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. In this paper, we first provide foundations for dynamic group blind signatures where we provide formal security definitions and present a security model to capture all the security properties. In doing so, we identify and address some issues which were not considered by previous constructions and (informal) security definitions.
We then present a practical construction which has a round-optimal signing phase and yields signatures of a constant-size. Our construction allows for members of the group to join dynamically and concurrently and yet its security does not rely on any idealized assumptions. We provide different instantiations of the construction, all of which are proven secure in the standard model.
In addition, the building blocks we present are of interest in their own right and could be used either on their own or as building blocks for other cryptographic constructions.

Keywords: Group Signatures, Blind Signatures, Group Blind Signatures.

1 Introduction

Background. Traditional group signatures as introduced by [17] 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 scheme (BS) [16] is a cryptographic construct which 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 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) as introduced by [34] is a cryptographic construct that 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 have a formalized model which includes the details of all the security experiments required to capture the different security properties which 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 to be signed is public. For instance, one needs to ensure the existence of some mechanism in order to prevent the adversary from cheating in the anonymity game 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.

In addition, existing constructions e.g. [34, 37] require either many rounds of interaction in the signing phase (deploying interactive divertible proofs of knowledge [8]) and hence requiring that the signing authority remains online throughout the signing process or requiring proofs in the random oracle model [6]. 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 [34], 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. [14, 5, 11, 3, 12, 28], only a few group blind signature schemes exist in the literature, e.g. [34, 37].

The subtlety one faces when designing group blind signatures lies in the need to use the same zero-knowledge proof to prove two different statements by two different parties which could be regarded in some sense as a malleable proof. 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 presented in [34] were based on different variants of the Camenisch-Stadler group signature [14] and their security required the use of random oracles. Other schemes, e.g. [37] used divertible zero-knowledge proofs [8, 38] 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 either require many rounds of interaction in the signing protocol or the Fiat-Shamir transformation [19] to eliminate the interaction required for the proofs which results in the security proof for such schemes lying in the random oracle model, e.g. [34].

Our Contribution. We first formalize the security model for dynamic group blind signatures and then present efficient dynamic group blind signature constructions which have a round-optimal signing phase and whose security does not rely on random oracles. Our schemes are practical and has a concurrent join protocol and yield signatures of a constant-size.

We start by showing how to construct CPA-anonymous schemes and then outline how such schemes can be extended to provide full anonymity. We also provide a proof of security for our schemes.

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 and describe the techniques underlying our constructions in Section 6. In Sections 7 and 8, we present and explain our constructions as well as provide a security proof for them. In Section 9, we outline how we can achieve full anonymity as well as other instantiations. Finally, we conclude the paper in Section 10.

2 Preliminary Definitions

Notation. We say that a function $\nu(.) : \mathcal{N} \rightarrow \mathcal{R}^+$ 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 $i \in \mathcal{N}$ 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.

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) $\hat{e}$. Here we will use multiplicative notation. We will 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 [22], 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 Security 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 $\mathbb{G}_2$.*

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 [33]. 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 $\mathbb{Z}_p$ [18]. 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 [26, 18].

Definition 2 (Dual Hidden-LRSW (DH-LRSW) Assumption). *Consider the following experiment. A challenger picks $x, y \in \mathbb{Z}_p$ and then computes $X \leftarrow G_2^x$, $Y \leftarrow G_2^y$. 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 $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{DH\text{-}LRSW}}(\lambda)$ in the above game, where the advantage is defined as follows:

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

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 its security in the generic group model [46, 35] in Appendix B.

Definition 3 (AWFCDH Assumption). *This assumption was introduced in [20]. We describe this assumption in asymmetric bilinear groups. Given* (G_1, G_2, G_1^a) *for* $G_1 \leftarrow \mathbb{G}_1$, $G_2 \leftarrow \mathbb{G}_2$ *and* $a \leftarrow \mathbb{Z}_p^*$, *it is computationally infeasible to output a tuple* $(G_1^b, G_1^{ab}, G_2^b, G_2^{ab})$ *for an arbitrary* $b \in \mathbb{Z}_p^*$.

The next assumption was introduced in [20] and is a variant of the q-SDH assumption [10].

Definition 4 (q-ADHSDH Assumption). *We describe this assumption in asymmetric bilinear groups. Given* $(G_1, G_2, F, K, X_1 \leftarrow G_1^x, X_2 \leftarrow G_2^x)$ *for* $G_1, F, K \leftarrow \mathbb{G}_1$, $G_2 \leftarrow \mathbb{G}_2$, $x \leftarrow \mathbb{Z}_p^*$, *and* $q-1$ *tuples*

$$\left(A_i \leftarrow (K \cdot G_1^{r_i})^{\frac{1}{x+c_i}}, C_{1,i} \leftarrow F^{c_i}, C_{2,i} \leftarrow G_2^{c_i}, R_{1,i} \leftarrow G_1^{r_i}, R_{2,i} \leftarrow G_2^{r_i}\right)_{i \in [1,q-1]}, \text{ for } c_i, r_i \leftarrow \mathbb{Z}_p,$$

it is computationally infeasible to output a new tuple $(A^*, C_1^*, C_2^*, R_1^*, R_2^*)$.

