mathcal{G}$ as a description of a bilinear group $\langle n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2\rangle$ of composite order $n = p.q$ where q-SDH assumption holds in $\mathbb{G}_1$. p and q are large prime numbers.
  - Choose a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{M}$ used to map the concatenation of some pre-agreed representation of message m and $\mathcal{R}$ to $\mathcal{M}$
  - $sParam \leftarrow$ `SSetup`(1^κ)
  - Publish public parameters $rParam = \langle sParam, mParam, \mathcal{H}\rangle$
- **RKeyGen**$(rParam)$: Key generation protocol is assumed to have run by each of the prospective ring members to generate their own secret-public key pairs.
  - Generate key-pair $(SK_i, PK_i) \leftarrow$ `SKeyGen`$(sParam, \mathcal{G})$ for all prospective ring member $i \in \mathcal{R}$
  - Extend the public key of the signature scheme by computing public integers $q_{ij} = sk_{ij}^2 \pmod n \in \mathbb{Z}_n, \forall sk_{ij} \in SK_i$ for all prospective ring member $i \in \mathcal{R}$. Going ahead, these components will help us in showing the correlation between the public key PK_i and ring $\mathcal{R}$. Augment the public key PK_i by including $q_i = \{q_{ij}\},\ j \in [1, M]$. $PK_i' = \langle PK_i, q_{i1}, q_{i2}, ..., q_{iM}\rangle$ acts as the extended public key for signer i of our signature algorithm.

- Publish extended public keys $\{PK'_i\}$ to the world.

– **RSign**$(m, SK_s, rParam)$: This algorithm is run by the *real* signer s having key-pair (SK_s, PK_s). A ring is an M-tuple $\mathcal{R} = \langle \mathcal{R}_1, \mathcal{R}_2, ..., \mathcal{R}_M \rangle$. $\mathcal{R}$ is formed by s choosing a set of k *potential* signers, which includes himself. There are as many ring components as q-components present in extended public key. We denote j-th ring component as $\mathcal{R}_j = \{q_{1j}, q_{2j}, ...q_{sj}, ..., q_{kj}\}$. Message m and ring information $\mathcal{R}$ are made available for public verification.

- Associate the message to the ring by computing $m' \leftarrow \mathcal{H}(m||\mathcal{R})$, $m \in \{0,1\}^*$
- Signer s runs $\Delta \leftarrow \texttt{SSign}(m', SK_s, sParam)$
- Generate GS proofs for the signature:

$$\phi_{sig} \leftarrow \texttt{GSProve}(\mathcal{G}, crs, \{\texttt{SVerify}(\underline{PK_s}, m', \underline{\Delta}) = 1\}, (PK_s, \Delta'))$$

- Compute witnesses:

$$W_j = \texttt{MemWitness}(mParam, \underline{q_{sj}}, \mathcal{R}_j), \forall j \in [1, M]$$

  Define $W = \{W_j\}, \forall j \in [1, M]$

- Generate the membership proof:

$$\phi_{mem_j} \leftarrow \texttt{MemProve}(mParam, \mathcal{R}_j, W_j), \forall j \in [1, M]$$

  Define $\phi_{mem} = \{\phi_{mem_j}\}, \forall j \in [1, M]$.

- Generate proofs of correlation between q_{sj} and sk_{sj}:

$$\phi_{q_j} \leftarrow \texttt{GSProve}(\mathcal{G}, crs, \{\underline{q_{sj}} = \underline{sk_{sj}}^2\}, (q_{sj}, sk_{sj})), \forall j \in [1, M]$$

  Define $\phi_q = \{\phi_{q_j}\}, \forall j \in [1, M]$

- Generate proofs of correlation between $SK'_{si} \subseteq SK_s$ and PK_s where $pk_{si} = f_{pk_i}(SK'_{si}), \bigcup_i SK'_{si} = SK_s$:

$$\phi_{pk_i} \leftarrow \texttt{GSProve}(\mathcal{G}, crs, \{\underline{pk_{si}} = f_{pk_i}(\underline{SK'_{si}})\}, (pk_{si}, SK'_{si})), \forall pk_{si} \in PK_s$$

  Define $\phi_{pk} = \{\phi_{pk_i}\}, \forall pk_{si} \in PK_s$. Note that computing $f^{-1}_{pk_i}$ is always hard.

- Publish message m, ring signature $\Sigma \leftarrow (\phi_{sig}, \phi_{mem}, \phi_q, \phi_{pk}, \Delta \backslash \Delta')$ and ring information $\mathcal{R}$

– **RVerify**$(m, \mathcal{R}, \Sigma, rParam)$: The verifier runs this algorithm to verify the validity of the ring signature on the message with respect to the ring information published.

- Let $VE = \{VE_i\}$ be the set of verification equations of the signature scheme `Sig`. Verify the consistency of the signature,

$$c_{sig} \leftarrow \texttt{GSVerify}(\mathcal{G}, crs, \{VE\}, \phi_{sig})$$

- Verify the membership of each q-component of signer's extended public key in the ring,

$$c_{mem_i} \leftarrow \texttt{MemVerify}(mParam, \mathcal{R}, \phi_{mem_i}), \forall \phi_{mem_i} \in \phi_{mem}$$

- Verify proofs of correlation between q_{sj} and sk_{sj}:

$$c_{q_j} \leftarrow \texttt{GSVerify}(\mathcal{G}, crs, \{\underline{q_{sj}} = \underline{sk_{sj}}^2\}, \phi_{q_j}), \forall j \in M$$

