Formalizing Group Blind Signatures and Practical Constructions without Random Oracles

E. Ghadafi

Dept. Computer Science,
University of Bristol,
United Kingdom.
ghadafi@cs.bris.ac.uk

Abstract. Group blind signatures combine anonymity properties of both group signatures and blind signatures and offer privacy for both the message to be signed and the signer. Their applications include multi-authority e-voting and distributed e-cash systems.
The primitive has been introduced with only informal definitions for its required security properties. We offer two main contributions: first, we provide foundations for the primitive where we present formal security definitions offering various flavors of anonymity relevant to this setting. In the process, we identify and address some subtle issues which were not considered by previous constructions and (informal) security definitions.
Our second main contribution is a generic construction that yields practical schemes with round-optimal signing and constant-size signatures. Our constructions permit dynamic and concurrent enrollment of new members, satisfy strong security requirements, and do not rely on random oracles.
In addition, we introduce some new building blocks which may be of independent interest.

Keywords: Group Signatures, Blind Signatures, Group Blind Signatures, Standard Model.

1 Introduction

Background. Traditional group signatures, introduced by Chaum and van Heyst [19], allow a member of a group to sign a message anonymously on behalf of the group while ensuring that a recipient of the signature cannot identify the group member who signed the message. However, there exists an entity called the Opener who possesses a special key that allows him to revoke anonymity and reveal the identity of the signer in the case of a dispute. A Blind Signature (BS), introduced by Chaum [18], allows a user to obtain a signature on a hidden message without the signer learning the message in question. The security of a blind signature scheme [34, 44] requires that the user is unable to fake new signatures for new messages (unforgeability) and that the signer does not learn the message he is signing nor be able to link a signature to the protocol run where it was obtained (blindness). A Group Blind Signature (GBS), introduced by Lysyanskaya and Zulfikar [37], combines the properties of both a group signature scheme and a blind signature scheme and therefore it maintains the anonymity of the signer as well as the message to be signed.

Group blind signatures are useful for many applications such as distributed e-cash systems where, for instance, it is required that a digital coin neither reveals the identity of its holder nor that of the issuing bank/branch. Another example is in a multi-authority e-voting system where it is required that the vote is certified by a governing authority without revealing either the actual value of the vote or the identity of the certifying authority/branch.

Motivation. Currently there is no formally defined security model for the primitive and hence it is important to provide formal security definitions that include the details of all the security experiments needed to capture the required security properties. Providing such a model would allow for proving the security of constructions formally and more rigorously. Some of the previous constructions assumed that the exact security definitions of group signatures directly translate into this setting. However, such an assumption is erroneous due to the signing protocol in this case being blind which is different from the group signature case where the message is public. For instance, one needs to ensure the existence of some mechanism to prevent the adversary from breaking anonymity by outputting a message-signature pair different from the one it got from interacting with the challenge oracle as we discuss later in the definition of the anonymity property. Also, the blindness definition should capture the case that different group members might collude to break blindness which is again different from the blind signature case where there is only one signer and signatures cannot be traced.

In addition, existing constructions [37, 41] require random oracles [8] and involve many rounds of interaction in the signing phase (some relying on interactive divertible proofs of knowledge [10]) and hence requiring that the signing authority remains online throughout the signing process. The schemes we propose have a round-optimal signing phase and their security does not require any idealized assumptions. Therefore, they are more suitable for practical applications.

Related Work. The concept of group blind signatures was first introduced by Lysyanskaya and Zulfikar [37], where it was mainly used to design a distributed e-cash system in which digital coins could be issued by different banks. While there exist a number of group signature schemes, e.g. [16, 7, 13, 4, 14, 31], only a few group blind signature schemes exist in the literature [37, 41].

The subtlety one faces when designing group blind signatures lies in the dual privacy requirement. On the one hand, the signer needs to hide his identity and parts of the signature that could identify him (i.e. anonymity of the signer requirement). On the other hand, the user wants to hide the message and parts of the signature which could lead to a linkage between a signature and its sign request (i.e. the blindness requirement). The schemes in [37] are based on variants of the Camenisch-Stadler group signature [16] and their security requires random oracles. Other schemes, e.g. [41], use divertible zero-knowledge proofs [10, 42] to realize those conflicting anonymity requirements. A divertible proof allows a mediator to use a proof it got from a party to prove a statement to a third party. Constructions that rely on such proofs require many rounds of interaction in the signing protocol and/or the Fiat-Shamir transformation [21] to eliminate the interaction and hence lying in the random oracle model.

Our Contribution. Our first contribution is the formalization of the security requirements for the primitive. In the process, we identify and address some subtle issues which arise when defining the different security notions.

Our second contribution is a generic construction which yields practical schemes in the standard model. We provide two example instantiations of the construction, the first of which is solely based on non-interactive complexity assumptions that are falsifiable [39]. The second instantiation is more efficient but makes use of an interactive assumption. All our constructions are round-optimal, yield constant-size signatures, and allow for members of the group to join dynamically and concurrently. Moreover, their security is proven in the standard model. We start by showing how to construct CPA-anonymous schemes and then outline how they can be extended to provide full anonymity. We also provide a proof of security for our constructions.

Paper Organization. The rest of the paper is organized as follows: In Section 2, we give some preliminary definitions. In Section 3, we define dynamic group blind signatures. We present the security model for dynamic group blind signatures in Section 4. We present the building blocks we use in Section 5. In Section 6, we present our constructions and provide a proof of their security. In Section 7, we outline how we can achieve full anonymity as well as other instantiations. Finally, we conclude the paper in Section 8.

2 Preliminary Definitions

Notation. We say that a function $\nu(.) : \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible in c if for every polynomial $p(.)$ and all sufficiently large values of c, it holds that $\nu(c) < \frac{1}{p(c)}$. Given a probability distribution S, we denote by $y \leftarrow S$ the operation of selecting an element according to S. If A is a probabilistic machine, we denote by $A(x_1, \ldots, x_n)$ the output distribution of A on inputs $(x_1, \ldots, x_n)$. By PPT we mean running in probabilistic polynomial time in the relevant security parameter. By $[1, n]$, we mean the set $\{1, 2, \ldots, n\}$. We denote by $\langle A, B \rangle$ an interactive protocol involving algorithms A and B. Occasionally we will use the notation $\langle A, B \rangle^i$ for $i \in \mathbb{N}$ denoting the number of times such an interactive protocol is allowed to take place. If $i = *$, such an interaction can be invoked unlimited number of times. For a matrix M, M^T denotes its transpose.

2.1 Bilinear Groups

Bilinear groups are a set of three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, of prime order p, along with a bilinear map (a deterministic function) $\hat{e}$. We will use multiplicative notation for all the groups although usually $\mathbb{G}_1$ and $\mathbb{G}_2$ are chosen to be additive groups. We let $\mathbb{G}^\times := \mathbb{G} \setminus \{1_\mathbb{G}\}$ and write $\mathbb{G}_1 := \langle G_1 \rangle, \mathbb{G}_2 := \langle G_2 \rangle$, for two explicitly given generators G_1 and G_2.

We define $\mathcal{P} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, G_1, G_2)$ to be the set of pairing group parameters. The function $\hat{e}$ must be bilinear where $\forall Q_1 \in \mathbb{G}_1$, $\forall Q_2 \in \mathbb{G}_2$, $\forall x, y \in \mathbb{Z}_p$, we have $\hat{e}(Q_1^x, Q_2^y) = \hat{e}(Q_1, Q_2)^{xy}$. We also require that the value $\hat{e}(G_1, G_2)$ generates $\mathbb{G}_T$ and that the function $\hat{e}$ is efficiently computable. Following [25], we categorize pairings into three distinct types (other types are possible, but the following three are the main ones utilized in practical protocols).

- **Type-1**: This is the symmetric pairing setting in which $\mathbb{G}_1 = \mathbb{G}_2$.
- **Type-2**: Here $\mathbb{G}_1 \neq \mathbb{G}_2$, but there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \longrightarrow \mathbb{G}_1$.
- **Type-3**: Again $\mathbb{G}_1 \neq \mathbb{G}_2$, but now there is no known efficiently computable isomorphism.

In the remainder of the paper, we will assume that all three groups are cyclic and that there is an algorithm $\mathsf{BGrpSetup}$ which takes as input a security parameter λ and produces a description of bilinear groups $\mathcal{P}$.

2.2 Complexity Assumptions

Definition 1. Symmetric External Diffie-Hellman (SXDH) Assumption: *In Type-3 pairings, the Decisional Diffie-Hellman (*DDH*) problem is believed to be hard in both groups* $\mathbb{G}_1$ *and* $\mathbb{G}_2$*.*

To present the next assumption, we define the subgroup $\hat{G}$ of $\mathbb{G}_1 \times \mathbb{G}_2$ as the image of the map

$$\gamma : \begin{cases} \mathbb{Z}_p \longrightarrow \mathbb{G}_1 \times \mathbb{G}_2 \\ \ x \ \longmapsto (G_1^x, G_2^x) \end{cases}$$

Note that if we are given an element $(A, B) \in \mathbb{G}_1 \times \mathbb{G}_2$ we can efficiently test whether $(A, B) \in \hat{G}$ by testing whether $\hat{e}(A, G_2) = \hat{e}(G_1, B)$. The following new assumption is a variant of the standard LRSW assumption [36]. It is similar to a number of blind-variants of the LRSW assumption, in that the adversary has an oracle for group elements and outputs "signatures" on group elements as opposed to elements in $\mathbb{Z}_p$ [20]. In addition, we assume that the elements being "signed" are in $\hat{G}$, as opposed to either $\mathbb{G}_1$ or $\mathbb{G}_2$. Apart from this latter change, the assumption is similar to prior blind-LRSW assumptions [29, 20].

Definition 2 (Dual Hidden-LRSW (DH-LRSW) Assumption). *Consider the following experiment. A challenger picks* $x, y \in \mathbb{Z}_p$ *and then computes* $X := G_2^x$, $Y := G_2^y$*. The tuple* (X, Y) *is given to an adversary* $\mathcal{A}$ *which has access to an oracle* $\mathsf{O}_{X,Y}(\cdot)$*. The oracle on input* $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$*, outputs*

- *The symbol* $\perp$ *if* $(M_1, M_2) \notin \hat{G}$*.*
- *Otherwise it outputs a tuple* $(A, B, C, D) := (G_1^a, G_1^{ay}, M_1^{ay}, G_1^{ax} \cdot M_1^{axy})$ *where* $a \leftarrow \mathbb{Z}_p^*$*.*

We let $\mathcal{Q}$ *denote the set of queries* (M_1, M_2) *passed to the oracle* $\mathsf{O}_{X,Y}(\cdot)$*. The adversary eventually terminates by outputing a tuple* $(M_1^*, M_2^*, A^*, B^*, C^*, D^*)$*. We say the output of* $\mathsf{BGrpSetup}$ *satisfies the* DH-LRSW assumption *if all PPT adversaries* $\mathcal{A}$ *have a negligible advantage* $\mathsf{Adv}_\mathcal{A}^{DH\text{-}LRSW}(\lambda)$ *in the above game, where the advantage is defined as follows:*

$$\mathsf{Adv}_\mathcal{A}^{DH\text{-}LRSW}(\lambda) := \Pr\left[\begin{array}{l} (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, G_1, G_2) \leftarrow \mathsf{BGrpSetup}(1^\lambda); x, y \leftarrow \mathbb{Z}_p; X := G_2^x; Y := G_2^y; \\ (M_1^*, M_2^*, A^*, B^*, C^*, D^*) \leftarrow \mathcal{A}^{\mathsf{O}_{X,Y}(\cdot)}(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, G_1, G_2, X, Y) : \\ \exists a \in \mathbb{Z}_p^* \ \ s.t. \ \ (M_1^*, M_2^*) \in \hat{G} \setminus \mathcal{Q} \ \wedge \ A^* = G_1^a \ \wedge \ B^* = G_1^{ay} \ \wedge \ C^* = M_1^{*ay} \\ \quad \wedge \ D^* = G_1^{ax} \cdot M_1^{*axy} \end{array}\right].$$

Note that the output from the oracle can be checked to be valid by checking whether

$$\begin{aligned}\hat{e}(A, Y) &= \hat{e}(B, G_2)\\ \hat{e}(B, M_2) &= \hat{e}(C, G_2)\\ \hat{e}(D, G_2) &= \hat{e}(A \cdot C, X).\end{aligned}$$

Also, note that only the M_1 component of the pair of values (M_1, M_2) used as input to the oracle are actually used. However, the fact that the oracle returns $\perp$ if $(M_1, M_2) \notin \hat{G}$ is crucial to our security proof in the generic group model, since it implies that the adversary learns nothing if he cannot produce the pair in $\hat{G}$; which intuitively implies that he "knows" the preimage of the pair under the map γ. Likewise the output from the adversary must be a pair $(M_1^*, M_2^*) \in \hat{G}$ for a similar reason.

To justify the hardness of this assumption, we prove it holds in the generic group model [50, 38] in Appendix C.

Definition 3 (AWFCDH Assumption). *This assumption was introduced in [23]. We describe this assumption in asymmetric bilinear groups. Given* $(G_1, G_1^a, G_2) \in \mathbb{G}_1^{\times 2} \times \mathbb{G}_2^{\times}$ *for* $a \leftarrow \mathbb{Z}_p$, *it is computationally infeasible to output a tuple* $(G_1^b, G_1^{ab}, G_2^b, G_2^{ab}) \in \mathbb{G}_1^{\times 2} \times \mathbb{G}_2^{\times 2}$ *for an arbitrary* $b \in \mathbb{Z}_p$.

The next assumption was introduced in [23] and is a variant of the q-SDH assumption [12].

Definition 4 (q-ADHSDH Assumption). *We describe this assumption in asymmetric bilinear groups. Given* $(G_1, F, K, G_1^x, G_2, G_2^x) \in \mathbb{G}_1^{\times 4} \times \mathbb{G}_2^{\times 2}$ *for* $x \leftarrow \mathbb{Z}_p$, *and* $q-1$ *tuples* $(A_i := (K \cdot G_1^{r_i})^{\frac{1}{x+c_i}}, C_{1,i} := F^{c_i}, C_{2,i} := G_2^{c_i}, R_{1,i} := G_1^{r_i}, R_{2,i} := G_2^{r_i})_{i=1}^{q-1}$, *where* $c_i, r_i \leftarrow \mathbb{Z}_p$, *it is computationally infeasible to output a new tuple* $(A^*, C_1^*, C_2^*, R_1^*, R_2^*)$ *of this form.*

3 Defining Dynamic Group Blind Signatures

The parties involved in a group blind signature are: an authority called the Issuer who controls who can join the group, an authority called the Opener who can revoke the anonymity of the signer by opening signatures and revealing who signed them. A number of signers Signer_i each of which has a unique identity and can sign on behalf of the group once they have joined the group. External users User_i which can ask for messages to be blindly signed by members of the group. In our definition, we do not require that users (i.e. entities who request signatures) are traceable (unless the identifying information is embedded in the messages themselves) and thus we do not assign keys to them. However, the model can be extended to provide this functionality. A group blind signature scheme GBS is a tuple of polynomial-time algorithms

$$\mathsf{GBS} := (\mathsf{GKg}, \mathsf{SKg}, \langle \mathsf{Join}, \mathsf{Issue} \rangle, \langle \mathsf{Obtain}, \mathsf{Sign} \rangle, \mathsf{GVf}, \mathsf{Open}, \mathsf{Judge}),$$

which are defined as follows:

- **GKg:** Is run by some trusted third party (TTP) which takes as input a security parameter $\lambda \in \mathbb{N}$ and generates the group public key gpk, the Issuer's key $\mathfrak{ik}$ and the Opener's key $\mathfrak{ok}$. The secret keys $\mathfrak{ik}$ and $\mathfrak{ok}$ are then securely transmitted to the respective authorities.
- **SKg:** Is run by a potential group member Signer_i, to generate his pair of personal secret/public keys $(\mathbf{ssk}[i], \mathbf{spk}[i])$ prior to requesting to join the group. We assume that the public key table $\mathbf{spk}$ is publicly accessible (possibly via some PKI).
- $\langle \mathsf{Join}(\mathsf{gpk}, i, \mathbf{ssk}[i]), \mathsf{Issue}(\mathfrak{ik}, i, \mathbf{spk}[i]) \rangle$**:** Is an interactive protocol between a signer Signer_i and the Issuer. After a successful completion of this protocol, Signer_i becomes a member of the group. If successful, the final state of the Issue algorithm is stored in the registration table at index i (i.e. $\mathbf{reg}[i]$) while that of the Join algorithm is stored in $\mathbf{gsk}[i]$. We assume that the communication in this interactive protocol takes place over a secure (i.e. private and authentic) channel. We also assume that the protocol is initiated by a call to Join.

- $\langle \mathsf{Obtain}(\mathsf{gpk}, m), \mathsf{Sign}(\mathbf{gsk}[i])\rangle$**:** Is an interactive protocol between a user User and an anonymous member of the group. If the protocol completes successfully, User obtains a blind signature Σ on the message m without members of the group learning what the message was or the user learning which group member produced the signature. If any of the parties abort, User outputs $\perp$. This protocol is initiated by a call to Obtain.
- $\mathsf{GVf}(\mathsf{gpk}, m, \Sigma)$**:** Is a deterministic algorithm which takes as input the group public key gpk, a message m and a group blind signature Σ and outputs 1 if Σ is a valid signature on the message m, or 0 otherwise.
- $\mathsf{Open}(\mathsf{gpk}, \mathfrak{ok}, \mathbf{reg}, m, \Sigma)$**:** Is a deterministic algorithm in which the Opener uses his key $\mathfrak{ok}$ to identify the identity i of the signer form the group blind signature Σ and produces a proof τ attesting to this claim.
- $\mathsf{Judge}(\mathsf{gpk}, i, \mathbf{spk}[i], m, \Sigma, \tau)$**:** This is a deterministic algorithm which takes as input an index i and returns 1 if the group blind signature Σ was produced by the group member Signer_i or 0 otherwise.