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 λ and generates the group public key gpk, the Issuer's key $\mathfrak{ik}$ and the Opener's key $\mathfrak{ok}$.
- **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 **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 [5, 7]. 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; a table **reg** where the element i in this table contains the registration information of the group member Signer_i; a table **spk** where $\mathbf{spk}[i]$ contains the personal public key of the group member Signer_i. The lists HSL, CSL and BSL are empty at initialization while the entries of the tables **reg** and **spk** are initialized to ϵ. The set of oracles the adversary has access to in the experiments are defined as follows:

- $\mathsf{AddS}(i)$: 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.

The details of those oracles are shown in Figure 1. 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.

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

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

$\underline{\mathsf{AddS}(i)}$:

- If $i \in \mathsf{HSL} \cup \mathsf{CSL} \cup \mathsf{BSL}$ Then Return $\perp$.
- $\mathsf{HSL} \leftarrow \mathsf{HSL} \cup \{i\}$.
- $(\mathbf{ssk}[i], \mathbf{spk}[i]) \leftarrow \mathsf{SKg}(1^\lambda)$.
- $\mathsf{cert}_i \leftarrow \perp$, $\mathsf{dec}^i_{\mathsf{Issue}} = \mathsf{cont}$.
- $\mathsf{St}^i_{\mathsf{Join}} \leftarrow (\mathsf{gpk}, i, \mathbf{ssk}[i])$.
- $\mathsf{St}^i_{\mathsf{Issue}} \leftarrow (\mathfrak{i}\mathfrak{k}, i, \mathbf{spk}[i])$.
- $(\mathsf{St}^i_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathsf{dec}^i_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{St}^i_{\mathsf{Join}}, \perp)$.
- 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 $\mathbf{reg}[i] \leftarrow \mathsf{St}^i_{\mathsf{Issue}}$.
- If $\mathsf{dec}^i_{\mathsf{Join}} = \mathsf{accept}$ Then $\mathbf{gsk}[i] \leftarrow \mathsf{St}^i_{\mathsf{Join}}$.
- Return $\mathbf{spk}[i]$.

$\underline{\mathsf{SndToI}(i, M_{in})}$:

- 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
  - $\mathsf{St}^i_{\mathsf{Issue}} \leftarrow (\mathfrak{i}\mathfrak{k}, i, \mathbf{spk}[i])$.
- $(\mathsf{St}^i_{\mathsf{Issue}}, M_{out}, \mathsf{dec}^i_{\mathsf{Issue}}) \leftarrow \mathsf{Issue}(\mathsf{St}^i_{\mathsf{Issue}}, M_{in})$.
- If $\mathsf{dec}^i_{\mathsf{Issue}} = \mathsf{accept}$ Then $\mathbf{reg}[i] \leftarrow \mathsf{St}^i_{\mathsf{Issue}}$.
- Return $(M_{out}, \mathsf{dec}^i_{\mathsf{Issue}})$.

$\underline{\mathsf{SndToS}(i, M_{in})}$:

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

$\underline{\mathsf{CrptS}(i, \mathsf{pk})}$:

- If $i \in \mathsf{HSL} \cup \mathsf{CSL} \cup \mathsf{BSL}$ Then Return $\perp$.
- $\mathsf{CSL} \leftarrow \mathsf{CSL} \cup \{i\}$.
- $\mathbf{spk}[i] \leftarrow \mathsf{pk}$.
- $\mathsf{St}^i_{\mathsf{Issue}} \leftarrow (\mathfrak{i}\mathfrak{k}, i, \mathbf{spk}[i])$.
- $\mathsf{dec}^i_{\mathsf{Issue}} \leftarrow \mathsf{cont}$.
- Return 1.

$\underline{\mathsf{ReadReg}(i)}$:

- Return $\mathbf{reg}[i]$.

$\underline{\mathsf{ModifyReg}(i, val)}$:

- $\mathbf{reg}[i] \leftarrow val$.

$\underline{\mathsf{SSK}(i)}$:

- If $i \notin \mathsf{HSL} \setminus \{\mathsf{CSL} \cup \mathsf{BSL}\}$ Then Return $\perp$.
- $\mathsf{BSL} \leftarrow \mathsf{BSL} \cup \{i\}$.
- Return $(\mathbf{gsk}[i], \mathbf{ssk}[i])$.