- Verify proofs of correlation between $SK'_{si} \subseteq SK_s$ and PK_s where $pk_{si} = f_{pk}(SK'_{si}), \bigcup_i SK'_{si} = SK_s$:

$$c_{pk_i} \leftarrow \texttt{GSVerify}(\mathcal{G}, crs, \{\underline{pk_{si}} = f_{pk}(\underline{SK'_s})\}, \phi_{pk_i}), \forall pk_{si} \in PK_s$$

- Announce 'Success' if $(c_{sig} \wedge (\bigwedge_i c_{mem_i}) \wedge (\bigwedge_i c_{q_i}) \wedge (\bigwedge_i c_{pk_i})) = 1$, 'Failure' otherwise

The idea of the proof system above is to show correlation among commitments shared across equations. If a witness is involved in more than one equation, prover must use the same set of randomness to commit to that particular witness throughout. On the other hand, verifier re-uses the same commitment during verification which makes the proofs correlated. Intuitively, public key $PK_s \xrightarrow{\phi_{pk}} SK_s \xrightarrow{\phi_q} q_s \xrightarrow{\phi_{mem}}$ ring $\mathcal{R}$

Theorem 2. *The generic construction of ring signature scheme outlined above is a secure one satisfying correctness, anonymity and unforgeability.*

Proof: Detailed proof of the theorem above is presented in Appendix D.

5 Instantiation Based on Full Boneh-Boyen Signature (FBB) Scheme

To quantify the reduction in size our construction offers, we will now pick up FBB scheme for a concrete instantiation. [3] proves the security of the signature scheme for prime order groups. Their proof translates directly to composite order model as shown by [8]. Our construction is in asymmetric setting over composite order group [24] where both q-SDH [9] and SXDH assumptions hold.

- **RSetup**$(1^\kappa, q)$: This algorithm is run by a trusted authority (possibly a distributed one to enhance security).
  - $mParam \leftarrow \texttt{MemSetup}(1^\kappa, q)$. Parse $mParam$ as public parameters $\langle \mathcal{G}, crs, qSDH, g_2^\beta \rangle$ of the constant size membership proof technique as outlined in section 3. Parse $\mathcal{G}$ as a description of a bilinear group $\langle n, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2 \rangle$ of composite order $n = p.q$ where q-SDH assumption holds in $\mathbb{G}_1$. p and q are large prime numbers.
  - Choose a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{Z}_n$
  - Publish public parameters $rParam \leftarrow \langle mParam, \mathcal{H} \rangle$
- **RKeyGen**$(rParam)$: Key generation protocol is assumed to have run by each of the prospective ring member $i \in \mathcal{R}$ to generate their own secret-public key pairs.
  - Uniformly choose secret key $SK_i = \langle a_i, b_i \rangle \in \mathbb{Z}_n^2$
  - Generate FBB public key $PK_i = \langle A_i, B_i \rangle = \langle g_2^{a_i}, g_2^{b_i} \rangle$. Compute $q_{ia} = a_i^2 \pmod n$ and $q_{ib} = b_i^2 \pmod n$. Extended public key $PK_i' = \langle PK_i, q_{ia}, q_{ib} \rangle$
  - Publish extended public keys $\{PK_i'\}$ to the outer world.
- **RSign**$(m, SK_s, rParam)$: This algorithm is run by the *real* signer s having key-pair $(SK_s = \langle a_s, b_s \rangle, PK_s = \langle A_s, B_s \rangle)$. Choose k potential signers to construct a ring $\mathcal{R} = \{\mathcal{R}_a, \mathcal{R}_b\}$. Message m and ring information $\mathcal{R}$ are made available for public verification. Rename $a = a_s, b = b_s, A = A_s, B = B_s$
  - Compute $m' \leftarrow \mathcal{H}(m||\mathcal{R})$, $m \in \{0,1\}^*$
  - Uniformly choose $r \leftarrow \mathbb{Z}_n \setminus \{\frac{-a+m'}{b}\}$
  - Generate the signature $\Delta \leftarrow g_1^{\frac{1}{a+r.b+m'}}$
  - Generate GS proofs for the signature:

$$\phi_{sig} \leftarrow \texttt{GSProve}(\mathcal{G}, crs, \{\underline{B}^r = \underline{B}' \wedge e(\underline{\Delta}, \underline{A})e(\underline{\Delta}, \underline{B}')e(\underline{\Delta}, g_2^{m'}) = e(g_1, g_2)\}, (\Delta, A, B, B'))$$

  - Compute witnesses $W = \langle W_a, W_b \rangle$:

$$W_a \leftarrow \texttt{MemWitness}(mParam, \underline{q_{sa}}, \mathcal{R}_a)$$
$$W_b \leftarrow \texttt{MemWitness}(mParam, \underline{q_{sb}}, \mathcal{R}_b)$$

- Generate the membership proof $\phi_{mem} = \langle \phi_{mem_a}, \phi_{mem_b} \rangle$:

$$\phi_{mem_a} \leftarrow \texttt{MemProve}(mParam, \mathcal{R}_a, W_a)$$
$$\phi_{mem_b} \leftarrow \texttt{MemProve}(mParam, \mathcal{R}_b, W_b)$$

- Generate proofs of correlation $\phi_q = \langle \phi_{q_a}, \phi_{q_b} \rangle$ between $q_s = \langle q_{sa}, q_{sb} \rangle$ and $SK_s = \langle a, b \rangle$:

$$\phi_{q_a} \leftarrow \texttt{GSProve}(\mathcal{G}, crs, \{\underline{q_{sa}} = \underline{a}^2\}, (q_{sa}, a))$$
$$\phi_{q_b} \leftarrow \texttt{GSProve}(\mathcal{G}, crs, \{\underline{q_{sb}} = \underline{b}^2\}, (q_{sb}, b))$$