4 Security of Dynamic Group Blind Signatures

In defining the security model, we build on the security models used for group signatures [7, 9]. We assume that the group manager is divided into two separate entities: the Issuer and the Opener. The security properties we require from a dynamic group blind signature GBS are: correctness, anonymity, traceability, non-frameability, and blindness.

The security of group blind signatures is formulated via a set of experiments in which the adversary has access to a set of oracles. In those experiments, a set of global lists are maintained. The lists are: HSL is a set of honest signers; CSL is a set of corrupt signers whose keys have been chosen by the adversary and whose states have been learned by the adversary; BSL is a set of bad signers whose secret keys have been revealed to the adversary; CLA is a set containing the identities of the challenge signers used when calling the challenge oracle in the anonymity game; CLB is a list containing pairs of challenge message-signature in the blindness game; a table $\mathbf{reg}$ where the element i in this table contains the registration information of the group member Signer_i; a table $\mathbf{spk}$ where $\mathbf{spk}[i]$ contains the personal public key of the group member Signer_i. The lists HSL, CSL, BSL, CLA and CLB are empty at initialization, whereas the entries of the tables $\mathbf{reg}$ and $\mathbf{spk}$ are initialized to ϵ. The set of oracles the adversary has access to in the experiments are defined as follows, the details are shown in Figure 1.

- $\mathsf{AddS}(i)$: The adversary can use this oracle to add an honest signer Signer_i to the group.
- $\mathsf{CrptS}(i, \mathsf{pk})$: The adversary can use this oracle to create a new corrupt signer Signer_i, where Signer_i's public key $\mathbf{spk}[i]$ is chosen by the adversary. This oracle is usually called in preparation for calling the SndToI oracle.
- $\mathsf{SndToI}(i, M_{in})$: The adversary can use this oracle to engage in the $\mathsf{Join} - \mathsf{Issue}$ protocol with the honest, Issue-executing Issuer.
- $\mathsf{SndToS}(i, M_{in})$: This oracle models the scenario that the adversary has corrupted the Issuer. The adversary uses this oracle to engage in the $\mathsf{Join} - \mathsf{Issue}$ protocol with an honest, Join-executing Signer.
- $\mathsf{ReadReg}(i)$: The adversary can call this oracle to obtain the content of entry $\mathbf{reg}[i]$.
- $\mathsf{ModifyReg}(i, val)$: The adversary can call this oracle to modify the content of entry $\mathbf{reg}[i]$ by setting $\mathbf{reg}[i] = val$.
- $\mathsf{SSK}(i)$: The adversary can call this oracle to obtain both the personal secret key $\mathbf{ssk}[i]$ and the group signing key $\mathbf{gsk}[i]$ of group member Signer_i.
- $\mathsf{OSign}(i)$: This is an interactive oracle (i.e. the adversary must engage in an interaction with this oracle). After a successful interaction with this oracle, the adversary obtains a blind signature by group member Signer_i on a message of its choice.
- $\mathsf{CH}_b(i_0, i_1)$: This oracle is a left-right oracle for defining anonymity and is only called once. The adversary sends a couple of identities (i_0, i_1) and interacts with this oracle in order to produce a blind signature on a message m by the group member Signer_{i_b} for $b \leftarrow \{0, 1\}$.
- $\mathsf{Open}(m, \Sigma)$: This oracle allows the adversary to ask for signatures to be opened by revealing the identity of the group member who signed them.

We define the security properties as follows, where we use a set of experiments as shown in Figure 2 in which the adversary has access to the previously defined oracles.

```
$\underline{\mathsf{AddS}(i)}$:
  – If $i \in \mathsf{HSL} \cup \mathsf{CSL} \cup \mathsf{BSL}$ Then Return $\perp$.
  – $\mathsf{HSL} := \mathsf{HSL} \cup \{i\}$.
  – $(\mathbf{ssk}[i], \mathbf{spk}[i]) \leftarrow \mathsf{SKg}(1^\lambda)$.
  – $\mathsf{cert}_i := \perp$, $\mathsf{dec}^i_{\mathsf{Issue}} := \mathsf{cont}$.
  – $\mathsf{St}^i_{\mathsf{Join}} := (\mathsf{gpk}, i, \mathbf{ssk}[i])$.
  – $\mathsf{St}^i_{\mathsf{Issue}} := (\mathfrak{ik}, i, \mathbf{spk}[i])$.
  – $(\mathsf{St}^i_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathsf{dec}^i_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{St}^i_{\mathsf{Join}}, \perp)$.
  – While ($\mathsf{dec}^i_{\mathsf{Issue}} = \mathsf{cont}$ and $\mathsf{dec}^i_{\mathsf{Join}} = \mathsf{cont}$) Do
      • $(\mathsf{St}^i_{\mathsf{Issue}}, M_{\mathsf{Join}}, \mathsf{dec}^i_{\mathsf{Issue}}) \leftarrow \mathsf{Issue}(\mathsf{St}^i_{\mathsf{Issue}}, M_{\mathsf{Issue}})$.
      • $(\mathsf{St}^i_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathsf{dec}^i_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{St}^i_{\mathsf{Join}}, M_{\mathsf{Join}})$.
  – If $\mathsf{dec}^i_{\mathsf{Issue}} = \mathsf{accept}$ Then $\mathbf{reg}[i] := \mathsf{St}^i_{\mathsf{Issue}}$.
  – If $\mathsf{dec}^i_{\mathsf{Join}} = \mathsf{accept}$ Then $\mathbf{gsk}[i] := \mathsf{St}^i_{\mathsf{Join}}$.
  – Return $\mathbf{spk}[i]$.

$\underline{\mathsf{ModifyReg}(i, val)}$:
  – $\mathbf{reg}[i] := val$.

$\underline{\mathsf{SndToI}(i, M_{\mathsf{in}})}$:
  – If $i \notin \mathsf{CSL} \cup \mathsf{BSL}$ Then Return $\perp$.
  – If $\mathsf{dec}^i_{\mathsf{Issue}} \neq \mathsf{cont}$ Then Return $\perp$.
  – If $\mathsf{St}^i_{\mathsf{Issue}}$ is undefined Then
      • $\mathsf{St}^i_{\mathsf{Issue}} := (\mathfrak{ik}, i, \mathbf{spk}[i])$.
  – $(\mathsf{St}^i_{\mathsf{Issue}}, M_{\mathsf{out}}, \mathsf{dec}^i_{\mathsf{Issue}}) \leftarrow \mathsf{Issue}(\mathsf{St}^i_{\mathsf{Issue}}, M_{\mathsf{in}})$.
  – If $\mathsf{dec}^i_{\mathsf{Issue}} = \mathsf{accept}$ Then $\mathbf{reg}[i] := \mathsf{St}^i_{\mathsf{Issue}}$.
  – Return $(M_{\mathsf{out}}, \mathsf{dec}^i_{\mathsf{Issue}})$.

$\underline{\mathsf{SndToS}(i, M_{\mathsf{in}})}$:
  – If $i \in \mathsf{CSL} \cup \mathsf{BSL}$ Then Return $\perp$.
  – If $i \notin \mathsf{HSL}$ Then
      • $\mathsf{HSL} := \mathsf{HSL} \cup \{i\}$.
      • $(\mathbf{ssk}[i], \mathbf{spk}[i]) \leftarrow \mathsf{SKg}(1^\lambda)$.
      • $\mathbf{gsk}[i] := \epsilon$, $M_{\mathsf{in}} := \epsilon$.
  – If $\mathsf{dec}^i_{\mathsf{Join}} \neq \mathsf{cont}$ Then Return $\perp$.
  – If $\mathsf{St}^i_{\mathsf{Join}}$ is undefined Then
      • $\mathsf{St}^i_{\mathsf{Join}} := (\mathsf{gpk}, i, \mathbf{ssk}[i])$.
  – $(\mathsf{St}^i_{\mathsf{Join}}, M_{\mathsf{out}}, \mathsf{dec}^i_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{St}^i_{\mathsf{Join}}, M_{\mathsf{in}})$
  – If $\mathsf{dec}^i_{\mathsf{Join}} = \mathsf{accept}$ Then $\mathbf{gsk}[i] := \mathsf{St}^i_{\mathsf{Join}}$.
  – Return $(M_{\mathsf{out}}, \mathsf{dec}^i_{\mathsf{Join}})$.

$\underline{\mathsf{CrptS}(i, \mathsf{pk})}$:
  – If $i \in \mathsf{HSL} \cup \mathsf{CSL} \cup \mathsf{BSL}$ Then Return $\perp$.
  – $\mathsf{CSL} := \mathsf{CSL} \cup \{i\}$.
  – $\mathbf{spk}[i] := \mathsf{pk}$.
  – $\mathsf{St}^i_{\mathsf{Issue}} := (\mathfrak{ik}, i, \mathbf{spk}[i])$.
  – $\mathsf{dec}^i_{\mathsf{Issue}} := \mathsf{cont}$.
  – Return 1.

$\underline{\mathsf{SSK}(i)}$:
  – If $i \notin \mathsf{HSL} \setminus \{\mathsf{CSL} \cup \mathsf{BSL}\}$ Then Return $\perp$.
  – $\mathsf{BSL} := \mathsf{BSL} \cup \{i\}$.
  – Return $(\mathbf{gsk}[i], \mathbf{ssk}[i])$.

$\underline{\mathsf{ReadReg}(i)}$:
  – Return $\mathbf{reg}[i]$.

$\underline{\mathsf{OSign}(i)}$:
  – If $i \in \mathsf{CSL} \cup \mathsf{BSL}$ or $\mathbf{gsk}[i] = \epsilon$ Then Return $\perp$.
  – Call $\mathsf{Sign}(\mathbf{gsk}[i])$.

$\underline{\mathsf{CH}_b(i_0, i_1)}$:
  – If $i_0 \notin \mathsf{HSL} \cup \mathsf{BSL}$ or $i_1 \notin \mathsf{HSL} \cup \mathsf{BSL}$ Then Return
    $\perp$.
  – If $\mathbf{gsk}[i_0] = \epsilon$ or $\mathbf{gsk}[i_1] = \epsilon$ Then Return $\perp$.
  – $\mathsf{CLA} := \mathsf{CLA} \cup \{i_0, i_1\}$.
  – Call $\mathsf{Sign}(\mathbf{gsk}[i_b])$.

$\underline{\mathsf{Open}(m, \Sigma)}$:
  – If $\mathsf{GVf}(\mathsf{gpk}, m, \Sigma) = 0$ Then Return $(\perp, \perp)$.
  – $(id, \tau) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathfrak{ok}, \mathbf{reg}, m, \Sigma)$.
  – If game is Anonymity
      • If $id \in \mathsf{CLA}$ Then Return $(\perp, \perp)$.
  – ElseIf game is Blindness
      • If $(m, \Sigma) \in \mathsf{CLB}$ Then Return $(\perp, \perp)$. [1]
  – Return $(id, \tau)$.
```

Fig. 1. Oracles used in the security experiments for dynamic group blind signatures

4.1 Correctness

A dynamic group blind signature scheme GBS is correct if: all correctly produced signatures are accepted by the GVf algorithm, the Opener is always able to identify the honest group member who produced a signature and the Judge algorithm always accepts the decision made by the Opener.

[1] This condition assuming that the signing protocol is strongly unforgeable. If weak unforgeability is required then this is replaced by the following check: If $(m, \cdot) \in \mathsf{CLB}$ Then Return $(\perp, \perp)$.

More formally, a group blind signature scheme GBS is correct if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$ have a negligible advantage $\mathsf{Adv}^{Corr}_{\mathsf{GBS},\mathcal{A}}(\lambda)$ where the advantage is defined as follows:

$$\mathsf{Adv}^{Corr}_{\mathsf{GBS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{Corr}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1].$$

4.2 Anonymity

A dynamic group blind signature scheme GBS is anonymous if the adversary is unable to tell which group member produced a signature. We require that given two signers of its choice, the adversary cannot tell which of the two signers produced a signature. We provide the adversary with strong capabilities, for instance, it can fully corrupt the Issuer and can ask for signers' personal secret keys/group signing keys to be revealed including the two signers it chooses for the challenge which ensures security against full key exposure. The only restriction we impose on the adversary is that it may not query the Open oracle on the signature it is being challenged on.

As is the case with the definition of anonymity for group signatures, we distinguish between CPA-anonymity and Full Anonymity (FA). In CPA-anonymity [13] (which is analogous to IND-CPA security for public-key encryption schemes [28]), the adversary is not given access to an Open oracle that revokes the anonymity of signatures by returning the identity of the group member who produced them.

On the contrary, in full anonymity [9] (which is analogous to IND-CCA2 security for public-key encryption schemes [45]), the adversary can ask Open queries on any signature except the one it is being challenged on at any stage of the game. One can also consider a weaker non-adaptive variant of full anonymity which we refer to as Weaker Full Anonymity (WFA) (which is analogous to IND-CCA1 security for public-key encryption schemes [40]) in which the adversary can only ask Open queries before it calls the challenge oracle.

The issue with defining full anonymity for group blind signatures is that the signing protocol is a blind one and unlike in group signatures, the challenge message-signature pair is only revealed by the adversary (playing the role of the user) at the end of the signing interaction with the CH_b oracle. Therefore, it is problematic to identify the signature the adversary obtained from interacting with the CH_b oracle. An adversary can trivially break anonymity by revealing a bogus message-signature pair different from the one it got from interacting with the CH_b oracle and then at a later stage asks the Open oracle to open the original challenge signature. Note that the adversary's knowledge of the secret keys of the challenge group members used in calling the challenge oracle makes the adversary capable of signing on behalf of those members.

The First Attempt. To catch the adversary if it returns a bogus challenge signature, one could use the Opener's key known the challenger running the game. Assume that the adversary runs in 3 respective modes (find, obtain, and guess). In the obtain phase, the adversary interacts with the challenge oracle but access to the the Open oracle is disabled. At the end of the obtain phase, the adversary must either output the challenge signature or opts not to. In the former case, the challenger opens the signature and verifies that the signer indeed matches the one used by the CH_b oracle. If the signer does not match the one used by the CH_b oracle, the game aborts. In the guess phase, the adversary is re-granted access to the Open oracle providing that it is not queried on the challenge signature. If the adversary opts not to output the challenge signature, access to the Open oracle remains disabled in the guess phase.

Although the above definition seems reasonable for strongly unforgeable signing protocols, unfortunately, it might be problematic to use in practice. That is because the requirement that the returned signature is compared against the signer used by the CH_b oracle might be hard to fulfill in practice as usually in the security proof one would be reducing an attack against anonymity to an attack against a property of another primitive which might not permit such an operation.

The Definition. In the definition we propose (shown in Figure 2), if the adversary queries the Open oracle on a signature that opens to a signer in the challenge list, the oracle returns a special symbol instead of returning the identity of the signer. This restriction, which is similar to that used for IND-RCCA security [17] for encryption schemes, ensures that the Open oracle does not open a challenge signature. In the game, Experiment $\mathsf{Exp}^{Corr}_{\mathsf{GBS},\mathcal{A}}(\lambda)$

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} := \emptyset$; $\mathsf{CSL} := \emptyset$; $\mathsf{BSL} := \emptyset$.
- $(i, m) \leftarrow \mathcal{A}(\mathsf{gpk} : \mathsf{AddS}(\cdot), \mathsf{ReadReg}(\cdot))$.
- If $i \notin \mathsf{HSL}$ or $\mathbf{gsk}[i] = \epsilon$ Then Return 0.
- $(\Sigma, \perp) \leftarrow \langle \mathsf{Obtain}(\mathsf{gpk}, m), \mathsf{Sign}(\mathbf{gsk}[i]) \rangle$.
- If $\mathsf{GVf}(\mathsf{gpk}, m, \Sigma) = 0$ Then Return 1.
- $(j, \tau) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathfrak{ok}, \mathbf{reg}, m, \Sigma)$; If $i \neq j$ Then Return 1.
- If $\mathsf{Judge}(\mathsf{gpk}, i, \mathbf{spk}[i], m, \Sigma, \tau) \neq 1$ Then Return 1 Else Return 0.

Experiment $\mathsf{Exp}^{Anon\text{-}b}_{\mathsf{GBS},\mathcal{A}}(\lambda)$

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} := \emptyset$; $\mathsf{CSL} := \emptyset$; $\mathsf{BSL} := \emptyset$; $\mathsf{CLA} := \emptyset$.
- $b^* \leftarrow \mathcal{A}^{\langle \cdot, \mathsf{CH}_b(\cdot,\cdot) \rangle^1}(\mathsf{gpk}, \mathfrak{ik} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{ModifyReg}(\cdot,\cdot), \mathsf{Open}(\cdot,\cdot), \mathsf{SSK}(\cdot))$.
- Return b^*.

Experiment $\mathsf{Exp}^{Trace}_{\mathsf{GBS},\mathcal{A}}(\lambda)$

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} := \emptyset$; $\mathsf{CSL} := \emptyset$; $\mathsf{BSL} := \emptyset$.
- $(m, \Sigma) \leftarrow \mathcal{A}(\mathsf{gpk}, \mathfrak{ok} : \mathsf{AddS}(\cdot), \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToI}(\cdot,\cdot), \mathsf{ReadReg}(\cdot), \mathsf{SSK}(\cdot))$.
- If $\mathsf{GVf}(\mathsf{gpk}, m, \Sigma) = 0$ Then Return 0.
- $(i, \tau) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathfrak{ok}, \mathbf{reg}, m, \Sigma)$.
- If $i = 0$ or $\mathsf{Judge}(\mathsf{gpk}, i, \mathbf{spk}[i], m, \Sigma, \tau) = 0$ Then Return 1 Else Return 0.