$\underline{\mathsf{OSign}(i)}$:

- 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 $\perp$.
- Call $\mathsf{Sign}(\mathbf{gsk}[i_b])$.

$\underline{\mathsf{Open}(m, \Sigma)}$:

- Let $\mathcal{T}$ be the completed transcript of the CH_b oracle.
- If $\mathsf{IdentifyTrans}(\mathcal{T}, m, \Sigma) = 1$ Then Return $\perp$.
- Return $\mathsf{Open}(\mathsf{gpk}, \mathfrak{o}\mathfrak{k}, \mathbf{reg}, m, \Sigma)$.

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

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 employ the same anonymity definition used by [5, 7], where we require that given two signers of its choice, the adversary still cannot distinguish which of the two signers produced a signature. Note that our model provides 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 [11] (which is analogous to IND-CPA security for public-key encryption schemes [25]), 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.

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

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} \leftarrow \emptyset$; $\mathsf{CSL} \leftarrow \emptyset$; $\mathsf{BSL} \leftarrow \emptyset$.
- $(i, m) \leftarrow \mathcal{A}(\mathsf{gpk} : \mathsf{AddS}(\cdot), \mathsf{ReadReg}(\cdot))$.
- 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 $\mathsf{Exp}^{Anon-b}_{\mathsf{GBS},\mathcal{A}}(\lambda)$

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

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

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} \leftarrow \emptyset$; $\mathsf{CSL} \leftarrow \emptyset$; $\mathsf{BSL} \leftarrow \emptyset$.
- $(m, \Sigma) \leftarrow \mathcal{A}(\mathsf{gpk}, \mathfrak{ok} : \mathsf{AddS}(\cdot), \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToI}(\cdot,\cdot), \mathsf{ReadReg}(\cdot), \mathsf{SSK}(\cdot))$.
- 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 $\mathsf{Exp}^{Non-Frame}_{\mathsf{GBS},\mathcal{A}}(\lambda)$

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} \leftarrow \emptyset$; $\mathsf{CSL} \leftarrow \emptyset$; $\mathsf{BSL} \leftarrow \emptyset$.
- $(id, \{(m_1, \Sigma_1), \ldots, (m_{l+1}, \Sigma_{l+1})\}) \leftarrow \mathcal{A}^{\langle \cdot, \mathsf{OSign}(\cdot) \rangle^*}(\mathsf{gpk}, \mathfrak{ok}, \mathfrak{ik} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{ModifyReg}(\cdot,\cdot), \mathsf{SSK}(\cdot))$.
- 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 $\mathsf{Exp}^{Blind-b}_{\mathsf{GBS},\mathcal{A}}(\lambda)$

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} \leftarrow \emptyset$; $\mathsf{CSL} \leftarrow \emptyset$; $\mathsf{BSL} \leftarrow \emptyset$.
- $(m_0, m_1, \mathsf{state}_{\mathsf{find}}) \leftarrow \mathcal{A}_{\mathsf{find}}(\mathsf{gpk}, \mathfrak{ik} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{ReadReg}(\cdot), \mathsf{SSK}(\cdot), \mathsf{Open}(\cdot,\cdot))$.
- $\mathsf{state}_{\mathsf{sign}} \leftarrow \mathcal{A}_{\mathsf{sign}}^{\langle \mathsf{Obtain}(\mathsf{gpk}, m_b), \cdot \rangle^1, \langle \mathsf{Obtain}(\mathsf{gpk}, m_{1-b}), \cdot \rangle^1}(\mathsf{state}_{\mathsf{find}})$.
- Let Σ_0 and Σ_1 be the outputs of the honest Obtain algorithm after interactions on m_b and m_{1-b}, respectively. If $\Sigma_0 = \perp$ or $\Sigma_1 = \perp$ Then $\Sigma_0 \leftarrow \perp; \Sigma_1 \leftarrow \perp$.
- $b^* \leftarrow \mathcal{A}_{\mathsf{guess}}(\mathsf{state}_{\mathsf{sign}}, \Sigma_0, \Sigma_1 : \mathsf{Open}(\cdot,\cdot))$. // Open queries on either (m_b, Σ_0) or (m_{1-b}, Σ_1) would be rejected.
- Return b^*.

Fig. 2. Security experiments for dynamic group blind signatures

On the contrary, in full anonymity [7] (which is analogous to IND-CCA2 security for public-key encryption schemes [41]), the adversary can ask Open queries on any signature except the one it is being challenged on at any stage of the security 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 [36]) in which the adversary can only ask Open queries before it sees the challenge signature.

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 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. Remember that in the anonymity game the adversary has access to the group signing keys of all group members including the two it chooses for the challenge and therefore it can generate signatures on behalf of those two signers.

All previous security definitions did not consider this issue and therefore their definition for anonymity is vulnerable to an attack where the adversary reveals a 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 and hence trivially breaks anonymity.