- Generate proofs of correlation $\phi_{pk} = \langle \phi_{pk_A}, \phi_{pk_B} \rangle$ between SK_s and PK_s:

$$\phi_{pk_A} \leftarrow \texttt{GSProve}(\mathcal{G}, crs, \{\underline{A} = g_2^{\underline{a}}\}, (A, a))$$
$$\phi_{pk_B} \leftarrow \texttt{GSProve}(\mathcal{G}, crs, \{\underline{B} = g_2^{\underline{b}}\}, (B, b))$$

- Publish message m, ring signature $\Sigma \leftarrow (\phi_{sig}, \phi_{mem}, \phi_q, \phi_{pk}, \Delta \backslash \Delta', r)$ and ring information $\mathcal{R}$

– **RVerify**$(m, \Sigma, \mathcal{R})$: The verifier runs this algorithm to verify the validity of the ring signature on the message with respect to the ring information published.

- Verify the consistency of the signature

$$c_{sig} \leftarrow \texttt{GSVerify}(\mathcal{G}, crs, \{\underline{B}^r = \underline{B}' \wedge e(\underline{\Delta}, \underline{A}) e(\underline{\Delta}, \underline{B}') e(\underline{\Delta}, g_2^{m'}) = e(g_1, g_2)\}, \phi_{sig})$$

- Verify the membership of each q-component of signer's extended public key in the ring,

$$c_{mem_a} \leftarrow \texttt{MemVerify}(mParam, \mathcal{R}_a, \phi_{mem_a})$$
$$c_{mem_b} \leftarrow \texttt{MemVerify}(mParam, \mathcal{R}_b, \phi_{mem_b})$$

- Verify proofs of correlation between q_s and SK_s:

$$c_{q_a} \leftarrow \texttt{GSVerify}(\mathcal{G}, crs, \{\underline{q_{sa}} = \underline{a}^2\}, \phi_{q_a})$$
$$c_{q_b} \leftarrow \texttt{GSVerify}(\mathcal{G}, crs, \{\underline{q_{sb}} = \underline{b}^2\}, \phi_{q_b})$$

- Verify proofs of correlation between SK_s and PK_s:

$$c_{pk_A} \leftarrow \texttt{GSVerify}(\mathcal{G}, crs, \{\underline{A} = g_2^{\underline{a}}\}, \phi_{pk_A})$$
$$c_{pk_B} \leftarrow \texttt{GSVerify}(\mathcal{G}, crs, \{\underline{B} = g_2^{\underline{b}}\}, \phi_{pk_B})$$

- Announce 'Success' if $(c_{sig} \wedge (c_{mem_a} \wedge c_{mem_b}) \wedge (c_{q_a} \wedge c_{q_b}) \wedge (c_{pk_A} \wedge c_{pk_B})) = 1$, 'Failure' otherwise

Theorem 3. *The construction of ring signature scheme outlined above is a secure conditioned if SXDH assumption holds, q-SDH assumption hold in $\mathbb{G}_1$ and the hash function $\mathcal{H}$ is collision-resistant.*

Proof: The proof for the theorem above follows directly from Theorem 2 and the security proof for FBB scheme [3].

Complexity of the signature instantiation: Under SXDH instantiation of GS proof system, size of our ring signature construction based on FBB signature scheme is $\mathbb{G}_1^{50} + \mathbb{G}_2^{42} + \mathbb{Z}_p^3$ elements. Detailed calculation of the size of the signature elements is given in Appendix B.

6 Conclusion and Open Problems

Our main contribution is a construction of an $O(1)$ size ring signature scheme based on FBB signatures without using random oracle. We introduce a general purpose constant size membership proof technique which 'plugs-in' to a compatible signature scheme to yield a ring signature of constant size.

It would be interesting to explore the possibility of applying the generic construction to signature schemes other than FBB to achieve the same objective. Extending the ring signature protocol to obtain a constant-sized blind ring signature is also a problem which we leave open for further research.

References

[1] Abe, M., Ohkubo, M., and Suzuki, K. "1-out-of-n Signatures from a Variety of Keys". In: *Advances in Cryptology - ASIACRYPT'02*, Springer LNCS 2501. 2002, pp. 415–432.

[2] Bender, A., Katz, J., and Morselli, R. "Ring signatures: Stronger definitions, and constructions without random oracles". In: *Halevi, S., Rabin, T. (eds.) TCC*, Springer Heidelberg LNCS 3876. 2006, pp. 60–79.

[3] Boneh, D. and Boyen, X. "Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups". In: *Journal of Cryptology*, 21(2). 2008, pp. 149–177.

[4] Boneh, D. et al. "Aggregate and verifiably encrypted signatures from bilinear maps". In: *Advances in Cryptology EUROCRYPT'03*, Springer LNCS 2656. 2003, pp. 416–432.

[5] Boneh, Dan, Shen, Emily, and Waters, Brent. "Strongly unforgeable signatures based on computational Diffie-Hellman". In: *Proceedings of PKC'06*. Springer Verlag, 2006, pp. 229–240.

[6] Boyen, X. "Mesh Signatures". In: *Advances in Cryptology EUROCRYPT'07*, Springer LNCS 4515. 2007, pp. 210–227.

[7] Brakerski, Z. and Kalai, Y.T. "A Framework for Efficient Signatures, Ring Signatures and Identity Based Encryption in the Standard Model". In: *Cryptology ePrint Archive, Report 2010/086*. URL: `http://eprint.iacr.org/2010/086.pdf`.

[8] Chandran, N., Groth, J., and Sahai, A. "Ring Signatures of Sub-linear Size Without Random Oracles". In: *ICALP'07*, Springer LNCS 4596. 2007, pp. 423–434.