Experiment $\mathsf{Exp}^{Non\text{-}Frame}_{\mathsf{GBS},\mathcal{A}}(\lambda)$

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} := \emptyset$; $\mathsf{CSL} := \emptyset$; $\mathsf{BSL} := \emptyset$.
- $(id, \{(m_i, \Sigma_i)\}_{i=1}^{l+1}) \leftarrow \mathcal{A}^{\langle \cdot, \mathsf{OSign}(\cdot) \rangle^*}(\mathsf{gpk}, \mathfrak{ok}, \mathfrak{ik} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{ModifyReg}(\cdot,\cdot), \mathsf{SSK}(\cdot))$.
- If $id \notin \mathsf{HSL} \setminus \mathsf{BSL}$ or $\mathbf{gsk}[id] = \epsilon$ Then Return 0.
- If all of the following conditions are satisfied Then Return 1 Else Return 0:
  - $\mathsf{GVf}(\mathsf{gpk}, m_j, \Sigma_j) = 1$ for all $j \in [1, l+1]$.
  - $(id_j, \tau_j) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathfrak{ok}, \mathbf{reg}, m_j, \Sigma_j)$; $id = id_j$ for all $j \in [1, l+1]$.
  - $\mathsf{Judge}(\mathsf{gpk}, id_j, \mathbf{spk}[id_j], m_j, \Sigma_j, \tau_j) = 1$ for all $j \in [1, l+1]$.
  - $\mathcal{A}$ interacted with $\mathsf{OSign}(id)$ no more than l times.
  - $\forall i, j$ where $1 \leq i, j \leq l+1$, we have that if $i \neq j$ then $m_i \neq m_j$.

Experiment $\mathsf{Exp}^{Blind\text{-}b}_{\mathsf{GBS},\mathcal{A}}(\lambda)$

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} := \emptyset$; $\mathsf{CSL} := \emptyset$; $\mathsf{BSL} := \emptyset$; $\mathsf{CLB} := \emptyset$.
- $(m_0, m_1, \mathsf{state}_{\mathsf{find}}) \leftarrow \mathcal{A}_{\mathsf{find}}(\mathsf{gpk}, \mathfrak{ik} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{ReadReg}(\cdot), \mathsf{SSK}(\cdot), \mathsf{Open}(\cdot,\cdot))$.
- $\mathsf{state}_{\mathsf{sign}} \leftarrow \mathcal{A}_{\mathsf{sign}}^{\langle \mathsf{Obtain}(\mathsf{gpk}, m_b), \cdot \rangle^1, \langle \mathsf{Obtain}(\mathsf{gpk}, m_{1-b}), \cdot \rangle^1}(\mathsf{state}_{\mathsf{find}})$.
- Let Σ_b and Σ_{1-b} be the outputs of the honest Obtain algorithm after interactions on m_b and m_{1-b}, respectively.
  If $\Sigma_0 = \perp$ or $\Sigma_1 = \perp$ or $\mathsf{GVf}(\mathsf{gpk}, m_b, \Sigma_b) = 0$ or $\mathsf{GVf}(\mathsf{gpk}, m_{1-b}, \Sigma_{1-b}) = 0$ Then $(\Sigma_0, \Sigma_1) := (\perp, \perp)$
  Else $\mathsf{CLB} := \mathsf{CLB} \cup \{(m_b, \Sigma_b), (m_{1-b}, \Sigma_{1-b})\}$.
- $b^* \leftarrow \mathcal{A}_{\mathsf{guess}}(\mathsf{state}_{\mathsf{sign}}, \Sigma_0, \Sigma_1 : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{ReadReg}(\cdot), \mathsf{SSK}(\cdot), \mathsf{Open}(\cdot,\cdot))$.
- Return b^*.

Fig. 2. Security experiments for dynamic group blind signatures

the adversary can corrupt the Issuer and ask for the secret keys of any signer including those it will use when calling the challenge oracle thus capturing full key exposure attacks.

The definition is appropriate even for the case where the final signature is weakly unforgeable, i.e. given a signature on a message, anyone can generate a new signature on the same message without knowledge of the signing key. WLOG in order to simplify the resulting security proofs, we only allow the adversary a single invocation of the challenge oracle. We show in Appendix A that this is sufficient by showing a reduction from an adversary that invokes the challenge oracle polynomially many times to one which is allowed a single invocation.

Our definition of anonymity also captures unlinkability of signatures. If an adversary can link signatures by the same signer, it can break our notion of anonmity. Note that the adversary in our definition is allowed to learn the secret keys of any group member including the challenge signers it uses in calling the CH_b oracle. Thus, it can produce signatures on behalf of any group member. Therefore, unlike [37], we do not define unlinkability as a separate requirement.

Formally, a dynamic group blind signature scheme GBS is anonymous if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$ have a negligible advantage $\mathsf{Adv}^{Anon}_{\mathsf{GBS},\mathcal{A}}(\lambda)$ where the advantage is defined as follows:

$$\mathsf{Adv}^{Anon}_{\mathsf{GBS},\mathcal{A}}(\lambda) := \left|\Pr[\mathsf{Exp}^{Anon-0}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{Anon-1}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1]\right|.$$

4.3 Traceability

A dynamic group blind signature scheme GBS is traceable if it is always the case that the Opener is able to identify the signer when given a message/signature pair. Also, the honest Opener is able to produce a proof for his claim that will be accepted by the Judge algorithm. Note that in the traceability experiment, we require that the Issuer is honest because a dishonest Issuer will always be able to create dummy signers whose signatures cannot be traced and for that reason the adversary in the traceability game is not given the Issuer's key. In addition, we require that the Opener is partially but not fully corrupt because a fully corrupt Opener can simply refuse to open signatures.

Formally, a dynamic group blind signature scheme GBS is traceable if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$ have a negligible advantage $\mathsf{Adv}^{Trace}_{\mathsf{GBS},\mathcal{A}}(\lambda)$ where the advantage is defined as follows:

$$\mathsf{Adv}^{Trace}_{\mathsf{GBS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{Trace}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1].$$

4.4 Non-Frameability

A dynamic group blind signature scheme GBS is non-frameable if it is impossible to prove that a particular group member produced a signature unless this group member himself has indeed produced the signature. To capture this, we use a similar definition to that used for the unforgeability of blind signatures [34, 44]. The adversary wins if it outputs $l+1$ signatures on $l+1$ distinct messages all signed by the same honest group member but the adversary only asked for l signatures by that group member.

This requirement should hold even if both the Opener and the Issuer are fully corrupt and that is the reason why we give $\mathcal{A}$ access to both $\mathfrak{ik}$ and $\mathfrak{ok}$ keys. Formally, a dynamic group blind signature scheme GBS is non-frameable if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$ have a negligible advantage $\mathsf{Adv}^{Non-Frame}_{\mathsf{GBS},\mathcal{A}}(\lambda)$ where the advantage is defined as follows:

$$\mathsf{Adv}^{Non-Frame}_{\mathsf{GBS},\mathcal{A}}(\lambda) := \Pr[\mathsf{Exp}^{Non-Frame}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1].$$

4.5 Blindness

A dynamic group blind signature scheme GBS is blind if the adversary is unable to tell which message it is signing. Also, the adversary cannot link a signature to the protocol run via which it was obtained. In blind signatures [34], blindness is defined via a game in which the adversary freely chooses two messages (and possibly the signing/verification keys, e.g. [1]), and then after interacting with an honest Obtain oracle that requests signatures on those two messages in an arbitrary order (unknown to the adversary), the adversary wins if it correctly guesses the order in which the two messages were signed.

In the context of group blind signatures, things are a bit different as the group contains more than one signer. To capture the case that group members (including the Issuer) might collude to break blindness, the adversary is allowed to use different (possibly corrupt) keys in producing the challenge signatures. This definition would then also imply the anonymity requirement, i.e. if a malicious signer from the group can recognize a signature he has produced then he can trivially break blindness. Also, unlike [52] which necessitates that the group must be static, we only require that both challenge signatures verify w.r.t. the same group public key. Otherwise, the adversary can trivially break blindness.

In the definition (shown in Figure 2), we equip the adversary with strong capabilities such as corrupting the Issuer as well as corrupting and/or learning the personal secret key/group signing key of any group member. However, the adversary is denied access to the Opener's key. Again, as was the case with the anonymity definition, one can consider different variants where the adversary can ask open queries at the different stages with the exception that it is not allowed to ask open queries on either of the two challenge signatures. Otherwise, blindness can be trivially broken. In CPA-Blindness, the adversary is not granted access to the Open oracle. On the other hand, in Full Blindness (FB), the adversary is granted access to the Open oracle both before and after the challenge phase. When defining full blindness, if security is w.r.t. weak unforgeability then the Open oracle returns a special symbol if the signature is on either challenge message. On the other hand, when security is w.r.t. strong unforgeability, the restriction imposed on the the Open oracle is that it may not open the signature if the message-signature pair matches either of the challenge message-signature pairs.

We require that if either of the challenge interactions does not finish successfully (i.e. if either $\Sigma_0 = \perp$ or $\Sigma_1 = \perp$), then the adversary is not informed about the other signature.

Formally, a dynamic group blind signature scheme GBS is blind if for all $\lambda \in \mathbb{N}$, all PPT adversaries $\mathcal{A}$ have a negligible advantage $\mathsf{Adv}^{Blind}_{\mathsf{GBS},\mathcal{A}}(\lambda)$ where the advantage is defined as follows:

$$\mathsf{Adv}^{Blind}_{\mathsf{GBS},\mathcal{A}}(\lambda) := \left| \Pr[\mathsf{Exp}^{Blind-0}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{Blind-1}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1] \right|.$$

Finally, we note that in [35] the authors used a different approach for capturing blindness in the context of blind ring signatures (i.e. group blind signatures which have neither an opener nor an issuer). In their definition, the adversary is required to only participate in one interaction to produce one challenge signature on either of the two messages it has chosen (without the adversary knowing which message being signed). The adversary then is asked to guess the message being signed relying only on the information it gathered from the interaction, i.e. without knowledge of the resulting challenge signature. While it is clear that such a game captures the requirement that a signer cannot learn the message it is signing, it is unclear how this definition captures the requirement that a signer should also not be able to link signatures to the protocol runs where they were obtained.

5 Tools Used

In this section, we present the building blocks we use in our constructions.

5.1 A New Structure-Preserving Signature Scheme

We introduce a new variant of the CL signature scheme [15] which we refer to as (NCL). It signs group elements and it is structure-preserving [3]. The unforgeability of the new scheme is proven under the DH-LRSW assumption. The scheme is given by the following triple of algorithms. Given the description of bilinear groups $\mathcal{P}$ output by $\mathsf{BGrpSetup}(1^\lambda)$.

- $\mathsf{NCLKeyGen}(\mathcal{P})$: Set $\mathsf{sk_{NCL}} := (x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ and $\mathsf{pk_{NCL}} := (X, Y) := (G_2^x, G_2^y) \in \mathbb{G}_2^2$.
- $\mathsf{NCLSign}(\mathsf{sk_{NCL}}, (M_1, M_2))$: To sign a message $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$, if $\hat{e}(M_1, G_2) \neq \hat{e}(G_1, M_2)$ return $\perp$. Otherwise, select $a \leftarrow \mathbb{Z}_p^*$, and set $A := G_1^a$, $B := A^y$, $C := M_1^{ay}$, $D := A^x \cdot M_1^{axy}$. Output $\sigma := (A, B, C, D) \in \mathbb{G}_1^4$.

– $\mathsf{NCLVerify}(\mathsf{pk}_{\mathsf{NCL}}, (M_1, M_2), \sigma)$: Output 1 iff $A \neq 1_{\mathbb{G}_1} \quad \wedge \quad \hat{e}(A, Y) = \hat{e}(B, G_2) \quad \wedge \quad \hat{e}(B, M_2) = \hat{e}(C, G_2) \quad \wedge \quad \hat{e}(D, G_2) = \hat{e}(A \cdot C, X) \quad \wedge \quad \hat{e}(G_1, M_2) = \hat{e}(M_1, G_2)$.

Theorem 1. *Assuming the DH-LRSW assumption holds, the* NCL *signature scheme is existentially unforgeable against adaptive chosen-message attacks.*

Proof. It is straightforward to see that the unforgeability of the signature scheme reduces to breaking the DH-LRSW assumption.

5.2 Groth-Sahai(GS) Proofs

In [32, 33] Groth and Sahai presented a way to construct non-interactive witness-indistinguishable and zero-knowledge proofs in the CRS model. Groth-Sahai proofs can be instantiated under different security assumptions but since as noted by [27] the most efficient Groth-Sahai proofs are those instantiated under the SXDH assumption, we will be focusing on this instantiation. The equations one can prove with the GS proof system are as follows where in the description $X_1, \ldots, X_m \in \mathbb{G}_1$, $Y_1, \ldots, Y_n \in \mathbb{G}_2$, $x_1, \ldots, x_m, y_1, \ldots, y_n \in \mathbb{Z}_p$ are the secret variables (hence underlined) and $A_i, T_1 \in \mathbb{G}_1$, $B_i, T_2 \in \mathbb{G}_2$, $a_i, b_i, \gamma_{i,j}, t \in \mathbb{Z}_p$, $t_T \in \mathbb{G}_T$ are public constants.

– **Pairing Product Equation (PPE):**

$$\prod_{i=1}^{n} \hat{e}(A_i, \underline{Y_i}) \cdot \prod_{i=1}^{m} \hat{e}(\underline{X_i}, B_i) \cdot \prod_{i=1}^{m} \prod_{j=1}^{n} \hat{e}(\underline{X_i}, \underline{Y_j})^{\gamma_{i,j}} = t_T. \tag{1}$$

– **Multi-Scalar Multiplication Equation in $\mathbb{G}_1$ (MSME1):**

$$\prod_{i=1}^{n} A_i^{\underline{y_i}} \prod_{i=1}^{m} \underline{X_i}^{b_i} \prod_{i=1}^{m} \prod_{j=1}^{n} \underline{X_i}^{\gamma_{i,j} \cdot \underline{y_j}} = T_1. \tag{2}$$

– **Multi-Scalar Multiplication Equation in $\mathbb{G}_2$ (MSME2):**

$$\prod_{i=1}^{n} \underline{Y_i}^{a_i} \prod_{i=1}^{m} B_i^{\underline{x_i}} \prod_{i=1}^{m} \prod_{j=1}^{n} \underline{Y_i}^{\gamma_{i,j} \cdot \underline{x_j}} = T_2. \tag{3}$$

– **Quadratic Equation (QE) in $\mathbb{Z}_p$:**

$$\sum_{i=1}^{n} a_i \cdot \underline{y_i} + \sum_{i=1}^{m} \underline{x_i} \cdot b_i + \sum_{i=1}^{m} \sum_{j=1}^{n} \gamma_{i,j} \cdot \underline{x_i} \cdot \underline{y_j} = t. \tag{4}$$

The proof system can be instantiated in two settings: the binding setting which yields perfectly sound proofs and the hiding setting which yields perfectly witness indistinguishable/perfectly zero-knowledge proofs. The proof system consists of the algorithms

$$\mathsf{GS} := (\mathsf{GSSetup}, \mathsf{GSProve}, \mathsf{GSVerify}, \mathsf{GSExtract}, \mathsf{GSSimSetup}, \mathsf{GSSimProve}).$$

GSSetup takes as input the description of bilinear groups $\mathcal{P}$ and outputs a binding reference string crs and a trapdoor information xk which allows for witness extraction. GSProve takes as input a set of equations, the string crs and a witness and produces a proof Ψ for the satisfiability of the equations. For clarity, we will underline the elements of the witness to distinguish them from public constants. GSVerify takes as input a set of equations, a CRS, a proof Ψ and outputs 1 if the proof is valid or 0 otherwise. In the rest of the paper we will omit the set of equations from the input to the GSVerify algorithm. GSExtract takes as input a binding reference string crs, a valid proof Ψ and the extraction key xk and outputs the witness used in the proof. GSSimSetup takes as input the description of bilinear groups $\mathcal{P}$ and outputs a simulated string, $\mathsf{crs}_{\mathsf{sim}}$, and a trapdoor key tr that allows to simulate proofs. $\mathsf{GSSimProve}$ takes as input a simulated CRS, $\mathsf{crs}_{\mathsf{sim}}$, and the simulation key tr and produces a simulated proof Ψ_{sim}. The security of the proof system requires that strings crs and $\mathsf{crs}_{\mathsf{sim}}$ are indistinguishable and that simulated proofs are indistinguishable from proofs generated by an honest prover.

The proof system has the following properties:

– **Prefect Completeness:** On a correctly generated CRS, crs, and a valid proof Ψ, the algorithm $\mathsf{GSVerify}$ always accepts the proof.
– **Perfect Soundness:** On a correctly generated CRS, crs, it is impossible to generate a proof unless the equations are satisfiable (i.e. unless the prover knows a witness).
– **Composable Witness-Indistinguishability:** The CRS crs output by $\mathsf{GSSetup}$ is computationally indistinguishable from the CRS $\mathsf{crs}_{\mathsf{sim}}$ output by $\mathsf{GSSimSetup}$. We also have for any PPT adversary $\mathcal{A}$ that is given $\mathsf{crs}_{\mathsf{sim}}$ and is allowed to choose any statement y and two distinct witnesses w_0 and w_1 for the statement y that

$$\Pr\begin{bmatrix} b \leftarrow \{0,1\}; \Psi \leftarrow \mathsf{GSProve}(\mathsf{crs}_{\mathsf{sim}}, w_b, y); b' \leftarrow \mathcal{A}(\Psi) \\ : b = b' \wedge (w_0, y) \in R \wedge (w_1, y) \in R \end{bmatrix} = \frac{1}{2} + \nu(\lambda).$$

– **Composable Zero-Knowledge:** Again, the CRS crs output by $\mathsf{GSSetup}$ is computationally indistinguishable from the CRS $\mathsf{crs}_{\mathsf{sim}}$ output by $\mathsf{GSSimSetup}$. Moreover, we have for any PPT adversary $\mathcal{A}$ that

$$\Pr\begin{bmatrix} (\mathsf{crs}_{\mathsf{sim}}, \mathsf{tr}) \leftarrow \mathsf{GSSimSetup}(\mathcal{P}); (w, y) \leftarrow \mathcal{A}(\mathsf{crs}_{\mathsf{sim}}, \mathsf{tr}); \\ \Psi \leftarrow \mathsf{GSProve}(\mathsf{crs}_{\mathsf{sim}}, w, y) : \mathcal{A}(\Psi) = 1 \end{bmatrix}$$
$$= \Pr\begin{bmatrix} (\mathsf{crs}_{\mathsf{sim}}, \mathsf{tr}) \leftarrow \mathsf{GSSimSetup}(\mathcal{P}); (w, y) \leftarrow \mathcal{A}(\mathsf{crs}_{\mathsf{sim}}, \mathsf{tr}); \\ \Psi_{\mathsf{sim}} \leftarrow \mathsf{GSSimProve}(\mathsf{crs}_{\mathsf{sim}}, \mathsf{tr}, y) : \mathcal{A}(\Psi_{\mathsf{sim}}) = 1 \end{bmatrix},$$

where (w, y) output by $\mathcal{A}$ satisfy $(w, y) \in R$.