[9] Chase, Melissa and Meiklejohn, Sarah. "Déjà Q: Using Dual Systems to Revisit q -Type Assumptions". In: *Proceedings of Eurocrypt'06*. Springer Verlag, 2014.

[10] Chaum, David. and Van Heyst, Eug'ene. "Group signatures". In: *Proceedings of Advances in Cryptology - EUROCRYPT'91*. Ed. by Davies, D.W., Springer LNCS 547. Berlin, 1991, pp. 257–265.

[11] Chow, Sherman S. M. et al. "Ring Signatures without Random Oracles". In: *ASIACCS'06: Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ACM Press. New York, NY, USA, 2006, pp. 297–302.

[12] Dodis, Y. et al. "Anonymous identification in ad hoc groups". In: *Proceedings of EUROCRYPT'04*. Ed. by Cachin, C. and Camenisch, J.L., Springer LNCS 3027. 2004, pp. 609–626.

[13] Galbraith, Steven D., Paterson, Kenneth G., and Smart, Nigel P. "Pairings for cryptographers". In: *Discrete Applied Mathematics*, 156(16). 2008, pp. 3113–3121.

[14] Ghadafi, E. "Sub-linear Blind Ring Signatures without Random Oracles". In: *Proceedings of IMACC '13*, Springer LNCS 8308. 2013, pp. 304–323.

[15] Goldwasser, Shafi, Micali, Silvio, and Rivest, Ronald L. "A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks". In: 1995.

[16] Groth, J. and Sahai, A. "Efficient Non-interactive Proof Systems for Bilinear Groups". In: *Advances in Cryptology – EUROCRYPT'08*, Springer LNCS 4965. 2008, pp. 415–432.

[17] Groth, J. and Sahai, A. "Efficient Non-interactive Proof Systems for Bilinear Groups (Full Version)". In: URL: `https://eprint.iacr.org/2007/155.pdf`.

[18] Groth, Jens. "Simulation-sound NIZK proofs for a practical language and constant size group signatures". In: *Proceedings of ASIACRYPT'06*, Springer LNCS 4248. 2006, pp. 444–459.

[19] Groth, Jens., Ostrovsky, Rafail., and Sahai, Amit. "Perfect non-interactive zero-knowledge for NP". In: *Proceedings of EUROCRYPT'06*, Springer LNCS 4004. 2006, pp. 339–358.

[20] Herranz, J. and Sez., G. "Forking Lemmas for Ring Signature Schemes". In: *Progress in Cryptology INDOCRYPT'03*, Springer LNCS 2904. 2003, pp. 266–279.

[21] Herzberg, A. et al. "Proactive Secret Sharing Or: How to Cope With Perpetual Leakage". In: *Proceedings of CRYPTO'95*, Springer LNCS 963. 1995, pp. 339–352.

[22] Jakobsson, M., Sako, K., and Impagliazzo, R. "Designated verifier proofs and their applications". In: *Proceedings of EUROCRYPT'96*. Ed. by Maurer, U.M., Springer LNCS 1070. 1996, pp. 143–154.

[23] Kate, A., Zaverucha, G. M., and Goldberg, I. "Constant-Size Commitments to Polynomials and Their Applications". In: *Proceedings of ASIACRYPT'10*, Springer LNCS 6477. 2010, pp. 177–194.

[24] Meiklejohn, Sarah and Shacham, Hovav. "New Trapdoor Projection Maps for Composite-Order Bilinear Groups". In: *Cryptology ePrint Archive, Report 2013/657*. URL: `https://eprint.iacr.org/2013/657.pdf`.

[25] Menezes, Alfred J., Vanstone, Scott A., and Oorschot, Paul C. Van. "Handbook of Applied Cryptography". In: Boca Raton, FL, USA: CRC Press, Inc., 1996. ISBN: 0849385237.

[26] Micali, S., Rabin, M., and Kilian, J. "Zero-knowledge sets". In: *Proceedings of FOCS'03*, IEEE. 2003, pp. 80–91.

[27] Naor, M. "Deniable ring authentication". In: *Proceedings of CRYPTO'02*. Ed. by Yung, M., Springer LNCS 2442. 2002, pp. 481–498.
[28] Naor, M. and Reingold, O. "On the construction of pseudo-random permuations: Luby-racko revisited". In: *Proceedings of 29th Symposium on Theory of Computing (STOC)*. 1997, pp. 189–199.
[29] Rivest, R.L., Shamir, A., and Tauman, Y. "How to leak a secret: Theory and applications of ring signatures". In: *ASIACRYPT'01*, Springer LNCS 3895. 2001, pp. 164–186.
[30] Rudnicki, P. "Little Bezout theorem (Factor Theorem)". In: *Formalized Mathematics'04*, 12. 2004.
[31] Schage, S. and Schwenk, J. "A CDH-Based Ring Signature Scheme with Short Signatures and Public Keys". In: *Financial Cryptography'10*, Springer LNCS 6052. 2010, pp. 129–142.
[32] Shacham, H. and Waters, B. "Efficient ring signatures without random oracles". In: *Proceedings of PKC'07*, Springer LNCS 4450. 2007, pp. 166–180.
[33] Smart, Nigel P. and Vercauteren, Frederik. "On computable isomorphisms in efficient asymmetric pairing-based systems". In: *Discrete Applied Mathematics*, 155(4). 2007, pp. 538–547.