Randomizable GS Proofs. As noted by [5], GS proofs can be re-randomized by re-randomizing the underlying GS commitments and updating the proofs accordingly so that we obtain fresh proofs that are unlinkable to the original ones. The re-randomization is done without knowledge of neither the witness concealed in the commitments or the associated randomness. We define an algorithm $\mathsf{GSRandomize}$ which takes as input a CRS crs and a proof Ψ and outputs a proof Ψ' which is a randomized version of the proof Ψ.

– **Re-randomizability of GS proofs [5]:** We have for all PPT adversaries $(\mathcal{A}_1, \mathcal{A}_2)$, the following probability is negligbly close to 1/2

$$\Pr\begin{bmatrix} (\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{GSSetup}(\mathcal{P}); (w, y, \Psi, \mathsf{state}_{\mathsf{find}}) \leftarrow \mathcal{A}_1(\mathsf{crs}); \\ \Psi_0 \leftarrow \mathsf{GSProve}(\mathsf{crs}, w, y); \Psi_1 \leftarrow \mathsf{GSRandomize}(\mathsf{crs}, \Psi); b \leftarrow \{0,1\}; \\ b^* \leftarrow \mathcal{A}_2(\mathsf{state}_{\mathsf{find}}, \Psi_b) : b^* = b \ \wedge \ \mathsf{GSVerify}(\mathsf{crs}, \Psi) = 1 \ \wedge \ (w, y) \in R \end{bmatrix}.$$

As shown in [5], GS proofs have composable randomizability, where randomizability holds even if we switch to the hiding setting and give the adversary the simulation trapdoor key tr.

In Appendix B, we show how the GS proof system is instantiated in the SXDH setting.

5.3 Blind Signatures

Blind signatures as introduced by Chaum [18] allow a user to obtain signatures on messages hidden from the signer. The signing protocol in these schemes is interactive $\langle \mathsf{Obtain}(\mathsf{pk}, m), \mathsf{Sign}(\mathsf{sk}) \rangle$ between a user who knows a message m and a signer who possesses a secret signing key sk. If the protocol is completed successfully, the user obtains a signature σ on the message m.

The standard security model for blind signatures [34, 44] consists of two properties: blindness and unforgeability. Intuitively, blindness says that an adversarial signer cannot learn the message being signed and he is unable to match a signature to the protocol run where it was obtained. On the other hand, unforgeability deals with an adversarial user whose goal is to obtain $l + 1$ distinct message/signature pairs given only l interactions with the honest signer.

In [23, 2], the authors presented a blind signature scheme whose unforgeability reduces to the AWFCDH and q-ADHSDH assumptions. The scheme uses GS proofs and is akin to the idea used in Fischlin's generic construction [22]. The signer signs a commitment to the message. However, unlike the generic construction, the user transforms the signature on the commitment to the message into a signature on the message itself instead of proving that he knows a signature on a commitment to the message as in Fischlin's construction. In addition, the user proves knowledge of the message and the randomness used in the commitment when requesting a signature. The message space of the scheme is $\mathcal{M} := \{(G_1^m, G_2^m) | m \in \mathbb{Z}_p\}$.

Exploiting some properties of this blind signature scheme, we will show later how this scheme can be used to hide the identity of the signer while signing hidden messages. The blind signature scheme is given in Figure 3.

```
Setup(1^λ):
  – $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, G_1, G_2) \leftarrow \mathsf{BGrpSetup}(1^\lambda)$.
  – $\mathcal{P} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, G_1, G_2)$.
  – $(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
  – $F, K, T \leftarrow \mathbb{G}_1$.
  – $\mathsf{param} := (\mathcal{P}, F, K, T, \mathsf{crs})$.
  – Return $\mathsf{param}$.

KeyGen(param):
  – Choose $s \leftarrow \mathbb{Z}_p$ and set $S_1 := G_1^s$, $S_2 := G_2^s$.
  – $\mathsf{sk} := s$, $\mathsf{pk} := (S_1, S_2)$. Return $(\mathsf{sk}, \mathsf{pk})$.

Verify(pk, (M_1, M_2), Σ):
  – Parse $\Sigma$ as $\vec{\Omega}$.
  – Return 1 if $\mathsf{GSVerify}(\mathsf{crs}, \vec{\Omega}) = 1$, Else Return 0.

The signing protocol $\langle \mathsf{Obtain}(\mathsf{pk}, (M_1, M_2)), \mathsf{Sign}(\mathsf{sk}) \rangle$

Obtain → Sign
  Choose $q \leftarrow \mathbb{Z}_p$ and compute $Q_i := G_i^q$ for $i = 1, 2$ and $Co := T^q \cdot M_1$.
  $\vec{\Psi} \leftarrow \mathsf{GSProve}\big(\mathsf{crs}, \{M_1, M_2, Q_1, Q_2\}, \{\hat{e}(\underline{M_1}, G_2) = \hat{e}(G_1, \underline{M_2}) \ \wedge$
      $\hat{e}(\underline{Q_1}, G_2) = \hat{e}(G_1, \underline{Q_2}) \ \wedge$
      $\hat{e}(T, \underline{Q_2}) \cdot \hat{e}(\underline{M_1}, G_2) = \hat{e}(Co, G_2)\}\big)$.
  Send $(Co, \vec{\Psi})$ to Sign.

Sign → Obtain
  If $\mathsf{GSVerify}(\mathsf{crs}, \vec{\Psi}) \neq 1$ Then Abort().
  Choose $c, r \leftarrow \mathbb{Z}_p$ and compute $C_1 := F^c$, $C_2 := G_2^c$, $H := (K \cdot T^r \cdot Co)^{\frac{1}{s+c}}$,
      $R_1' := G_1^r$, $R_2' := G_2^r$.
  Send $\sigma := (H, C_1, C_2, R_1', R_2')$ to Obtain.

Obtain
  Compute $R_i := R_i' \cdot Q_i$ for $i = 1, 2$.
  Abort if $\hat{e}(R_1, G_2) \neq \hat{e}(G_1, R_2)$ or $\hat{e}(F, C_2) \neq \hat{e}(C_1, G_2)$
      or $\hat{e}(H, S_2 \cdot C_2) \neq \hat{e}(K \cdot M_1, G_2) \cdot \hat{e}(T, R_2)$.
  $\vec{\Omega} \leftarrow \mathsf{GSProve}\big(\mathsf{crs}, \{H, C_1, C_2, R_1, R_2\}, \{\hat{e}(\underline{R_1}, G_2) = \hat{e}(G_1, \underline{R_2}) \ \wedge$
      $\hat{e}(\underline{C_1}, G_2) = \hat{e}(F, \underline{C_2}) \ \wedge$
      $\hat{e}(\underline{H}, S_2 \cdot \underline{C_2}) \cdot \hat{e}(T^{-1}, \underline{R_2}) = \hat{e}(K \cdot M_1, G_2)\}\big)$.
  Output $\Sigma := \vec{\Omega}$.
```

Fig. 3. The automorphic blind signature scheme from [2]

6 Our Constructions

In this section we present our constructions. We start by listing the techniques we use and then give a general overview of our generic construction before presenting two example instantiations of the construction.

6.1 Techniques Used

We identify and make use of a number of observations and desired properties the building blocks we use have, which allows us to construct a practical dynamic group blind signature scheme that has a round-optimal signing protocol and yet does not rely on random oracles. We summarize the observations as follows:

- We exploit the fact that GS proofs are re-randomizable [5] and given a proof Ψ, we can compute a new fresh proof $\Psi' \leftarrow \mathsf{GSRandomize}(\mathsf{crs}, \Psi)$ even without knowing the witnesses used in Ψ. The new proof Ψ' is unlinkable to the original proof Ψ.
- Groth-Sahai proofs are independent of any public terms (i.e. public monomials) in the equations being proved.[2] Thus, given a witness-indistinguishable GS proof, Ψ, (i.e. one for an equation with a non-trivial public right-hand side τ), we can later (even without knowing the original witnesses in the proof) transform Ψ into a related witness-indistinguishable/zero-knowledge proof Ψ' by splitting the public right-hand side τ into a set of witnesses and adding them to the list of witness in the original proof.
  One can construct GS NIZK proofs for pairing product equations (Equation 1) if either t_T is trivial (i.e. $t_T = 1$) or if one can factor t_T by finding P_i, Q_i such that $t_T = \prod_{i=1}^{n} \hat{e}(P_i, Q_i)$. It is the latter case that applies to our construction as the user knows how to open the commitment to the message and therefore can factor t_T. This will become clearer when we present our construction.
- Another observation which is of independent interest and is analogous to [24] (Lemma 4 for PPE equations), we prove the following lemma for multi-scalar multiplication equations in Appendix D.2.

  Lemma 1. *Let $((\mathcal{C}_z, \mathcal{C}_A), \Psi)$ be a GS proof for the equation $\mathcal{E} := \underline{A}^{\underline{z}} = Z$, where $\mathcal{C}_z, \mathcal{C}_A$ are the GS commitments to $z \in \mathbb{Z}_p$ and $A \in \mathbb{G}$ respectively and $Z \in \mathbb{G}$ then $(({\mathcal{C}_z}^{z'}, \mathcal{C}_A), \Psi^{z'})$ is a GS proof for the equation $\mathcal{E}' := \underline{A}^{\underline{z'\cdot z}} = Z^{z'}$.*

  The same argument holds for quadratic equations over $\mathbb{Z}_p$ where exponentiation of the public right-hand side is replaced by multiplication.
- The blind signatures we use are a set of GS proofs of knowledge of values satisfying pairing product equations which allow us to take advantage of the above properties of GS proofs.

6.2 Overview of the Constructions

The General Idea. In Fischlin's generic construction for blind signatures [22], the user sends a commitment to the message to the signer, who in turn returns a signature on the commitment. The user then computes the blind signature by providing a NIZK proof of knowledge of the signature and the commitment s.t. the signature is valid on the commitment and the commitment is to the message in question.

Now assume in the above framework we want to hide the signer's identity from the user. So instead of sending the signature on the user's commitment (to the message) in the clear, the signer sends a proof of knowledge of such a signature and his verification key to the user. If the proofs used are re-randomizable and transformable in the sense of Observation 1 from Section 6.1, where in the equations used in the signer's proofs, the terms involving the commitment are public, then the user can re-randomize the proofs and transform them to hide the commitment to the message and compute the final blind signature. Unforgeability follows from that of Fischlin's framework, blindness also follows from that of the framework plus the re-randomizability of the proofs, and the anonymity of the signer is ensured by the hiding properties of the proofs. Thus, we obtain blind signatures while the signer remain anonymous.

What remains is to extend the signer-anonymous blind signature to the group setting by requiring a compatible signature scheme to certify signer's keys when they join the group, and also providing an anonymity revocation mechanism for the Opener.

The Construction. We base our signing protocol on the blind signature scheme from [2] (cf. Figure 3), which uses Groth-Sahai proofs and has the required properties needed for our construction. However, the same

[2] This observation was also independently noted by [24].

methodology can similarly be applied to other Groth-Sahai based instantiations of Fischlin's construction satisfying the properties required for our paradigm, e.g. the instantiation by Abe et al. in [3].

In order to issue membership certificates (i.e. credentials) for new group members, we can use any signature scheme that can sign group elements. We will use the prefix CERT for the signature scheme used by the Issuer in generating the membership certificates.

The Issuer gets the secret signing key $\mathsf{sk_{CERT}}$ for the CERT signature scheme. Each potential group member Signer_i would have created his pair of personal secret/public keys $(\mathbf{ssk}[i], \mathbf{spk}[i])$ prior to joining the group. When requesting to join the group, Signer_i generates a pair of secret signing/public verification keys $(\mathsf{sk}_i, \mathsf{pk}_i)$ for the blind signature scheme. To stop a corrupt Issuer from framing group members, we ask that the group member signs his verification key pk_i with his personal secret key $\mathbf{ssk}[i]$, the resulting signature sig_i will be used as a proof when verifying the Opener's claim. We will use the CERT scheme for this purpose as well. Thus, WLOG we assume that the key pair $(\mathbf{ssk}[i], \mathbf{spk}[i])$ is a valid pair for the CERT scheme.

The Issuer first verifies the signature sig_i and if it is valid, he issues a signature on pk_i using the CERT signature scheme and his secret issuing key $\mathfrak{ik}$. Upon successful completion of the $\langle \mathsf{Join}, \mathsf{Issue} \rangle$ protocol, Signer_i's secret group signing key $\mathbf{gsk}[i]$ is $(\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{cert}_i)$, where cert_i is the Issuer's signature on pk_i. The registration information for Signer_i is set to $\mathbf{reg}[i] := (\mathsf{pk}_i, \mathsf{sig}_i)$.

The group public key gpk contains the public key of the CERT signature scheme, the public values used in the blind signature scheme and two GS reference strings crs_1 and crs_2, which are used in constructing GS proofs used in the first and second rounds of the signing protocol, respectively. Note that although we could use the same GS reference string for both rounds, we use separate strings because we believe this provides extra functionality such as preventing the opener from revoking the anonymity of the message in the signing phase or to allow for having a different opener for revoking anonymity of the message if needed. To open signatures, the Opener is given the extraction key for the GS proof system.

The signing protocol $\langle \mathsf{Obtain}, \mathsf{Sign} \rangle$ between an external user who knows a secret message $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ and an anonymous group member Signer_i who possesses a secret group signing key $\mathbf{gsk}[i]$ consists of two rounds. In the first round, the user commits to the message using $\mathsf{Pedersen}$ commitment $Co := T^q \cdot M_1$ for some random $q \leftarrow \mathbb{Z}_p$ and computes $Q_i := G_i^q$ for i=1,2. He then sends the commitment Co along with GS proofs of knowledge $\vec{\Psi}$ to prove that: the commitment Co is indeed to the message M_1 and that the message and the randomness pairs are well-formed.

In the second round of the protocol, Signer_i verifies the GS proofs and if they are valid, produces a signature using the blind signature scheme defined in Figure 3. The signature σ on the commitment Co contains the components $(H, C_1, C_2, R'_1, R'_2)$. Now since we require that the signer remains anonymous, we hide the parts of the signature which identify the signer. Thus, the signer commits to his membership certificate cert_i, his public verification key pk_i and the components (H, C_1, C_2) of the signature σ. Note that the components R'_1 and R'_2 of σ do not identify the signer and hence at this phase we can send them in the clear to the user.

The signer's response consists of GS proofs $\vec{\Omega}$ to prove that: the signer has a valid certificate he got from the Issuer on his public verification key pk_i, his public verification key pk_i is well-formed, and the signature was produced by the owner of public key pk_i, and the components R'_1, R'_2 of σ.

The user first verifies the GS proofs $\vec{\Omega}$. If they are valid, the user first re-randomizes those proofs using the algorithm $\mathsf{GSRandomize}$. The new proofs are unlinkable to the original ones. Note here that the last equation proven in $\vec{\Omega}$ is $\hat{e}(\underline{H}, \underline{S_2} \cdot \underline{C_2}) = \hat{e}(K \cdot Co, G_2) \cdot \hat{e}(T, R'_2)$, where Co and R'_2 are public at this stage and hence not part of the witness.

The user then updates the components R'_1 and R'_2 to include the randomness used in the commitment (by computing $R_1 := R'_1 \cdot Q_1$ and $R_2 := R'_2 \cdot Q_2$) and then decomposes the commitment and transforms the proof for the last equation in $\vec{\Omega}$ into a proof based on the message M_1 instead of the commitment Co by adding the value R_2 to the witness. The transformation is done without knowledge of H, C_2 or S_2. The proof is now for the equation $\hat{e}(\underline{H}, \underline{S_2} \cdot \underline{C_2}) \cdot \hat{e}(T^{-1}, \underline{R_2}) = \hat{e}(K \cdot M_1, G_2)$, where H, S_2, C_2 and R_2 are parts of the witness. In addition, the user adds to $\vec{\Omega}'$ a new GS proof to prove that R_1 and R_2 hide the same exponent. The final signature Σ is a set of GS proofs of knowledge $\vec{\Omega}'$ to prove that: the group member has a valid credential from the Issuer, his public key is well-formed and the blind signature verifies w.r.t. his key.

To open a signature, the Opener uses his secret extraction key to extract the verification key pk, the signature σ and the membership certificate cert from the proofs. Besides those, the Opener returns the index i of the group member and the signature sig in support of his claim. The Judge algorithm can verify the correctness of the Opener's decision by verifying those components and checking that the group member has indeed signed the key pk with his secret key $\mathbf{ssk}[i]$. The construction is illustrated in Figure 4.

Next we present two example instantiations of the construction. The first instantiation is less efficient but its security relies solely on non-interactive assumptions which are falsifiable [39]. The second instantiation is more efficient as it makes use of the new efficient structure-preserving signature scheme that we construct but at the expense of basing traceability on an interactive complexity assumption. Both constructions achieve CPA-anonymity and we outline in Section 7 how to extend them to achieve full anonymity.

6.3 Instantiation I

In this construction we will use the asymmetric automorphic signature scheme (AFPV) (underlying the blind signature scheme in Figure 3) from [23], which has the extra beneficial property of being able to sign its own public keys, to instantiate the CERT scheme. Although this will result in a less efficient construction than Instantiation II in Section 6.4, we get the bonus of basing the security of the construction on non-interactive complexity assumptions which are falsifiable [39].

6.4 Instantiation II

To get better efficiency, we instantiate the CERT scheme using the new NCL scheme (Section 5.1). Since in our construction, the final group blind signature hides the components of the certificate and hence one cannot directly verify that the certificate is non-trivial (i.e. that $A \neq 1_{\mathbb{G}_1}$), we require that the signer additionaly proves this statement. Otherwise, the adversary can create untraceable signatures by faking trivial certificates. We suggest two re-randomizable proofs in Appendix D to achieve this. Despite the need for this extra proof, this construction is still more efficient than construction I.

6.5 Efficiency Analysis

Assuming that GS proofs are instantiated in the SXDH setting, we get the following efficiency:

Instantiation I. The final signature consists of seven GS proofs $\vec{\Omega}' = (\Omega'_i)_{i=1}^{7}$: proofs $(\Omega'_1, \Omega'_2, \Omega'_3)$ to prove that the member has a valid membership certificate, proof Ω'_4 to prove that his key is well-formed and proofs $(\Omega'_5, \Omega'_6, \Omega'_7)$ to prove that the blind signature verifies w.r.t. his public key. All of those proofs are quadratic PPE proofs and hence each of size $2 \cdot \mathbb{G}_1^2 \times 2 \cdot \mathbb{G}_2^2$. Thus, the total size of the proofs is $28 \cdot |\mathbb{G}_1| + 28 \cdot |\mathbb{G}_2|$. The total number of GS commitments used in those proofs are 7 commitments in group $\mathbb{G}_1^2$ and 5 commitments in group $\mathbb{G}_2^2$ and therefore the total size of the signature is $42 \cdot |\mathbb{G}_1| + 38 \cdot |\mathbb{G}_2|$.

Instantiation II. The final signature Σ consists of eight GS proofs $\vec{\Omega}' = (\Omega'_i)_{i=0}^{7}$, where Ω'_0 is Protocol II (cf. Appendix D.2) to prove that $A \neq 1$ which is of size $\mathbb{G}_1^2 \times 2 \cdot \mathbb{G}_2^2$. Two of those proofs (i.e. proofs Ω'_1 and Ω'_3) are for linear equations where all elements of the witness lie in the same group. Thus, we have $\Omega'_i = (\vec{\theta}_i, \vec{\pi}_i) \in 2 \cdot \mathbb{G}_1^2 \times 2 \cdot \mathbb{G}_2^2$ for $i = 2, 4, 5, 6, 7$ and $\Omega'_i = \pi_i \in \mathbb{G}_2^2$ for $i = 1, 3$ and hence the total size of the proofs is $22 \cdot |\mathbb{G}_1| + 28 \cdot |\mathbb{G}_2|$. The total number of GS commitments used in those proofs are 8 commitments in group $\mathbb{G}_1^2$ and 4 commitments in group $\mathbb{G}_2^2$ and therefore the total size of the signature is $38 \cdot |\mathbb{G}_1| + 36 \cdot |\mathbb{G}_2|$.

6.6 Security Analysis

Here we prove the security of our constructions.

Theorem 2. *The generic construction in Figure 4 is a secure group blind signature scheme providing that the* CERT *scheme is unforgeable, the GS proof system is sound, hiding (i.e. witness-indistinguishable/zero-knowledge) and re-randomizable and the blind signature scheme is secure (i.e. unforgeable and blind).*

```
$\underline{\mathsf{GKg}(1^\lambda)}$
– $\mathcal{P} \leftarrow \mathsf{BGrpSetup}(1^\lambda)$.
– $F, K, T \leftarrow \mathbb{G}_1$.
– $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
– $(\mathsf{crs}_2, \mathsf{xk}_2) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
– $(\mathsf{sk}_{\mathsf{CERT}}, \mathsf{pk}_{\mathsf{CERT}}) \leftarrow \mathsf{CERTKeyGen}(\mathcal{P})$.
– $\mathsf{gpk} := (\mathcal{P}, \mathsf{crs}_1, \mathsf{crs}_2, F, K, T, \mathsf{pk}_{\mathsf{CERT}})$.
– $\mathfrak{ik} := \mathsf{sk}_{\mathsf{CERT}}$; $\mathfrak{ok} := \mathsf{xk}_2$.
– Return $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok})$.
$\underline{\mathsf{SKg}(\mathcal{P})}$:
– $(\mathbf{ssk}[i], \mathbf{spk}[i]) \leftarrow \mathsf{CERTKeyGen}(\mathcal{P})$.
– Return $(\mathbf{ssk}[i], \mathbf{spk}[i])$.
$\underline{\mathsf{GVf}(\mathsf{gpk}, (M_1, M_2), \Sigma)}$:
– Parse $\Sigma$ as $\vec{\Omega}$.
– Return 1 if $\mathsf{GSVerify}(\mathsf{crs}_2, \vec{\Omega}) = 1$.
   Else Return 0.
```

```
$\underline{\mathsf{Open}(\mathsf{gpk}, \mathfrak{ok}, (M_1, M_2), \Sigma)}$
– Parse $\mathfrak{ok}$ as $\mathsf{xk}_2$.
– Parse $\mathsf{gpk}$ as $(\mathcal{P}, \mathsf{crs}_1, \mathsf{crs}_2, F, K, T, \mathsf{pk}_{\mathsf{CERT}})$.
– $(\sigma, \mathsf{cert}, \mathsf{pk}) \leftarrow \mathsf{GSExtract}(\mathsf{crs}_2, \mathsf{xk}_2, \Sigma)$.
– If $\exists i$ s.t. $\mathbf{reg}[i].\mathsf{pk} = \mathsf{pk}$ Then
      Return $(i, (\sigma, \mathsf{cert}, \mathsf{pk}, \mathbf{reg}[i].\mathsf{sig}))$
   Else Return $(0, (\sigma, \mathsf{cert}, \mathsf{pk}, \epsilon))$.
$\underline{\mathsf{Judge}(\mathsf{gpk}, i, \mathbf{spk}[i], (M_1, M_2), \Sigma, \tau)}$
– Parse $\mathsf{gpk}$ as $(\mathcal{P}, \mathsf{crs}_1, \mathsf{crs}_2, F, K, T, \mathsf{pk}_{\mathsf{CERT}})$.
– Parse $\tau$ as $(i, (H, C_j, R_j), \mathsf{cert}, \mathsf{pk}, \mathsf{sig})$ for $j = 1, 2$.
– Parse $\mathsf{pk}$ as $(S_1, S_2)$.
– If $i > 0$ and $\hat{e}(H, S_2 \cdot C_2) = \hat{e}(K \cdot M_1, G_2) \cdot \hat{e}(T, R_2)$
      and $\hat{e}(R_1, G_2) = \hat{e}(G_1, R_2)$ and $\hat{e}(F, C_2) = \hat{e}(C_1, G_2)$
      and $\mathsf{CERTVerify}(\mathbf{spk}[i], \mathsf{pk}, \mathsf{sig}) = 1$
      and $\mathsf{CERTVerify}(\mathsf{pk}_{\mathsf{CERT}}, \mathsf{pk}, \mathsf{cert}) = 1$
      Then Return 1 Else Return 0.
```

$\underline{\mathsf{Join}(\mathsf{gpk}, i, \mathbf{ssk}[i])}$		$\underline{\mathsf{Issue}(\mathfrak{ik}, i, \mathbf{spk}[i])}$
$s \leftarrow \mathbb{Z}_p$, $\mathsf{sk}_i := s$, $\mathsf{pk}_i := (S_1 := G_1^s,\ S_2 := G_2^s)$.		
$\mathsf{sig}_i \leftarrow \mathsf{CERTSign}(\mathbf{ssk}[i], \mathsf{pk}_i)$.	$\xrightarrow{\mathsf{sig}_i, \mathsf{pk}_i}$	Parse pk_i as (S_1, S_2).
		Abort if $\mathsf{pk}_i = \mathsf{pk}_j$ for any j or $\hat{e}(S_1, G_2) \neq \hat{e}(G_1, S_2)$.
		If $\mathsf{CERTVerify}(\mathbf{spk}[i], \mathsf{pk}_i, \mathsf{sig}_i) = 0$ Then Abort.
Abort if $\mathsf{CERTVerify}(\mathsf{pk}_{\mathsf{CERT}}, \mathsf{pk}_i, \mathsf{cert}_i) = 0$.	$\xleftarrow{\mathsf{cert}_i}$	$\mathsf{cert}_i \leftarrow \mathsf{CERTSign}(\mathfrak{ik}, \mathsf{pk}_i)$.
$\mathbf{gsk}[i] := (\mathsf{sk}_i, \mathsf{pk}_i, \mathsf{cert}_i)$.		$\mathbf{reg}[i] := (\mathsf{pk}_i, \mathsf{sig}_i)$.

The signing protocol $\langle \mathsf{Obtain}(\mathsf{gpk}, (M_1, M_2)), \mathsf{Sign}(\mathbf{gsk}[i]) \rangle$

```
$\underline{\mathsf{Obtain} \rightarrow \mathsf{Sign}}$: Choose $q \leftarrow \mathbb{Z}_p$ and set $Q_1 := G_1^q$, $Q_2 := G_2^q$ and $Co := T^q \cdot M_1$.
    $\vec{\Psi} \leftarrow \mathsf{GSProve}\big(\mathsf{crs}_1, \{M_1, M_2, Q_1, Q_2\}, \big\{\hat{e}(\underline{M_1}, G_2) = \hat{e}(G_1, \underline{M_2}) \ \wedge$
        $\hat{e}(\underline{Q_1}, G_2) = \hat{e}(G_1, \underline{Q_2}) \ \wedge$
        $\hat{e}(T, \underline{Q_2}) \cdot \hat{e}(\underline{M_1}, G_2) = \hat{e}(Co, G_2)\big\}\big)$.
    Send $(Co, \vec{\Psi})$ to $\mathsf{Sign}$.
$\underline{\mathsf{Sign} \rightarrow \mathsf{Obtain}}$: If $\mathsf{GSVerify}(\mathsf{crs}_1, \vec{\Psi}) \neq 1$ Then Abort().
    Choose $r, c \leftarrow \mathbb{Z}_p$, set $H := (K \cdot T^r \cdot Co)^{\frac{1}{s+c}}$, $C_1 := F^c$ and $C_2 := G_2^c$, $R_1' := G_1^r$, $R_2' := G_2^r$.
    Set $\sigma := (H, C_1, C_2, R_1', R_2')$ and parse $\mathsf{pk}_i$ as $(S_1, S_2)$.
    Compute $\vec{\Omega}' \leftarrow \mathsf{GSProve}\,(\mathsf{crs}_2, \{\mathsf{cert}, S_1, S_2, H, C_1, C_2\},$
        $\big\{\mathsf{CERTVerify}(\mathsf{pk}_{\mathsf{CERT}}, (\underline{S_1}, \underline{S_2}), \underline{\mathsf{cert}}) = 1 \ \wedge$
        $\hat{e}(\underline{S_1}, G_2) = \hat{e}(G_1, \underline{S_2}) \ \wedge$
        $\hat{e}(\underline{C_1}, G_2) = \hat{e}(F, \underline{C_2}) \ \wedge$
        $\hat{e}(\underline{H}, \underline{S_2} \cdot \underline{C_2}) = \hat{e}(K \cdot Co, G_2) \cdot \hat{e}(T, R_2')\big\}\big)$.
    Send $\big(R_1', R_2', \vec{\Omega}'\big)$ to $\mathsf{Obtain}$.
$\underline{\mathsf{Obtain}}$: If $\mathsf{GSVerify}(\mathsf{crs}_2, \vec{\Omega}') \neq 1$ or $\hat{e}(R_1', G_2) \neq \hat{e}(G_1, R_2')$ Then Abort().
    Set $R_i := R_i' \cdot Q_i$ for $i = 1, 2$.
    $\vec{\Omega}' \leftarrow \mathsf{GSRandomize}(\mathsf{crs}_2, \vec{\Omega}')$.
    Modify[2] $\vec{\Omega}'$ to $\vec{\Omega} \leftarrow \mathsf{GSProve}\,(\mathsf{crs}_2, \{\mathsf{cert}_i, S_1, S_2, H, C_1, C_2, R_1, R_2\},$
        $\big\{\mathsf{CERTVerify}(\mathsf{pk}_{\mathsf{CERT}}, (\underline{S_1}, \underline{S_2}), \underline{\mathsf{cert}}) = 1 \ \wedge$
        $\hat{e}(\underline{S_1}, G_2) = \hat{e}(G_1, \underline{S_2}) \ \wedge$
        $\hat{e}(\underline{C_1}, G_2) = \hat{e}(F, \underline{C_2}) \ \wedge$
        $\hat{e}(\underline{H}, \underline{S_2} \cdot \underline{C_2}) \cdot \hat{e}(T^{-1}, \underline{R_2}) = \hat{e}(K \cdot M_1, G_2) \ \wedge$
        $\hat{e}(\underline{R_1}, G_2) = \hat{e}(G_1, \underline{R_2})\big\}\big)$.
    Output $\Sigma := \vec{\Omega}$.
```

[2] The transformation is done without knowledge of the original witness of the proof.

Fig. 4. The Construction

Proof. Correctness follows from that of the building blocks and is straightforward to verify.

Lemma 2. *The construction is anonymous (against full key exposure) providing that GS proofs are hiding (i.e. witness-indistinguishable/zero-knowledge) and that the SXDH assumption holds.*

Proof. We prove that
$$\mathsf{Adv}^{Anon}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{Hiding}_{\mathsf{GS},\mathcal{A}_1}(\lambda) + \mathsf{Adv}^{SXDH}_{\mathcal{A}_2}(\lambda).$$

By the perfect witness-indistinguishability/zero-knowledge of GS proofs in the hiding setting we have that $\mathsf{Adv}^{Hiding}_{\mathsf{GS},\mathcal{A}_1}(\lambda) = 0$. Also, the adversary has a negligible advantage (by the security of the SXDH assumption) in telling apart a binding CRS from a hiding one [33] and therefore this only negligibly changes $\mathcal{B}$ success probability and hence the scheme is anonymous. The reduction to the SXDH assumption was proven in [33] and hence we skip it.

We will use adversary $\mathcal{B}$ to construct an adversary $\mathcal{A}_1$ which can distinguish between witnesses of GS proofs/simulated proofs and real proofs in the hiding setting and hence break the witness-indistinguishabiliy/zero-knowledge properties of the GS proof system.

The algorithm $\mathcal{A}_1$ is shown in Figure 5(a)[3]. Adversary $\mathcal{A}_1$ has access to a $\mathsf{Prove}(\cdot,\cdot,\cdot)$ oracle which when given two possible witnesses w_0 and w_1, generates a set of proofs that involve one of the two witnesses and $\mathcal{A}$ wins its game if it could tell which witness was used in the proofs (or given a proof and it is required to tell whether the proof is real or simulated in the case of zero-knowledge). The success probability of adversary $\mathcal{A}_1$ is given by $\mathsf{Adv}^{Hiding}_{\mathsf{GS},\mathcal{A}_1}(\lambda) := \left|\Pr[\mathsf{Exp}^{Hiding-0}_{\mathsf{GS},\mathcal{A}_1}(\lambda) = 1] - \Pr[\mathsf{Exp}^{Hiding-1}_{\mathsf{GS},\mathcal{A}_1}(\lambda) = 1]\right|$, where the experiment $\mathsf{Exp}^{Hiding-b}_{\mathsf{GS},\mathcal{A}_1}$ is defined in Figure 5(b).

Adversary $\mathcal{A}_1$ starts out by creating the group public key gpk, where $\mathcal{A}_1$ gets crs_2 from its environment, chooses the Issuer's key $\mathfrak{ik}$ and generates the rest of gpk itself. It then calls $\mathcal{B}$ with input $(\mathsf{gpk}, \mathfrak{ik} := \mathsf{sk}_{\mathsf{CERT}})$.

All $\mathcal{B}$'s queries are answered by adversary $\mathcal{A}_1$ as in Figure 1 except the following queries:

- CH_b: Adversary $\mathcal{A}_1$ will use the group signing keys of members Signer_{i_0} and Signer_{i_1} (i.e. $\mathbf{gsk}[i_0]$ and $\mathbf{gsk}[i_1]$) to produce two signatures σ_0 by signer Signer_{i_0} and σ_1 by Signer_{i_1}. In producing those two challenge signatures, $\mathcal{A}$ will use the same randomness r to construct both signatures and hence the same R'_1, R'_2 components are used in both signatures. This does not give away any information about the identity of the signer because R'_1 and R'_2 are completely independent of the signer's key. Adversary $\mathcal{A}_1$ then forwards $(\mathbf{gsk}[i_0], \mathbf{gsk}[i_1], Co, \sigma_0, \sigma_1, R'_1, R'_2)$ to its Prove oracle and receives a proof $\vec{\Omega}'$. $\mathcal{A}_1$ then forwards $(R_1, R_2, \vec{\Omega}')$ to $\mathcal{B}$.
- Open: Our scheme is CPA-anonymous and hence this oracle is not available to the adversary. We outline in Section 7 how to make our scheme fully anonymous.

Finally, when $\mathcal{B}$ outputs its guess for the bit b^*, $\mathcal{A}_1$ returns this as its answer in the $\mathsf{Exp}^{Hiding-b}_{\mathsf{GS},\mathcal{A}_1}$ game. Clearly, if $\mathcal{B}$ wins its game then $\mathcal{A}_1$ breaks the WI/ZK properties of the proof system.