7 Appendix A: Cost of Membership Proof

We provide here the detailed break up of the cost of membership proof (in terms of number of group elements involved) below:

Type - 1 ($\mathbf{DLIN}_G$):

Table 3. Cost split up under DLIN assumption[17, p. 28]

Item Type	Item	$\text{Cost}_{\mathbb{G}}$	$\text{Cost}_{\mathbb{Z}_p}$
Commitment	Υ_w	$\mathbb{G}^3$	-
Commitment	Υ_D	$\mathbb{G}^3$	-
Commitment	Υ_{α_δ}	$\mathbb{G}^3$	-
PPE Proof	$\Gamma_{mem_{PPE}}$	$\mathbb{G}^9$	-
MSME Proof	$\Gamma_{mem_{MSME}}$	$\mathbb{G}^9$	-
Total Cost	-	$\mathbb{G}^{27}$	-

Type - 2 ($\mathbf{DDH}_{G_1}$ + $\mathbf{DLIN}_{G_2}$):

Table 4. Cost split up under DDH and DLIN assumption[17, p. 23,28]

Item Type	Item	$\text{Cost}_{\mathbb{G}_1}$ + $\text{Cost}_{\mathbb{G}_2}$		$\text{Cost}_{\mathbb{Z}_p}$
Commitment	Υ_w	$\mathbb{G}_1^2$	-	-
Commitment	Υ_D	-	$\mathbb{G}_2^3$	-
Commitment	Υ_{α_δ}	$\mathbb{G}_1^2$	-	-
PPE Proof	$\Gamma_{mem_{PPE}}$	$\mathbb{G}_1^4$	$\mathbb{G}_2^3$	-
MSME Proof	$\Gamma_{mem_{MSME}}$	$\mathbb{G}_1^4$	$\mathbb{G}_2^3$	-
Total Cost	-	$\mathbb{G}_1^{12} + \mathbb{G}_2^9$		-

Type - 3 (SXDH):

Table 5. Cost split up under SXDH assumption[17, p. 23]

Item Type	Item	$\text{Cost}_{\mathbb{G}_1}$ + $\text{Cost}_{\mathbb{G}_2}$		$\text{Cost}_{\mathbb{Z}_p}$
Commitment	Υ_w	$\mathbb{G}_1^2$	-	-
Commitment	Υ_D	-	$\mathbb{G}_2^2$	-
Commitment	Υ_{α_δ}	$\mathbb{G}_1^2$	-	-
PPE Proof	$\Gamma_{mem_{PPE}}$	$\mathbb{G}_1^4$	$\mathbb{G}_2^4$	-
MSME Proof	$\Gamma_{mem_{MSME}}$	$\mathbb{G}_1^4$	$\mathbb{G}_2^2$	-
Total Cost	-	$\mathbb{G}_1^{12} + \mathbb{G}_2^8$		-

8 Appendix B: Cost of FBB Instantiation

We provide cost split up of FBB (Full Boneh - Boyen) instantiation of ring signature (in terms of number of group elements involved) below:

Type - 3 (SXDH):

Table 6. Cost split up under SXDH assumption[17, p. 23,28]

Item Type	Item	$\text{Cost}_{\mathbb{G}_1}$ + $\text{Cost}_{\mathbb{G}_2}$		$\text{Cost}_{\mathbb{Z}_p}$
Commitment	Υ_Δ	$\mathbb{G}_1^2$	-	-
Commitment	Υ_A	-	$\mathbb{G}_2^2$	-
Commitment	Υ_B	-	$\mathbb{G}_2^2$	-
Commitment	$\Upsilon_{B'}$	-	$\mathbb{G}_2^2$	-
Commitment	Υ_a	$\mathbb{G}_1^2$	$\mathbb{G}_2^2$	-
Commitment	Υ_b	$\mathbb{G}_1^2$	$\mathbb{G}_2^2$	-
PPE Proof	$\Gamma_{sig_{PPE}}$	$\mathbb{G}_1^4$	$\mathbb{G}_2^4$	-
Linear MSME($\mathbb{G}_2$)	$\Gamma_{sig_{MSME}}$	-	-	$\mathbb{Z}_p^2$
Constant	r	-	-	$\mathbb{Z}_p^1$
Membership Proof	ϕ_{mem_a}	$\mathbb{G}_1^{12}$	$\mathbb{G}_2^8$	-
Membership Proof	ϕ_{mem_b}	$\mathbb{G}_1^{12}$	$\mathbb{G}_2^8$	-
Correlation Proof	Γ_{q_a}	$\mathbb{G}_1^2$	$\mathbb{G}_2^2$	-
Correlation Proof	Γ_{q_b}	$\mathbb{G}_1^2$	$\mathbb{G}_2^2$	-
QE Equality Proof*	-	$\mathbb{G}_1^2$	$\mathbb{G}_2^2$	-
QE Equality Proof*	-	$\mathbb{G}_1^2$	$\mathbb{G}_2^2$	-
Correlation Proof	Γ_{pk_A}	$\mathbb{G}_1^4$	$\mathbb{G}_2^2$	-
Correlation Proof	Γ_{pk_B}	$\mathbb{G}_1^4$	$\mathbb{G}_2^2$	-
Total Cost	-	$\mathbb{G}_1^{50} + \mathbb{G}_2^{42} + \mathbb{Z}_p^3$		

*We considered correlation QEs to be of the form $x_1y_1 - x_2 = 0$ having $\langle\{a, a\}, q_{sa}\rangle$ and $\langle\{b, b\}, q_{sb}\rangle$ as witnesses. To deal with only one group of variables in $\mathbb{Z}_n$, we implicitly added equations of the form $x_1 = y_1$

9 Appendix C: Security of Membership Proof

Theorem 1. *The set membership proof technique is correct, perfectly sound and zero-knowledge.*

Proof: Correctness, perfect soundness and zero-knowledgeness of the set membership proof technique follow from Lemma 1, 2 and 3 respectively.

Lemma 1. *If GS proof system is perfectly complete, the set membership proof technique is correct.*

Proof: The set membership proof technique is correct if an honest prover committing to an element α_δ can convince an honest verifier whenever the element belongs to the set and the prover holds a witness

tuple $W = \langle \alpha_\delta, w, D \rangle$ testifying to the fact.

$$Pr[mParam \leftarrow \texttt{MemSetup}(1^\kappa, q) \ \wedge \ W \leftarrow \texttt{MemWitness}(mParam, \alpha_\delta, S) \wedge \\ \phi_{mem} \leftarrow \texttt{MemProve}(mParam, S, W) \ \wedge \ \texttt{MemVerify}(mParam, S, \phi_{mem}) = 1, \text{if } \alpha_\delta \in S] = 1$$

Verifying correctness of the algorithm is trivial, because

$$\begin{aligned} e(w, g_2^\beta / D) &= e(g_1^{\psi(\beta)}, g_2^\beta / g_2^{\alpha_\delta}) \\ &= e(g_1, g_2)^{\psi(\beta)(\beta - \alpha_\delta)} \\ &= e(g_1, g_2)^{F(\beta)} \\ &= e(g_1^{F(\beta)}, g_2) \\ &= e(C, g_2) \\ &= t \end{aligned}$$

Correctness of membership proof remains intact since GS protocol is complete.

Lemma 2. *If GS proof system is perfectly sound and q-SDH assumption holds in $\mathbb{G}_1$, the set membership proof technique is perfectly sound.*

Proof: Perfect soundness prohibits an adversary from proving a false statement. The technique is perfectly sound if for all adversarial PPT algorithm $\mathcal{A} = \{\mathcal{A}_1, \mathcal{A}_2\}$, we have

$$Pr[mParam \leftarrow \texttt{MemSetup}(1^\kappa, q) \ \wedge \ W' \leftarrow \mathcal{A}_1(mParam, \alpha_\delta, S) \wedge \\ \phi'_{mem} \leftarrow \mathcal{A}_2(mParam, S, W') \ \wedge \ \texttt{MemVerify}((mParam, S, \phi'_{mem}) = 1, \text{if } \alpha_\delta \notin S] < \nu(\kappa)$$

Suppose there exists an adversary $\mathcal{A}_1$ against the soundness of the scheme that produces a witness tuple $\langle \alpha^*_\delta, w^*, D^* \rangle$ accepted by `MemVerify` though $\alpha^* \notin S$. We demonstrate, using $\mathcal{A}_1$ how a challenger $\mathcal{B}$ can be constructed to break q-SDH assumption.

It follows from polynomial remainder theorem (*Little Bezout theorem* [30, 21]) that for any univariate polynomial $P(x) \in \mathbb{Z}_n[x]$, $(x - \alpha)$ is a divisor of $(P(x) - P(\alpha)), \forall \alpha \in \mathbb{Z}_n$. Recall that, $w^* = g_1^{\psi^*(\beta)}$, $D^* = g_2^{\alpha^*}$. Defining $\psi'(\beta) = \frac{F(\beta) - F(\alpha^*)}{\beta - \alpha^*}, w' = g_1^{\psi'(\beta)}, D' = g_2^{\alpha^*}$, it can trivially be checked that $e(w', g_2^\beta / D') e(g_1, g_2)^{F(\alpha^*)} = e(g_1^{F(\beta)}, g_2) = e(C, g_2) = t$ [$\because \alpha^* \notin S \therefore F(\alpha^*) \neq 0$]. Adversary manages to satisfy verification equation, hence $e(w^*, g_2^\beta / D^*) = t$. Equating both the equations above,

$$\begin{aligned} e(w', g_2^\beta / D') e(g_1, g_2)^{F(\alpha^*)} &= e(w^*, g_2^\beta / D^*) \\ e(g_1^{\psi'(\beta)}, g_2^{\beta - \alpha^*}) e(g_1, g_2)^{F(\alpha^*)} &= e(g_1^{\psi^*(\beta)}, g_2^{\beta - \alpha^*}) \\ \psi'(\beta)(\beta - \alpha^*) + F(\alpha^*) &= \psi^*(\beta)(\beta - \alpha^*) \\ (\psi^*(\beta) - \psi'(\beta))(\beta - \alpha^*) &= F(\alpha^*) \\ \frac{1}{\beta - \alpha^*} &= \frac{\psi^*(\beta) - \psi'(\beta)}{F(\alpha^*)} \\ g_1^{\frac{1}{\beta - \alpha^*}} &= g_1^{\frac{\psi^*(\beta) - \psi'(\beta)}{F(\alpha^*)}} \\ g_1^{\frac{1}{\beta - \alpha^*}} &= \left(\frac{w^*}{w'} \right)^{\frac{1}{F(\alpha^*)}} \end{aligned}$$

Therefore the game between challenger $\mathcal{B}$ and adversary $\mathcal{A}_1$ proceeds as follows:

- $\mathcal{B}$ presents a q-SDH instance to $\mathcal{A}_1$
- $\mathcal{A}_1$ returns a witness tuple $\langle \alpha^*_\delta, w^*, D^* \rangle$ accepted by `MemVerify` though $\alpha^* \notin S$
- $\mathcal{B}$ honestly computes the tuple $\langle \alpha^*, w', D' \rangle$
- After a small constant amount of calculation, $\mathcal{B}$ returns $\langle -\alpha^*, g_1^{\frac{1}{\beta - \alpha^*}} \rangle$ as a solution to q-SDH instance

It's apparent that the success probability of $\mathcal{B}$ is same as that of $\mathcal{A}_1$. Also, $\mathcal{A}_2$ has little probability of success since GS protocol is perfectly sound.