Lemma 3. *The construction is traceable providing that the* CERT *scheme is unforgeable and GS proofs are sound.*

Proof. We prove that
$$\mathsf{Adv}^{Trace}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{Unforg}_{\mathsf{CERT},\mathcal{A}_1}(\lambda) + \mathsf{Adv}^{Soundness}_{\mathsf{GS},\mathcal{A}_2}(\lambda).$$

Intuitively, the only way that adversary $\mathcal{B}$ can create an untraceable signature is by either forging a new certificate for a signer that has not joined the group or by faking GS proofs. GS proofs are perfectly sound and therefore $\mathsf{Adv}^{Soundness}_{\mathsf{GS},\mathcal{A}_2}(\lambda) = 0$. Thus, we have $\mathsf{Adv}^{Trace}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{Unforg}_{\mathsf{CERT},\mathcal{A}_1}(\lambda)$. Using adversary $\mathcal{B}$ that wins the traceability game, we can construct an adversary $\mathcal{A}_1$ that breaks the unforgeability of the CERT signature scheme. Since we have that $\mathsf{Adv}^{Unforg}_{\mathsf{CERT},\mathcal{A}_1}(\lambda) \leq \nu(\lambda)$, we also have that $\mathsf{Adv}^{Trace}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \nu(\lambda)$ and therefore the scheme is traceable. The Opener uses the proof system's extraction key xk_2 to open signatures

[3] For the zero-knowledge property, we assume that $w_1 = w_0$ and that Prove_1 calls $\mathsf{SimProve}$.

```
Algorithm: $\mathcal{A}_1(\mathcal{P}, \mathsf{crs}_2 : \mathsf{Prove}_b(\cdot,\cdot,\cdot))$
  – $(\mathsf{sk}_{\mathsf{CERT}}, \mathsf{pk}_{\mathsf{CERT}}) \leftarrow \mathsf{CERTKeyGen}(\mathcal{P})$.
  – $F, K, T \leftarrow \mathbb{G}_1$.
  – $(\mathsf{crs}_1, \mathsf{xk}_1) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
  – $\mathsf{gpk} := (\mathcal{P}, \mathsf{crs}_1, \mathsf{crs}_2, F, K, T, \mathsf{pk}_{\mathsf{CERT}})$; $\mathfrak{ik} := \mathsf{sk}_{\mathsf{CERT}}$.
  – $b^* \leftarrow \mathcal{B}^{\langle\cdot,\mathsf{CH}_b(\cdot,\cdot)\rangle^1}(\mathsf{gpk}, \mathfrak{ik} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot),$
                                        $\mathsf{ModifyReg}(\cdot,\cdot), \mathsf{SSK}(\cdot))$.
  – Return $b^*$.
```

```
Experiment: $\mathsf{Exp}^{Hiding-b}_{\mathsf{GS},\mathcal{A}}(\lambda)$
  – $\mathcal{P} \leftarrow \mathsf{BGrpSetup}(1^\lambda)$.
  – $(\mathsf{crs}, \mathsf{tr}) \leftarrow \mathsf{GSSimSetup}(\mathcal{P})$.
  – $(y, w_0, w_1, \mathsf{state}_{\mathsf{find}}) \leftarrow \mathcal{A}_{\mathsf{find}}(\mathsf{crs}, \mathcal{P})$.
  – If $(w_0, y) \notin R$ or $(w_1, y) \notin R$ Then Return $\perp$.
  – $\Pi_b \leftarrow \mathsf{Prove}_b(y, w_0, w_1)$.
  – $b^* \leftarrow \mathcal{A}_{\mathsf{guess}}(\mathsf{state}_{\mathsf{find}}, \Pi_b)$.
  – Return $b^*$.
```

Fig. 5. Adversary $\mathcal{A}$ against the hiding property of the GS proof system (left) and security experiment for GS proof system hiding property (right)

and since we are using GS proofs in the binding setting we are guaranteed to be able to extract a satisfying witness from the proofs.

Adversary $\mathcal{A}_1$ gets as its input the public key $\mathsf{pk}_{\mathsf{CERT}}$ of the CERT scheme from its game. It then runs the GKg algorithm and initiates GS proofs in the binding setting (i.e. crs_2 is chosen to be a binding CRS), except that the Issuer's public key $\mathsf{pk}_{\mathsf{CERT}}$ is set to be its own input.

When calling $\mathcal{B}$, $\mathcal{A}_1$ sends gpk and $\mathfrak{ok}$. All $\mathcal{B}$'s queries are answered as in Figure 1 except the following queries:

– AddS and SndToI: Here $\mathcal{A}_1$ simulates the Issuer, where it forwards $\mathsf{pk}_i := (S_{i,1}, S_{i,2})$ to its sign oracle and uses the output of the oracle as the certificate cert_i for group member Signer_i.

Finally, when $\mathcal{B}$ outputs its signature Σ, adversary $\mathcal{A}_1$ extracts the certificate cert and the verification key $\mathsf{pk} := (S_1, S_2)$ from the signature Σ and returns $((S_1, S_2), \mathsf{cert})$ as its forgery in its game.

Clearly, if $\mathcal{B}$ wins its game then $\mathcal{A}_1$ breaks the unforgeability of the CERT signature scheme since having a group blind signature produced by a signer who has not joined the group corresponds to a forgery in the CERT unforgeability game.

Lemma 4. *The construction is non-frameable if the blind signature scheme is unforgeable (i.e. If assumptions AWFCDH and q-ADHSDH hold and GS proofs are sound) and the* CERT *signature scheme (used for producing* sig *upon joining the group) is unforgeable.*

Proof. We prove that

$$\mathsf{Adv}^{Non-Frame}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq n(\lambda) \cdot (\mathsf{Adv}^{Unforg}_{\mathsf{BS-DS},\mathcal{A}_1}(\lambda) + \mathsf{Adv}^{Unforg}_{\mathsf{CERT},\mathcal{A}_2}(\lambda)) + \mathsf{Adv}^{Soundness}_{\mathsf{GS},\mathcal{A}_3}(\lambda),$$

where $n(\lambda)$ is a polynomial in λ representing the maximum number of honest group members adversary $\mathcal{B}$ creates during the game. By the perfect soundness of GS proofs, we have that $\mathsf{Adv}^{Soundness}_{\mathsf{GS},\mathcal{A}_3}(\lambda) = 0$.

Also, the signature scheme $\mathsf{BS-DS}$ used in the signing protocol and the CERT signature scheme used for producing the signature sig upon joining are unforgeable and therefore adversaries against the unforgeability of those schemes have a negligible advantage. Thus, we have that $\mathsf{Adv}^{Unforg}_{\mathsf{BS-DS},\mathcal{A}_1}(\lambda) \leq \nu(\lambda)$, $\mathsf{Adv}^{Unforg}_{\mathsf{CERT},\mathcal{A}_2}(\lambda) \leq \nu(\lambda)$ and therefore have that $\mathsf{Adv}^{Non-Frame}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \nu(\lambda)$.

The details of the security proof for the unforgeability of the underlying blind signature scheme can be found in [23]. We will construct an adversary $\mathcal{A}_1$ that launches a successful chosen-message attack against the unforgeability of the signature scheme $\mathsf{BS-DS}$ underlying the blind signature scheme where $\mathcal{A}_1$ uses $\mathcal{B}$ as an oracle to achieve its goal. Adversary $\mathcal{A}_1$ has access to a sign oracle $\mathsf{SSign}(\mathsf{sk}, \cdot)$ in its unforgeability game. It starts out by running the algorithm GKg to generate the group public key gpk, where it initiates GS proofs in the binding setting (i.e. crs_2 is chosen to be a binding CRS). Then it calls $\mathcal{B}$ on input gpk, the Issuer's key $\mathfrak{ik} := \mathsf{sk}_{\mathsf{CERT}}$, and the Opener's key $\mathfrak{ok} := \mathsf{xk}_2$.

Let j be the identity of t^{th} honest signer $\mathcal{B}$ creates, where $t \in [1, n(\lambda)]$. Adversary $\mathcal{A}_1$ guesses that j is the identity of the group member $\mathcal{B}$ will attempt to frame. All $\mathcal{B}$'s queries are answered as in Figure 1 except the following queries:

– SndToS: For all honest group members Signer_i, $\mathcal{A}_1$ will generate the personal secret/public keys ($\mathbf{ssk}[i]$, $\mathbf{spk}[i]$) itself. Adversary $\mathcal{A}_1$ will also generate all the signing/verification keys $(\mathsf{sk}_i, \mathsf{pk}_i)$ for all honest group members except for Signer_j where pk_j is set to the public key that $\mathcal{A}_1$ gets from its SSign oracle in its unforgeability game and thus $\mathsf{sk}_j = \perp$ because it corresponds to the secret key available to $\mathcal{A}_1$'s SSign oracle which is unknown to $\mathcal{A}_1$.
– SSK: Is answered as in Figure 1 with the exception that $\mathcal{B}$ is not allowed to ask the query $\mathsf{SSK}(j)$ i.e. it cannot ask for the secret signing key of the group member it intends to frame.
– OSign: For a group member Signer_i where $i \neq j$, $\mathcal{A}_1$ will use $\mathbf{gsk}[i]$ to generate the group blind signature Σ. For a group member Signer_i where $i = j$, $\mathcal{A}_1$ will forward the request to its SSign oracle and gets σ back and then generates the rest of the group blind signature Σ itself. It then forwards the signature Σ to $\mathcal{B}$.

Finally, when $\mathcal{B}$ outputs its $l+1$ signatures, adversary $\mathcal{A}_1$ aborts if the framed group member is different from the one it has guessed. Otherwise, $\mathcal{A}_1$ uses the extraction key xk_2 to extract the extra signature $\sigma^* := (H^*, C_1^*, C_2^*, R_1^*, R_2^*)$ on the message (M_1^*, M_2^*) that it did not query its oracle on and returns σ^* and (M_1^*, M_2^*) as its output in its unforgeability game.

We now turn to constructing an adversary $\mathcal{A}_2$ that launches a successful chosen-message attack against the unforgeability of the signature scheme CERT used for producing the signature sig in the Join protocol. Adversary $\mathcal{A}_2$ has access to a sign oracle $\mathsf{SSign}(\mathsf{sk}, \cdot)$ in its unforgeability game. It starts out by running the algorithm GKg to generate the group public key gpk, where it initiates GS proofs in the binding setting (i.e. crs_2 is chosen to be a binding CRS). Adversary $\mathcal{A}_2$ calls $\mathcal{B}$ on input gpk, the Issuer's key $\mathfrak{ik} := \mathsf{sk}_{\mathsf{CERT}}$, and the Opener's key $\mathfrak{ok} := \mathsf{xk}_2$. Again, $\mathcal{A}_2$ guesses that j is the identity of the group member that $\mathcal{B}$ will attempt to frame. All $\mathcal{B}$'s queries are answered as in Figure 1 except the following queries:

– SndToS: For all honest group members Signer_i, $\mathcal{A}_2$ will generate the signing/verification keys $(\mathsf{sk}_i, \mathsf{pk}_i)$. It also generates the personal secret/public keys $(\mathbf{ssk}[i], \mathbf{spk}[i])$ and the signatures sig_i for all honest members except for member j whose signature sig_j is obtained by a call to the $\mathsf{SSign}(\mathsf{sk}, \cdot)$ oracle $\mathcal{A}_2$ has access to. The key $\mathbf{spk}[j]$ is set to the public key $\mathcal{A}_2$ obtains from its game and thus $\mathbf{ssk}[j] = \perp$ because it corresponds to the secret key available to $\mathcal{A}_2$'s SSign oracle which is unknown to $\mathcal{A}_2$.

Finally, when $\mathcal{B}$ outputs a successful forgery, adversary $\mathcal{A}_2$ aborts if the framed group member is different from the one it has guessed. Otherwise, $\mathcal{A}_2$ returns $(\mathsf{pk}^*, \mathsf{sig}^*)$ as its answer in its unforgeability game. By the unforgeability of the CERT scheme, we have that $\mathsf{Adv}^{Unforg}_{\mathsf{CERT},\mathcal{A}_2}(\lambda) \leq \nu(\lambda)$.

We conclude that $\mathsf{Adv}^{Non-Frame}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \nu(\lambda)$ and hence the group blind signature scheme is non-frameable.

Lemma 5. *The construction is blind providing that the GS proof system is hiding and re-randomizable, the* $\mathsf{Pedersen}$ *commitment (used in the first round of the signing protocol) is hiding and that the SXDH assumption holds.*

Proof. We prove that

$$\mathsf{Adv}^{Blind}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{Hiding}_{\mathsf{GS},\mathcal{A}_1}(\lambda) + \mathsf{Adv}^{Hiding}_{\mathsf{Pedersen},\mathcal{A}_2}(\lambda) + \mathsf{Adv}^{SXDH}_{\mathcal{A}_3}(\lambda).$$

By the perfect witness-indistinguishability/zero-knowledge of GS proofs in the hiding setting, we have that $\mathsf{Adv}^{Hiding}_{\mathsf{GS},\mathcal{A}_1}(\lambda) = 0$. Also, the commitment used in the first round of the signing protocol, the $\mathsf{Pedersen}$ commitment scheme, is perfectly hiding, and therefore we also have $\mathsf{Adv}^{Hiding}_{\mathsf{Pedersen},\mathcal{A}_2}(\lambda) = 0$. Furthermore, we have by the security of GS proofs (i.e. the security of the SXDH assumption [33]) that an adversary has a negligible advantage in telling apart a binding CRS from a hiding one and therefore this only negligibly changes $\mathcal{B}$'s success probability. Thus, the scheme is blind.

In this game, the adversary plays the role of a signer in the group. In the CPA-version of the scheme, adversary $\mathcal{B}$ is not granted access to the Open oracle (for the same reason we mentioned in the proof for anonymity).

Note that the commitments, Co_b and Co_{1-b}, $\mathcal{B}$ sees in the first round are $\mathsf{Pedersen}$ commitments which are perfectly hiding and have identical distributions. Moreover, by making proof $\vec{\Psi}$ zero-knowledge, we can simulate all the proofs that the user sends to the signer and hence we can build an adversary against the hiding property of $\mathsf{Pedersen}$ commitment (i.e. adversary $\mathcal{A}_2$ that wins the security game in Figure 6) where

$$\mathsf{Adv}^{Hiding}_{\mathsf{Pedersen},\mathcal{A}_2}(\lambda) := |\Pr[\mathsf{Exp}^{Hiding}_{\mathsf{Pedersen},\mathcal{A}_2}(\lambda) = 1] - \frac{1}{2}|.$$

Also, by switching crs_1 and crs_2 to hiding strings, the proofs $\mathcal{B}$ sees are distributed uniformly and hence again

```
Experiment : $\mathsf{Exp}^{Hiding}_{\mathsf{Pedersen},\mathcal{A}_2}(\lambda)$
  – $\mathsf{sk} \leftarrow \mathbb{Z}_p$; $\mathsf{pk} \leftarrow G_1^{\mathsf{sk}}$.
  – $(m_0, m_1, \mathsf{state_{find}}) \leftarrow \mathcal{A}_{2_{\mathsf{find}}}(\mathsf{pk})$, where $m_0, m_1 \in \mathbb{Z}_p$.
  – $b \leftarrow \{0, 1\}$.
  – $r_0, r_1 \leftarrow \mathbb{Z}_p$.
  – $\mathsf{Co}_0 := G^{m_b} \cdot \mathsf{pk}^{r_0}$.
  – $\mathsf{Co}_1 := G^{m_{1-b}} \cdot \mathsf{pk}^{r_1}$.
  – $b^* \leftarrow \mathcal{A}_{2_{\mathsf{guess}}}(\mathsf{Co}_0, \mathsf{Co}_1, \mathsf{state_{find}})$.
  – If $b = b^*$ Then Return 1 Else Return 0.
```

Fig. 6. Security experiment for $\mathsf{Pedersen}$ commitment scheme's hiding property

if $\mathcal{B}$ can break the blindness of the group blind signature scheme, we can construct an adversary $\mathcal{A}_1$ that uses $\mathcal{B}$ (in a similar game to that used in Figure 5(b)) to break the witness-indistinguishability/zero-knowledge properties of GS proofs by distinguishing between different witnesses $\mathcal{A}_1$ gets from its Prove oracle.

Note that even if the malicious signer dishonestly chooses the randomness it uses in the GS commitments, the new proofs after the re-randomization process by the user are independent of those sent by the signer and hence they cannot be linked. Moreover, the re-randomized proofs are indistinguishable from fresh proofs generated for the same statements. Thus, the scheme is blind.

7 Achieving Full Anonymity

Since GS proofs are not simulation-sound and hence when instantiated in the hiding setting we can no longer apply the $\mathsf{GSExtract}$ algorithm to extract the witness used in the proof. This means that whenever there is a need to simulate the proofs, the Opener can no longer answer Open queries. To achieve full anonymity where the Opener can always respond to Open queries, we need an extractable simulation-sound proof system [46, 47] which allows for extraction while simulating.

Alternatively, we can use an IND-CCA secure encryption scheme to encrypt the signature σ and add an extra GS proof to prove that the encrypted signature is the same as that used as a witness in the other proofs. In this case, the Opener gets the decryption key for the encryption scheme which allows him to decrypt any ciphertext and hence recover the underlying signature from its encryption if he cannot extract it from GS proofs. It appears that the encryption scheme we require has to be re-randomizable but yet the IND-CCA security contradicts re-randomizability of ciphertexts which in some way considered a form of malleability. However, there exist a number of encryption schemes with properties that seem to suffice for this purpose, e.g. [30, 43].

8 Conclusion

We have presented a formal security model for dynamic group blind signatures which allows for obtaining rigorous proofs of security. In doing so, we have identified and remedied a number of issues which were not considered by previous constructions.

We have also presented new schemes whose security does not rely on random oracles. All our schemes have a concurrent joining protocol, a round-optimal signing protocol and yield signatures of a constant-size. We have also proved the security of our constructions.

Acknowledgments

This work was supported by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO and EPSRC via grant EP/H043454/1. The author would like to thank Nigel Smart and Bogdan Warinschi for many helpful discussions, and David Bernhard and Georg Fuchsbauer for useful feedback on an earlier version of the paper.