Lemma 3. *If GS proofs are zero-knowledge for the satisfiability of quadratic equations, the set membership proof technique is zero-knowledge.*

Proof: Groth-Sahai (GS) proofs are inherently non-interactive witness-indistinguishable (NIWI). Verification equation is in the form of a Pairing Product Equation (PPE) in GS framework. Non-interactive zero-knowledge (NIZK) proofs for such equations are exactly same as NIWI proofs, but there is an additional cost involved in attaining composable ZK (*cf.* 5) for PPE with $e(C, g_2)$ for known C and g_2. In this case, the verification PPE is rewritten and an extra MSME is added to the set for the ZK simulator to be able to simulate zero knowledge. ZK simulator can simply simulate proofs by committing for trivial witnesses by setting all variables to zero as outlined in [17].

10 Appendix D: Security of Generic Ring Signature

Theorem 2. *The generic construction of ring signature scheme outlined in section 4 is a secure one satisfying correctness, anonymity and unforgeability.*

Proof: Correctness, anonymity and unforgeability of the ring signature scheme follow from Lemma 4, 5 and 6 respectively.

Lemma 4. *If GS proof system is perfectly complete and underlying signature scheme* `Sig` *is correct, the ring signature scheme is correct.*

Proof: Obvious.

Lemma 5. *If GS proof system is hiding (i.e. witness-indistinguishable/zero-knowledge), then the ring signature scheme is anonymous under full key exposure.*

Proof: Let $\mathcal{A}$ be any PPT adversary against the anonymity of the ring signature scheme in the full-key-exposure security game described in [2, p. 7]. We will show that the simulator $\mathcal{B}$ can break the hiding property of GS commitment scheme if $\mathcal{A}$ breaks the anonymity of the ring signature scheme under full key exposure.

We instantiate GS system in simulation setting for membership (ϕ_{mem}), signature (ϕ_{sig}) and correlation (ϕ_q, ϕ_{pk}) proofs. All three instantiations of GS proof system, *viz.* Subgroup Decision, SXDH and DLIN, two types of commitment keys, *i.e.* hiding and binding keys are computationally indistinguishable. Hence, adversary $\mathcal{A}$ has negligible advantage in distinguishing between simulation and soundness settings. All the queries to signature oracle $\texttt{OSign}(s, M, \mathcal{R})$ will return ring signatures with hiding commitments.

We are committing signer's public key PK_i and all those components of signature Δ' which depend on signer's secret key SK_i while forming signature proof (ϕ_{sig}). In all other proofs, we are using the same commitment for signer's extended public key PK'_i. Whatever be the public values ring signature scheme exposes, those are independent of signer's secret or public key, therefore not immediately traceable to signer's identity. Also an adversary $\mathcal{A}$ gets little advantage by querying the corrupt oracle $\texttt{Corrupt}(i)$ to obtain random coins $\omega_i,\ \forall i \in [1, |\mathcal{R}|]$ and thereby learning the key-pairs $\langle SK_i, PK_i \rangle,\ \forall i \in [1, |\mathcal{R}|]$, since the signature Σ hides signer's key-pair behind GS commitments. If the adversary can reveal the witnesses $\langle SK_i, PK_i \rangle$ used in the proofs, it reduces to simulator $\mathcal{B}$ breaking the hiding property of GS commitments and therefore, compromising witness-indistinguishability/zero-knowledgeness of GS proof system [17].

Lemma 6. *If GS proof system is perfectly sound, the hash function $\mathcal{H}$ is collision-resistant, and the signature scheme* `Sig` *is existentially unforgeable against adaptive chosen-message attack, the ring signature scheme is unforgeable in the presence of insider corruption.*

Proof: Let $\mathcal{D}$ be a PPT adversary against the unforgeability of the ring signature scheme in the presence of insider corruption (UF-IC) as described in [2, p. 8]. We identify five types of possible forgeries:

- **Type - I**: Forging signatures by producing proofs for false statements.
- **Type - II**: Forging signatures by producing a message m^* and ring $\mathcal{R}^*$ such that, $\mathcal{H}(m^*||\mathcal{R}^*) = \mathcal{H}(m_i||\mathcal{R}_i)$ for some previous query $(m_i, \mathcal{R}_i) \neq (m^*, \mathcal{R}^*),\ \forall i \in [1, q]$, assuming $\mathcal{A}_2$ has made q queries to sign oracle.

- **Type - III**: Forging signatures by producing a message m^* and ring $\mathcal{R}^*$ such that, $\mathcal{H}(m^*||\mathcal{R}^*) \neq \mathcal{H}(m_i||\mathcal{R}_i)$ for some previous query $(m_i, \mathcal{R}_i) \neq (m^*, \mathcal{R}^*)$, $\forall i \in [1, q]$, assuming $\mathcal{A}_3$ has made q queries to sign oracle.
- **Type - IV**: Forging signature by recovering secret key $SK_i = \{sk_{ij}\}$ of i-th member of ring having k members from $\{q_{ij}\}, i \in [1, k], j \in [1, M], M = |SK_i|$
- **Type - V**: Forging signature by recovering secret key $SK_i = \{sk_{ij}\}$ of i-th member of ring having k members from $\{pk_{ij}\}, i \in [1, k], j \in [1, M], M = |SK_i|$

Let $\mathcal{A}_1$ be an adversary against the perfect soundness of GS commitment scheme. We instantiate GS proof system in soundness setting. Due to inherent binding property, adversary $\mathcal{A}_1$ has zero advantage in faking proofs for false statements. Therefore, $Adv^{Sound}_{GS,\mathcal{A}_1}(\delta) = 0$

Let $\mathcal{A}_2$ be an adversary against the collision resistance property of hash function $\mathcal{H}$. We can reduce Type - II forgeries to breaking collision-resistance property of hash function. A collision-resistant hash functions does not have any known technique more efficient than brute-force to find collisions computationally. Therefore, $Adv^{Coll}_{\mathcal{H},\mathcal{A}_2}(\delta) \leq \nu(\delta)$

Let $\mathcal{A}_3$ be an adversary against the existential unforgeability of underlying signature scheme $\texttt{Sig}$ under chosen-message attack (EUF-CMA). Consider a simulator $\mathcal{B}$ playing the game EUF-CMA with $\mathcal{A}_3$. We will finally present a reduction of Type - III forgeries to EUF-CMA [15]. Our reduction proves non-negligible advantage of $\mathcal{A}_3$ in EUF-CMA security game, provided there exists an adversary $\mathcal{D}$ that successfully breaks the unforgeability of the ring signature [5, p. 3] in the presence of insider corruption.

The interaction between $\mathcal{A}_3$ and $\mathcal{D}$ is modeled as follows:

- $\mathcal{A}_3$ generates public parameter of the ring signature scheme $rParam$ by instantiating GS proof system in soundness setting. Also, $\mathcal{A}_3$ arbitrarily chooses $i^* \in [1, k]$ and generates $(k-1)$ secret-public key-pairs for $i \in [1, k]\backslash\{i^*\}$, where $k = n(t)$, $n(x)$ being a polynomial and $t \in \mathbb{N}$. $\mathcal{A}_3$ obtains the public key PK_{i^*} which it wants to be challenged on from simulator $\mathcal{B}$. $\mathcal{B}$ doesn't reveal corresponding secret key SK_{i^*} and keeps it with itself. $\mathcal{A}_3$ hands over the tuple $(rParam, \{PK_1, PK_2, ..., PK_{i^*}, ..., PK_k\})$ to ring adversary $\mathcal{D}$.
- When $\mathcal{D}$ queries for $\texttt{Corrupt}(i)$ on any $i \neq i^*$, $\mathcal{A}_3$ reveals corresponding secret key SK_i. If the same query is on $i = i^*$, $\mathcal{A}_3$ aborts with probability $1/n$.
- When $\mathcal{D}$ queries for $\texttt{Sign}(s, M, \mathcal{R})$ on any $i \neq i^*$, $\mathcal{A}_3$ plays the ring signature protocol itself and hands the output over to $\mathcal{D}$. If the same query is on $i = i^*$, $\mathcal{A}_3$ queries the sign oracle of its own unforgeability game on $\mathcal{H}(M||\mathcal{R})$, generates the remaining part of the ring signature itself and hands the output over to $\mathcal{D}$.
- After a series of $\texttt{Corrupt}(i)$ and $\texttt{Sign}(s, M, \mathcal{R})$ queries, $\mathcal{D}$ eventually terminates by outputting a message-ring signature-ring tuple $(M^*, \Sigma^*, \mathcal{R}^*)$. Let T be the set of corrupted users. $\mathcal{A}_3$ verifies that all members in $\mathcal{R}^*$ are indeed honest, in other words, $\mathcal{A}_3$ has never answered a query $\texttt{Sign}(*, M^*, \mathcal{R}^*)$, $\mathcal{R}^* \subseteq S - T$. If all conditions are met, $\mathcal{A}_3$ extracts public key PK^* and signature Δ^* by invoking $\texttt{GSExtract}(.)$ as it has access to secret trapdoor xk. In an extreme situation, $T = \{\}$ and $|\mathcal{R}^*| = n$. Note that, during corruption phase, $\mathcal{A}_3$ was unable to supply SK_{i^*}. Therefore, $PK_{i^*} \in \mathcal{R}^*$ with probability $1/n$ at most. If $PK^* \neq PK_{i^*}$, $\mathcal{A}_3$ aborts. Otherwise, $\mathcal{A}_3$ returns $(\mathcal{H}(M^*||\mathcal{R}^*), \Delta^*)$ as successful forgery to the challenger $\mathcal{B}$ in its own unforgeability game. Therefore, $Adv^{Unforg}_{\texttt{Sig},\mathcal{A}_3}(\delta) \geq (1/n) Adv^{Unforg}_{\texttt{RSig},\mathcal{D}}(\delta)$, or $Adv^{Unforg}_{\texttt{RSig},\mathcal{D}}(\delta) \leq n(\delta) \cdot Adv^{Unforg}_{\texttt{Sig},\mathcal{A}_3}(\delta)$

Let $\mathcal{A}_4$ be an adversary solving an **SQROOT** instance which we can reduce Type - IV forgeries to. Therefore, $Adv^{SQRT}_{\mathcal{A}_4}(\delta) \leq \nu(\delta)$

Let $\mathcal{A}_5$ be an adversary attempting to recover secret keys from the knowledge of public keys. We remark that such an adversary are no stronger than $\mathcal{A}_3$ type of adversary and advantage due to it is encompassed by $Adv^{Unforg}_{\texttt{Sig},\mathcal{A}_3}(\delta)$

Combining the advantages of all three types of adversaries,

$$Adv^{Unforg}_{\texttt{RSig},\mathcal{D}}(\delta) \leq Adv^{Sound}_{GS,\mathcal{A}_1}(\delta) + Adv^{Coll}_{\mathcal{H},\mathcal{A}_2}(\delta) + n(\delta) \cdot Adv^{Unforg}_{\texttt{Sig},\mathcal{A}_3}(\delta) + Adv^{SQRT}_{\mathcal{A}_4}(\delta)$$