References

1. M. Abdalla, C. Namprempre and G. Neven. On the (im)possibility of blind message authentication codes. In *CT-RSA 2006*, Springer LNCS 3860, 262–279, 2006.
2. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *Crypto 2010*, Springer LNCS 6223, 209–236, 2010.
3. M. Abe, K. Haralambiev and M. Ohkubo. Signing on Elements in Bilinear Groups for Modular Protocol Design. *Cryptology ePrint Archive*. Report 2010/133, available at `http://eprint.iacr.org/2010/133`.
4. G. Ateniese, J. Camenisch, S. Hohenberger and B. de Medeiros. Practical group signatures without random oracles. *Cryptology ePrint Archive*. Report 2005/385, available at `http://eprint.iacr.org/2005/385`.
5. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya and H. Shacham. Randomizable Proofs and Delegatable Anonymous Credentials. In *CRYPTO 2009*, Springer LNCS 5677, 108–125, 2009.
6. M. Belenkiy, M. Chase, M. Kohlweiss and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC 2008*, Springer LNCS 4948, 356–374, 2008.
7. M. Bellare, D. Micciancio and B. Warinschi. Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In *EUROCRYPT 2003*, Springer LNCS 2656, 614–629, 2003.
8. M. Bellare and P. Rogaway. Random oracles are practical: A Paradigm for Designing Efficient Protocols. In *ACM-CCS 1993*, ACM, pp. 62–73.
9. M. Bellare, H, Shi and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, Springer LNCS 3376, 136–153, 2005.
10. S. Bengio, Y. Desmedt and C. Goutier. Special Uses and Abuses of the Fiat–Shamir Passport Protocol. In *Crypto 1987*, Springer LNCS 293, 21–39, 1988.
11. O. Blazy, G. Fuchsbauer, M. Izabach‘ene, A. Jambert, H. Sibert and D. Vergnaud. Batch Groth-Sahai. *Cryptology ePrint Archive, Report 2010/040*. `http://eprint.iacr.org/2010/040`.
12. D. Boneh and X. Boyen. Short signatures without random oracles. In *Eurocrypt 2004*, Springer LNCS 3027, 56–73, 2004.
13. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures. In *CRYPTO 2004*, Springer LNCS 3152, 227–242, 2004.
14. X. Boyen and B.Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC 2007*, Springer LNCS 4450, 1–15, 2007.
15. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, Springer LNCS 3152, 56–72, 2004.
16. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO 1997*, Springer LNCS 1294, 410–424, 1997.
17. R. Canetti, H. Krawczyk, and J. Nielsen. Relaxing Chosen-Ciphertext Security. In *CRYPTO 2003*, Springer LNCS 2729, 565–582, 2003.
18. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO 1982*, Plenum Press, 199–203, 1983.
19. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT 1991*, Springer LNCS 547, 257–265, 1991.
20. L. Chen, P. Morrissey and N.P. Smart. DAA: Fixing the pairing based protocols. IACR e-print 2009/198. `http://eprint.iacr.org/2009/198`.
21. A. Fiat and A. Shamir. How to prove yourself: practical solutions to identification and signature problems. In *CRYPTO 1986*, Springer LNCS 263, 186–194, 1987.
22. M. Fischlin. Round-optimal composable blind signatures in the common reference string model. In *Advances in Cryptology – CRYPTO 2006*, Springer LNCS 4117, 60–77, 2006.
23. G. Fuchsbauer. Automorphic Signatures in Bilinear Groups and an Application to Round-Optimal Blind Signatures. In *Cryptology ePrint Archive, Report 2009/320*, `http://eprint.iacr.org/2009/320.pdf`.
24. G. Fuchsbauer. Commuting Signatures and Verifiable Encryption and an Application to Non-Interactively Delegatable Credentials. In *Cryptology ePrint Archive, Report 2010/233*, `http://eprint.iacr.org/2010/233.pdf`.
25. S. Galbraith, K. Paterson and N.P. Smart. Pairings for cryptographers. In *Discrete Applied Mathematics*, **156**, 3113–3121, 2008

26. E. Ghadafi, N.P. Smart and B. Warinschi. Practical zero-knowledge proofs for circuit evaluation. In *Coding and Cryptography: IMACC 2009*, Springer LNCS 5921, 469–494, 2009.
27. E. Ghadafi, N.P. Smart and B. Warinschi. Groth-Sahai proofs revisited. In *PKC 2010*, Springer LNCS 6056, 177–192, 2010.
28. S. Goldwasser and S. Micali. Probabilistic encryption. In *Journal of Computer and System Sciences*, 28:2, 270–299, 1984.
29. M. Green and S. Hohenberger. Universally Composable Adaptive Oblivious Transfer. In *Asiacrypt 2008*, Springer LNCS 5350, 179–197, 2008.
30. J. Groth. Rerandomizable and Replayable Adaptive Chosen Ciphertext Attack Secure Cryptosystems. In *Theory of cryptography– TCC 2004*, Springer-Verlag LNCS 2951, 152–170, 2004.
31. J. Groth. Fully anonymous group signatures without random oracles. In *Asiacrypt 2007*, Springer LNCS 4833, 164–180, 2007.
32. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, Springer LNCS 4965, 415–432, 2008.
33. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups (full version). `http://www.brics.dk/~jg/WImoduleFull.pdf`
34. A. Juels, M. Luby and R. Ostrovsky. Security of blind digital signatures. In *Advances in Cryptology – CRYPTO '97*, Springer LNCS 1294, 150–164, 1997.
35. J. Herranz and F. Laguillaumie. Blind Ring Signatures Secure Under the Chosen-Target-CDH Assumption. In *Information Security – ISC 2006*, Springer LNCS 4176, 117–130, 2006.
36. A. Lysyanskaya, R. Rivest, A. Sahai and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography – SAC 1999*, Springer LNCS 1758, 184–199, 1999.
37. A. Lysyanskaya and R. Zulfikar. Group blind digital signatures: A scalable solution to electronic cash. In *Financial Cryptography 1998*, Springer LNCS 1465, 184–197, 1998.
38. U. Maurer. Abstract models of computation in cryptography. In *Cryptography and Coding 2005*, Springer LNCS 3796, 1–12, 2005.
39. M. Naor. On cryptographic assumptions and challenges. In *Advances in Cryptology – Crypto 2003*, Springer LNCS 2729, 96–109, 2003.
40. M. Naor and M. Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM symposium on Theory of computing – STOC 1990*, 427–437, 1990.
41. K. Q. Nguyen, Y. Mu, and V. Varadharajan. Divertible Zero-Knowledge Proof of Polynomial Relations and Blind Group Signature. In *ACISP 1999*, Springer LNCS 1587, 117–128, 1999.
42. T. Okamoto and K. Ohta. Divertible Zero Knowledge Interactive Proofs and Commutative Random Self-Reducibility. In *EUROCRYPT 1989*, Springer LNCS 434, 134–149, 1990.
43. M. Prabhakaran and M. Rosulek. Rerandomizable RCCA encryption. In *Advances in Cryptology – CRYPTO 2007*, Springer-Verlag LNCS 4622, 517–534, 2007.
44. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, **13(3)**, 361–396, 2000.
45. C. Rackoff and D. Simon. Non-Interactive Zero-Knowledge Proof of Knowledge and Chosen Ciphertext Attack. In *Crypto 1991*, Springer-Verlag LNCS 576, 433–444, 1991.
46. A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In FOCS 1999, 543–553, 1999.
47. A. Sahai. Simulation-Sound Non-Interactive Zero Knowledge. In Manuscript, 2001.
48. C.P. Schnorr. Efficient signature generation by smart cards. In *Journal of Cryptology*, **4**, 161–174, 1991.
49. J. T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. In *J. ACM*, **27**, 701–717, 1980.
50. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt 1997*, Springer LNCS 1233, 256–266, 1997.
51. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *EUROCRYPT 2005*, Springer LNCS 3494, 114–127, 2005.
52. Q. Wu, F. Zhang, W. Susilo and Yi Mu. An Efficient Static Blind Ring Signature Scheme. In *Information Security and Cryptology – ICISC 2005*, Springer LNCS 3935, 410–423, 2006.

A From Many Challenge Queries to a Single Query

Here we use a hybrid argument similar to that used in [9] to show a reduction from an adversary $\mathcal{B}$ against the anonymity property which is allowed at most $n(\lambda)$ invocations (for some polynomial $n(\cdot)$) of the CH_b oracle to an adversary $\mathcal{A}$ which is allowed a single invocation of the CH_b oracle. The hybrid argument utilizes the fact that the adversary is allowed to learn the signing keys of any group member and hence it is capable of signing on behalf of any member.

Lemma 6. *Let* GBS *be a dynamic group blind signature scheme. For any polynomial-time adversary* $\mathcal{B}$ *attacking the anonymity of* GBS *with at most* $n(\lambda)$ CH_b *invocations, there exists an adversary* $\mathcal{A}$ *attacking the anonymity of* GBS *with only a single invocation of* CH_b*.*

Proof. We define a series of games $\{\mathsf{Game}_j\}_{j=0}^{n(\lambda)}$, where in game Game_j, the first j-th challenge queries by $\mathcal{B}$ are answered using the signing key $\mathbf{gsk}[i_0]$, whereas the rest of the queries are answered using $\mathbf{gsk}[i_1]$. Let P_j be the probability that adversary $\mathcal{B}$ wins Game_j. We have that

$$P_{n(\lambda)} = \Pr[\mathsf{Exp}^{Anon-0}_{\mathsf{GBS},\mathcal{B}}(\lambda) = 1]$$
$$P_0 = \Pr[\mathsf{Exp}^{Anon-1}_{\mathsf{GBS},\mathcal{B}}(\lambda) = 1]$$

Adversary $\mathcal{A}$ is shown in Figure 7, whereas the challenge and open oracles used by $\mathcal{A}$ to answer $\mathcal{B}$'s challenge and open queries are given in Figure 8.

```
Adversary $\mathcal{A}^{\langle\cdot,\mathsf{CH}_b(\cdot,\cdot)\rangle^1}(\mathsf{gpk}, \mathsf{i}\mathfrak{k} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{SSK}(\cdot), \mathsf{ModifyReg}(\cdot,\cdot), \mathsf{Open}(\cdot,\cdot))$
– $\mathsf{CLA} := \emptyset$.
– $\mathsf{index} := 0$.
– $i \leftarrow [1, n(\lambda)]$.
– $b^* \leftarrow \mathcal{B}^{\langle\cdot,\mathsf{CH}(\cdot,\cdot)\rangle^{n(\lambda)}}(\mathsf{gpk}, \mathsf{i}\mathfrak{k} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{SSK}(\cdot), \mathsf{ModifyReg}(\cdot,\cdot), \mathsf{Open}(\cdot,\cdot))$.
– If $\mathsf{index} < i$ Then Call $\mathsf{CH}_b(\epsilon,\epsilon)$.[4] \\ Oracle Query
– Return $b^*$.
```

Fig. 7. Adversary $\mathcal{A}$.

```
$\mathsf{CH}(i_0, i_1)$:
– $\mathsf{index} := \mathsf{index} + 1$.
– If $\mathsf{index} = i$ Then Call $\mathsf{CH}_b(i_0, i_1)$.
– $\mathsf{CLA} := \mathsf{CLA} \cup \{i_0, i_1\}$.
– $(\mathsf{gsk}_0, \mathsf{ssk}_0) \leftarrow \mathsf{SSK}(i_0)$. \\ Oracle Query
– $(\mathsf{gsk}_1, \mathsf{ssk}_1) \leftarrow \mathsf{SSK}(i_1)$. \\ Oracle Query
– If $\mathsf{gsk}_0 = \epsilon$ or $\mathsf{gsk}_1 = \epsilon$ Then Return $\perp$.
– If $\mathsf{index} < i$ Then
    • Sign using key $\mathsf{gsk}_0$.
– Else
    • Sign using key $\mathsf{gsk}_1$.
```

```
$\mathsf{Open}(m, \Sigma)$:
– $(id, \tau) \leftarrow \mathsf{Open}(m, \Sigma)$. \\ Oracle Query
– If $id = \perp$ Then Return $(\perp, \perp)$.
– If $id \in \mathsf{CLA}$ Then Return $(\perp, \perp)$.
– Return $(id, \tau)$.
```

Fig. 8. The challenge oracle (left) and the Open oracle (right) used by adversary $\mathcal{A}$.

[4] This statement is to ensure that $\mathcal{A}$ will always make one call to its challenge oracle.

In answering $\mathcal{B}$'s challenge queries, $\mathcal{A}$ randomly chooses $i \leftarrow [1, n(\lambda)]$, all challenge queries $j < i$ are answered using $\mathbf{gsk}_{i_0}$. The i-th challenge query is answered using $\mathcal{A}$'s challenge oracle and the rest challenge queries are answered using $\mathbf{gsk}_{i_1}$. The rest of $\mathcal{B}$'s queries are answered using the oracles available to $\mathcal{A}$.

Adversary $\mathcal{A}$ keeps an independent list CLA containing all the signers it uses in simulating the challenge oracle for $\mathcal{B}$. Every time $\mathcal{B}$ asks an Open query, $\mathcal{A}$ looks up the identity returned by its own Open oracle in this list and returns $(\perp, \perp)$ if either the identity exists in the list or the answer of the Open oracle was $(\perp, \perp)$. Otherwise, $\mathcal{A}$ returns whatever was returned by its own Open query.

.

Since i is chosen uniformly at random from the set $[1, n(\lambda)]$, we have for every $j \in [1, n(\lambda)]$ that

$$\Pr[\mathsf{Exp}^{Anon-0}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1 | i = j] = P_j$$
$$\Pr[\mathsf{Exp}^{Anon-1}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1 | i = j] = P_{j-1}$$

Thus, we have

$$\begin{aligned}\Pr[\mathsf{Exp}^{Anon-0}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1] &= \sum_{j=1}^{n(\lambda)} \Pr[\mathsf{Exp}^{Anon-0}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1 | i = j] \\ &= \sum_{j=1}^{n(\lambda)} P_j \cdot \frac{1}{n(\lambda)}\end{aligned}$$

Similarly, we have

$$\begin{aligned}\Pr[\mathsf{Exp}^{Anon-1}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1] &= \sum_{j=1}^{n(\lambda)} \Pr[\mathsf{Exp}^{Anon-1}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1 | i = j] \\ &= \sum_{j=1}^{n(\lambda)} P_{j-1} \cdot \frac{1}{n(\lambda)}\end{aligned}$$

By the above two equations, we have

$$\begin{aligned}\mathsf{Adv}^{Anon}_{\mathsf{GBS},\mathcal{A}}(\lambda) &= \left| \Pr[\mathsf{Exp}^{Anon-0}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{Exp}^{Anon-1}_{\mathsf{GBS},\mathcal{A}}(\lambda) = 1] \right| \\ &= \frac{1}{n(\lambda)} \cdot \left| P_{n(\lambda)} - P_0 \right| \\ &= \frac{1}{n(\lambda)} \cdot \mathsf{Adv}^{Anon}_{\mathsf{GBS},\mathcal{B}}(\lambda)\end{aligned}$$

B SXDH Instantiation of the GS Proof System

Here we show how GS proofs are instantiated under the SXDH assumption. Let $\mathcal{P} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, G_1, G_2)$ be a bilinear group in which SXDH assumption holds. We set $\mathbb{A}_1 := \mathbb{G}_1$, $\mathbb{A}_2 := \mathbb{G}_2$, $\mathbb{A}_T := \mathbb{G}_T$, $\mathbb{B}_1 := \mathbb{G}_1^2$, $\mathbb{B}_2 := \mathbb{G}_2^2$ and $\mathbb{B}_T := \mathbb{G}_T^4$, all with operations performed componentwise. We let

$$F : \begin{cases} \quad \mathbb{B}_1 \times \mathbb{B}_2 & \longrightarrow \mathbb{B}_T \\ (X_1, Y_1), (X_2, Y_2) & \longmapsto (\ \hat{e}(X_1, X_2),\ \hat{e}(X_1, Y_2),\ \hat{e}(Y_1, X_2),\ \hat{e}(Y_1, Y_2)\ ) \end{cases}$$

Since the underlying pairing $\hat{e}$ is bilinear, it follows that the map F is also bilinear. To generate the CRS, the trusted party chooses $a_i, t_i \leftarrow \mathbb{Z}_p$ for $i = 1, 2$ and computes $H_i := G_i^{a_i}$, $U_i := G_i^{t_i}$, $V_i := H_i^{t_i}$. Set $\mathcal{U}_{i,1} := (G_i, H_i) \in \mathbb{B}_i$.

– $\mathsf{GSSetup}$: Generates a binding key by setting $\mathcal{U}_{i,2} := \mathcal{U}_{i,1}^{t_i} := (U_i, V_i)$. The extraction key is $\mathsf{xk} := (a_1, a_2)$.
– $\mathsf{GSSimSetup}$: Generates a hiding key by setting $\mathcal{U}_{i,2} := \mathcal{U}_{i,1}^{t_i}/(1, G_i) := (U_i, V_i/G_i)$.

The CRS is then the set $\{\vec{\mathcal{U}_1}, \vec{\mathcal{U}_2}\} \in \mathbb{B}_1^2 \times \mathbb{B}_2^2$ where $\vec{\mathcal{U}_i} := \{\mathcal{U}_{i,1}, \mathcal{U}_{i,2}\}$. Under the SXDH assumption, one cannot tell a binding CRS from a hiding CRS.

To aid what follows, we set $\mathcal{W}_i := \mathcal{U}_{i,2} \cdot (1, G_i) := (W_{i,1}, W_{i,2}) \in \mathbb{B}_i$.

GS Commitments.
We define the maps

$$\iota_i : \begin{cases} \mathbb{G}_i \longrightarrow \mathbb{B}_i \\ X \longmapsto (1, X) \end{cases} \qquad \iota'_i : \begin{cases} \mathbb{Z}_p \longrightarrow \mathbb{B}_i \\ x \longmapsto \mathcal{W}_i^x \end{cases}$$

– **Committing to Group Elements:** To commit to $X \in \mathbb{G}_i$, choose $\vec{\tau_X} \leftarrow \mathbb{Z}_p \times \mathbb{Z}_p$ and compute $\mathcal{C}_X \leftarrow \mathsf{GScomm_i}(X, \vec{\tau_X}) := \iota_i(X) \cdot \vec{\mathcal{U}_i}^{\vec{\tau_x}} \in \mathbb{B}_i$. $\mathsf{GSExtract}$ then extracts the witness from a commitment $C = (C_1, C_2) \in \mathbb{B}_i$ by computing $\mathcal{C}_2 \cdot C_1^{-a_i}$.
– **Committing to Exponents:** To commit to $x \in \mathbb{Z}_p$, randomly choose $\tau_x \leftarrow \mathbb{Z}_p$ and compute $\mathcal{C}_x \leftarrow \mathsf{GScomm_i}(x, \tau_x) := \iota'_i(x) \cdot \mathcal{U}_{i,1}^{\tau_x} \in \mathbb{B}_i$. $\mathsf{GSExtract}$ then extracts the witness from a commitment $C = (G_i^{c_1}, G_i^{c_2})$ by computing $c_2 - a_i \cdot c_1$. Note that the algorithm needs to solve discrete logarithm to be able to extract a witness from a commitment to an exponent.

The proof is constructed by first committing to each element in the witness list and then constructing a proof $\Psi := (\theta, \pi) \in \mathbb{B}_1 \times \mathbb{B}_2$ for the satisfiability of each equation. Proofs for linear equations are simpler and consist of only one of the two elements and not both. The size of the proof includes the proof itself and the associated commitments.

For more details about the proof system we refer the reader to [33].

C DH-LRSW in the Generic Group Model

Here we show that the DH-LRSW assumption holds in the generic group model [50, 38].

Theorem 3. *Let $\mathcal{A}$ denote an adversary in the generic group model against the DH-LRSW assumption. Assume $\mathcal{A}$ in this game makes q_G group operation queries, q_P pairing queries, and q_O queries to the DH-LRSW oracle $\mathsf{O}_{DH\text{-}LRSW}$. The probability ϵ of adversary $\mathcal{A}$ winning the DH-LRSW game is bounded by $\epsilon \leq \frac{(q_G+q_P+4q_O+4)^2 \cdot 4}{p}$, where p is the (prime) order of the generic groups.*

Proof. A challenger $\mathcal{B}$ interacts with adversary $\mathcal{A}$ in the game in which $\mathcal{A}$ is given access to a set of oracles. Adversary $\mathcal{A}$ interacts with those oracles via group handles. We also define three random encoding functions $\xi_i : \mathbb{G}_i \longrightarrow \{0,1\}^*$ for $i = 1, 2, T$ where ξ_i maps elements from group $\mathbb{G}_i$ into random strings. The challenger keeps three lists $\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_T$ which contain pairs of the form (σ, P) where σ is a "random" encoding of the group element (i.e. σ is an output of the map ξ_i) and P is some polynomial in $\mathbb{F}_p[X, Y, A_1, \ldots, A_{q_O}]$.

To each list we associate an Update operation, that takes as input the specific list $\mathcal{G}_i$ and a polynomial P. The function $\mathsf{Update}(\mathcal{G}_i, P)$ searches the list for a pair whose second component is equal to P, if it finds one, it returns the first component as a result. Otherwise, a new random encoding σ, distinct from all other elements used so far is chosen, and the pair (σ, P) is added to the list $\mathcal{G}_i$. The value σ is then returned. Note that at no point $\mathcal{A}$ gets access to the second element in the pairs.

The challenger starts the game by calling: $\mathsf{Update}(\mathcal{G}_1, 1)$, $\mathsf{Update}(\mathcal{G}_2, 1)$, $\mathsf{Update}(\mathcal{G}_2, X)$ and $\mathsf{Update}(\mathcal{G}_2, Y)$. The oracles used in the game are defined as follows:

– **Group Operations:** Oracles O_1, O_2 and O_T allow $\mathcal{A}$ access to the group operations, via subtraction/addition operations. On a call to $\mathsf{O}_i(\sigma_1, \sigma_2)$ $\mathcal{B}$ searches list $\mathcal{G}_i$ for pairs of the form (σ_1, P_1) and (σ_2, P_2). If both exist, $\mathcal{B}$ returns the output of $\mathsf{Update}(\mathcal{G}_i, P_1 \pm P_2)$. Otherwise, it returns $\perp$.
Note that exponentiation operations can be performed by calls to the group operation oracles.

– **Pairing Operation:** Oracle O_P allows $\mathcal{A}$ to perform pairing operations. On a call to $\mathsf{O}_P(\sigma_1, \sigma_2)$, $\mathcal{B}$ searches the list $\mathcal{G}_1$ for the pair (σ_1, P_1), and the list $\mathcal{G}_2$ for the pair (σ_2, P_2). If both exist, $\mathcal{B}$ returns the output of $\mathsf{Update}(\mathcal{G}_T, P_1 \cdot P_2)$. Otherwise, it returns $\perp$.
– **DH-LRSW Oracle:** The adversary may make up to q_O queries of the form $\mathsf{O}_{DH\text{-}LRSW}(\sigma_1, \sigma_2)$. The challenger searches list $\mathcal{G}_1$ for a pair (σ_1, P_1) and list $\mathcal{G}_2$ for a pair (σ_2, P_2). If they do not exist or $P_1 \neq P_2$, $\mathcal{B}$ returns $\perp$. Otherwise, it executes the following operations, where A_i, X and Y are indeterminants:

$$\begin{aligned}\sigma_{A_i} &\leftarrow \mathsf{Update}(\mathcal{G}_1, A_i),\\ \sigma_{B_i} &\leftarrow \mathsf{Update}(\mathcal{G}_1, A_i \cdot Y),\\ \sigma_{C_i} &\leftarrow \mathsf{Update}(\mathcal{G}_1, A_i \cdot P_1 \cdot Y),\\ \sigma_{D_i} &\leftarrow \mathsf{Update}(\mathcal{G}_1, X \cdot A_i \cdot (1 + P_1 \cdot Y)).\end{aligned}$$

Returning the tuple $(\sigma_{A_i}, \sigma_{B_i}, \sigma_{C_i}, \sigma_{D_i})$ to $\mathcal{A}$.

At the end of the game, the total number of non-constant polynomials contained in the three lists $\mathcal{G}_1, \mathcal{G}_2$ and $\mathcal{G}_T$ after $\mathcal{A}$ has made its q_O queries to the DH-LRSW oracle is bounded from above by $t = q_G + q_P + 4q_O + 4$.

Using the above oracles, we can simulate the entire run of the adversary where the adversary may make no decision which depends on the particular encoding of group elements used.

<u>The Adversary Output</u>: Eventually, $\mathcal{A}$ will output a tuple $(\sigma_A, \sigma_B, \sigma_C, \sigma_D, \sigma_{M_1}, \sigma_{M_2})$, where σ_A, σ_B, σ_C, σ_D and σ_{M_1} are on list $\mathcal{G}_1$ while σ_{M_2} is on list $\mathcal{G}_2$. We let $P_A, P_B, P_C, P_D, P_{M_1}, P_{M_2}$ denote the polynomials associated with these encodings.

For the output of $\mathcal{A}$ to be correct, it must correspond to a solution to the DH-LRSW problem, so these output polynomials can be assumed to satisfy, for some assignment $(x, y, a_1, \ldots, a_{q_O}) \in \mathbb{F}_p^{2+q_O}$ to the variables $(X, Y, A_1, \ldots, A_{q_O})$, the equations

$$P_B = P_A \cdot Y \tag{5}$$
$$P_C = P_B \cdot P_{M_2} \tag{6}$$
$$P_D = X \cdot (P_A + P_C) \tag{7}$$
$$P_{M_1} = P_{M_2} \tag{8}$$

From this we wish to derive a contradiction, i.e. conclude that the adversary cannot solve the DH-LRSW assumption in the GGM. To do so we need to first ensure that these polynomial identities cannot hold identically, i.e. irrespective of any particular assignment $(x, y, a_1, \ldots, a_{q_O}) \in \mathbb{F}_p^{2+q_O}$ to the variables $(X, Y, A_1, \ldots, A_{q_O})$.

Let $(M_{1,i}, M_{2,i})$ denote the ith query to the DH-LRSW oracle, where we discount queries which return $\perp$. It is easy to see that the only polynomials on the list $\mathcal{G}_2$ are linear combinations of the terms $1, X$ and Y we see that we must have $P_{M_{2,i}} = r_i + s_i \cdot X + t_i \cdot Y$. Since we have $P_{M_{1,i}} = P_{M_{2,i}}$, this implies that the above polynomials must also appear on the list $\mathcal{G}_1$. However, it is also clear that there is no operation in $\mathbb{G}_1$ which we create a polynomial with a monomial term of X, nor one of Y. Thus, we can conclude that all queries to the DH-LRSW oracle correspond to elements whose polynomials are a constant term of the form $P_{M_{1,i}} = P_{M_{2,i}} = r_i$. By a similar argument, we can also deduce that the output of the adversary corresponds to polynomials with $P_{M_1} = P_{M_2} = r$. Note, this is precisely where we use the property that the oracle will return $\perp$ unless the input query lies in $\hat{G}$.

Since the queries are for constant polynomials only, we see that the only polynomials which can appear on the list $\mathcal{G}_1$ are of the form

$$v_0 + \sum_{i=1}^{q} v_{1,i} \cdot A_i + \sum_{i=1}^{q} v_{2,i} \cdot A_i \cdot Y + \sum_{i=1}^{q} v_{3,i} \cdot X \cdot A_i \cdot (1 + r_i \cdot Y) \tag{9}$$

where $v_0, v_{1,i}, v_{2,i}, v_{3,i} \in \mathbb{F}_p$.

By (5) and (6) we have that $P_B = P_A \cdot Y$ and $P_C = P_B \cdot P_{M_2}$, where $P_A, P_B, P_C \in \mathcal{G}_1$. This implies that we must have $P_A = \sum_{i=1}^{q} a_{1,i} \cdot A_i$, $P_B = \sum_{i=1}^{q} a_{1,i} \cdot A_i \cdot Y$ and $P_C = r \cdot \sum_{i=1}^{q} a_{1,i} \cdot A_i \cdot Y$, where $P_{M_1} = r$.

Equation (7) then implies that we have $X \cdot \left(\sum_{i=1}^{q} a_{1,i} \cdot A_i + r \cdot \sum_{i=1}^{q} a_{1,i} \cdot A_i \cdot Y \right) \in \mathcal{G}_1$, i.e. $\sum_{i=1}^{q} a_{1,i} \cdot X \cdot A_i(1 + r \cdot Y) \in \mathcal{G}_1$. But since all elements in $\mathcal{G}_1$ are of the form (9) this implies that we must have $r = r_{i'}$ and $a_{1,i} = 0$ if $i \neq i'$ for some value $i' \in [1, q_O]$. This would contradict the DH-LRSW assumption since the output pair (M_1, M_2) would then be identical to one of the queries to the oracle.

Thus, the adversary must win, or tell it is in a simulation, via a specific (random) assignment to the variables. We now turn to bounding the probability that the adversary wins (or detects the simulation) in this case.

Analysis of the challenger simulation: Now $\mathcal{B}$ chooses random values $x, y, a_i \in \mathbb{F}_p$ and evaluates the polynomials. We need to show that the challenger's simulation is sound. If $\mathcal{A}$ learned it was interacting in a simulated game, there would be two different polynomials $P_{i,j}(x, y, a_i) = P_{i,j'}(x, y, a_i)$ in list $\mathcal{G}_i$ where $P_{i,j} \neq P_{i,j'}$. The simulation will fail if any of the following is correct:

$$P_{1,j}(x, y, a_i) = P_{1,j'}(x, y, a_i) \tag{10}$$
$$P_{2,j}(x, y, a_i) = P_{2,j'}(x, y, a_i) \tag{11}$$
$$P_{T,j}(x, y, a_i) = P_{T,j'}(x, y, a_i) \tag{12}$$

Since the maximum degree of any polynomial in list $\mathcal{G}_1 \leq 3$, by applying [50][Lemma 1], we have that the probability of Equation (10) holding is $\leq \frac{3}{p}$. Similarly, since the maximum degree of any polynomial in list $\mathcal{G}_2 \leq 1$, we have that the probability of Equation (11) holding is $\leq \frac{1}{p}$. Finally, the probability of Equation (12) holding is $\leq \frac{4}{p}$.

Now summing over all possible values of j in each case, we bound this probability by

$$\epsilon \leq \binom{|\mathcal{G}_1|}{2} \frac{3}{p} + \binom{|\mathcal{G}_2|}{2} \frac{1}{p} + \binom{|\mathcal{G}_T|}{2} \frac{4}{p},$$

where $|\mathcal{G}_i|$ denotes the size of list $\mathcal{G}_i$.

In conclusion, the probability that an adversary breaks the DH-LRSW assumption in the GGM is bounded by $\epsilon \leq \frac{(q_G + q_P + 4q_O + 4)^2 \cdot 4}{p}$.

D Proofs that a committed value $A \in \mathbb{G}$ is non-trivial i.e. $A \neq 1$.

D.1 Protocol I

Assuming $A \in \mathbb{G}_1$, the prover randomly chooses a secret value $Z \leftarrow \mathbb{G}_2^{\times}$ and computes the following proof

$$\Psi \leftarrow \mathsf{GSPOK}\left(\{A, Z\}, \{\hat{e}(\underline{A}, \underline{Z}) = t_T\}\right).$$

The verifier can then verify the proof Ψ and additionally check that $t_T \neq 1$ which grantees that $A \neq 1$. The proof is of size $2 \cdot \mathbb{G}_1^2 \times 2 \cdot \mathbb{G}_2^2$ and we require one extra GS commitment in $\mathbb{G}_2^2$. Thus, the total size of the proof is $2 \cdot \mathbb{G}_1^2 \times 3 \cdot \mathbb{G}_2^2$.

Theorem 4. *The proof is re-randomizable and is witness-indistinguishable and can be made zero-knowledge.*

Proof. Correctness and soundness follow from that of GS proofs and the fact that if $A = 1$ then regardless of the value of Z, $t_T = 1$ which stops the prover from cheating.

To prove witness-indistinguishability, observe that when one switches to the hiding setting, GS commitments are perfectly hiding and thus do not reveal which witness they hide. Moreover, since Z is chosen uniformly, t_T perfectly hides A. Thus, the proof is witness-indistinguishable.

Re-randomizability. The bilinearity of the bilinear map and the structure of GS proofs allow a party who does not know the witness of the proof to re-randomize the proof by re-randomizing the value Z (without knowing it) and t_T and adjusting the proof accordingly. For a proof of this claim see [24] (Lemma 4).

Although we do not require the proof to be zero-knowledge, observe that if we factor t_T to $t_T = \hat{e}(A^r, Z^{r^{-1}})$ for some random $r \leftarrow \mathbb{Z}_p$ then A^r hides A and $Z^{r^{-1}}$ hides Z and thus we can make those randomized values public which makes the equation simultable [33] and hence the proof can be made zero-knowledge.

D.2 Protocol II

Instead of using a quadratic PPE equation (as is the case with Protocol I) which yields a less efficient proof and which is more costly to be made zero-knowledge, we use a proof for a multi-scalar multiplication equation.

Again, assuming $A \in \mathbb{G}_1$, the prover randomly chooses $z \leftarrow \mathbb{Z}_p$ and computes $Z := A^z$ and produces the following GS proof

$$\Psi \leftarrow \mathsf{GSPOK}\left(\{A, z\}, \{\underline{A}^{\underline{z}} = Z\}\right).$$

The verifier verifies the proof Ψ and additionally checks that $Z \neq 1$ which guarantees that $A \neq 1$. Proofs of this type of equations can be made zero-knowledge for free [33]. Note here that although GS proofs only provide F-extractability [6] when working with exponent witnesses, it does not affect the construction because at no point need we to efficiently extract the exponent z, we just need to ensure that the proof is sound.

The proof is of size $\mathbb{G}_1^2 \times 2 \cdot \mathbb{G}_2^2$ and we require one extra GS commitment in $\mathbb{G}_2^2$. Thus, the total size of the proof is $\mathbb{G}_1^2 \times 3 \cdot \mathbb{G}_2^2$ which is more efficient than the Protocol I.

Theorem 5. *The proof is witness-indistinguishable/zero-knowledge and is re-randomizable.*

Proof. Correctness and soundness follow from that of GS proofs and the fact that if $A = 1$ then regardless of the value of z, we will always have that $Z = 1$ and hence the prover cannot cheat.

To prove witness-indistinguishability/zero-knowledge, note that the GS commitments are perfectly hiding when we switch to the hiding setting. Moreover, since z is uniformly chosen, Z perfectly hides A. To get zero-knowledge for free, we just need to move Z to the left-hand side.

Re-randomization. To prove re-randomizability in the sense that we need, we prove that the same re-randomization technique [24] (Lemma 4) used for proofs for quadratic PPE equations (such as that proved in Protocol I), also works for proofs for quadratic multi-scalar multiplication equations.

The user re-randomizes the GS commitment to z by raising it to some random integer and doing the same to the proof. This re-randomizes z without knowing it. To prove this, we provide a proof for Lemma 1 in Section 6.1.

Proof of Lemma 1.

The original GS commitments used in the original proof are

$$\mathcal{C}_A \leftarrow \mathsf{GS}\mathrm{comm}_1(A, \vec{\tau_A}) := \iota_1(A) \cdot \vec{\mathcal{U}_1}^{\vec{\tau_A}}$$
$$\mathcal{C}_z \leftarrow \mathsf{GS}\mathrm{comm}_2(z, \tau_z) := \iota_2'(z) \cdot \mathcal{U}_{2,1}^{\tau_z}$$

The proof for the original equation is give by $\Psi := (\theta, \vec{\pi}) \in \mathbb{B}_1 \times \mathbb{B}_2^2$, where

$$\theta := \iota_1(A)^{\tau_z} \cdot \vec{\mathcal{U}_1}^{\vec{t}} \qquad\qquad \vec{\pi} := \iota_2'(z)^{\vec{\tau_A}^T} \cdot \mathcal{U}_{2,1}^{\tau_z \cdot \vec{\tau_A}^T - \vec{t}^T},$$

where $\vec{t} \leftarrow \mathbb{Z}_p \times \mathbb{Z}_p$. Now to re-randomize the secret value z to $z \cdot z'$ and adapt the proofs without knowledge of z where $z' \leftarrow \mathbb{Z}_p$, we compute $\mathcal{C}_z' := \mathcal{C}_z^{z'}$. We have that

$$\begin{aligned} \mathcal{C}_z' &= \mathcal{C}_z^{z'} \\ &= \iota_2'(z \cdot z') \cdot \mathcal{U}_{2,1}^{z' \cdot \tau_z} \end{aligned}$$

Now the proof $\Psi' := \Psi^{z'}$ is a proof for the equation $\underline{A}^{\underline{z \cdot z'}} = Z^{z'}$ as we have

$$\theta' = \iota_1(A)^{z' \cdot \tau_z} \cdot \vec{\mathcal{U}}_1^{z' \cdot \vec{t}} \qquad\qquad \vec{\pi'} = \iota_2'(z \cdot z')^{\vec{\tau_A}^T} \cdot \mathcal{U}_{2,1}^{z' \cdot (\tau_z \cdot \vec{\tau_A}^T - \vec{t}^T)}$$