To get around this issue, there are a number of possible solutions which we list as follows:

- The first solution requires not answering any open queries involving any signatures produced by any of the two signers i_0 and i_1 the adversary sent to the CH_b oracle. This could be done by returning a special symbol such as in IND-RCCA security [15]. However, this definition is rather restrictive.
- A second solution is to require that the two identities i_0 and i_1, the adversary chooses for its challenge, are for honest signers (i.e. $i_0 \in \mathsf{HSL} \setminus \mathsf{BSL}$ and $i_1 \in \mathsf{HSL} \setminus \mathsf{BSL}$). This is not because knowledge of the signing keys of those two signers would help the adversary identify the group member who produced the challenge signature, but rather to prevent it from using such keys to produce other signatures that it could claim that it got from the interaction with the CH_b oracle. This is also a rather restrictive assumption. Moreover, it might not be suitable for the case where the signing protocol is of weak unforgeability because signing protocols with such a property would always allow the holder of a valid signature to produce a new signature on the same message even without knowledge of the signing key.
- A third solution is to require that the adversary runs in 3 phases (find, obtain and guess respectively), where in the obtain phase it interacts with the CH_b oracle but does not get access to the Open oracle. At the end of that phase, the adversary must output the message/signature pair it used in interacting with the CH_b oracle. We can then apply the Open algorithm to this signature and reject if the returned signature is signed by a signer different from Signer_{i_b} used by the CH_b oracle. If the adversary can return a signature by the same signer as that used by the CH_b oracle then it can simply just guess the bit b.
  In the guess phase, we again grant the adversary access to the Open oracle providing that it does not ask for the challenge signature to be opened. Note that this approach is appropriate for the case where the signing protocol is of strong unforgeability. However, for the case where the signing protocol is of weak unforgeability, this definition is not sufficient because the adversary can always produce a new signature on the same message using the signature it produced from interacting with the CH_b oracle. We give the details in Appendix C for the sake of illustration.
- We could require the existence of a mechanism to match up the signing interaction transcript with the message/signature pair. Note here that such an identifying mechanism does not have to be efficient and it is sufficient even if it is only information-theoretically possible. For instance, computing it could require a special trapdoor information.

To use the latter approach, we define a predicate $\mathsf{IdentifyTrans}$ to stop the adversary from cheating in the anonymity game (using the attack we described above). We let $\mathcal{GSK}$, $\mathcal{M}$, $\mathcal{SIG}$, $\mathcal{R}_{\mathsf{CH}_b}$ and $\mathcal{R}_{\mathsf{Obtain}}$ be the spaces of all possible group signing keys, all possible messages, all possible signatures, randomness used by the CH_b oracle and randomness used by the adversary (i.e. the Obtain algorithm), respectively. We also let $\mathcal{T}$ denote a completed signing interaction transcript where

$$\mathcal{T} = (m, r_{\mathsf{Obtain}}, \mathbf{gsk}[i], r_{\mathsf{CH}_b}) \in \mathcal{M} \times \mathcal{R}_{\mathsf{Obtain}} \times \mathcal{GSK} \times \mathcal{R}_{\mathsf{CH}_b}.$$

The predicate is then defined as follows:

$$\mathsf{IdentifyTrans} : (\mathcal{M} \times \mathcal{R}_{\mathsf{Obtain}} \times \mathcal{GSK} \times \mathcal{R}_{\mathsf{CH}_b}) \times \mathcal{M} \times \mathcal{SIG} \longrightarrow \{0, 1\},$$

where we require that the following two conditions are satisfied:

- If $\mathcal{T}$ is the transcript of a completed interaction $(\Sigma, \perp) \longleftarrow \langle \mathsf{Obtain}(m; r_{\mathsf{Obtain}}), \mathsf{CH}_b(\mathbf{gsk}[i_b]; r_{\mathsf{CH}_b}) \rangle$, then $\mathsf{IdentifyTrans}(\mathcal{T}, m, \Sigma) = 1$.
- If $\mathsf{IdentifyTrans}(\mathcal{T}, m, \Sigma) = 1$, then there exists no other pair $(m', \Sigma') \neq (m, \Sigma)$ such that $\mathsf{IdentifyTrans}(\mathcal{T}, m', \Sigma') = 1$.

Note that for information-theoretically blind signing protocols, identifying the message/signature from the transcript might not be possible.

This way we can ensure that the adversary in the anonymity game cannot cheat by querying the Open oracle on the signature it got from the CH_b oracle. Our definition of anonymity also captures unlinkability of signatures and it is easy to show that if an adversary can link two different signatures by the same signer, it can win our anonymity game. Therefore, there is no need to define a separate experiment for unlinkability.

Formally, a dynamic group blind signature scheme GBS is anonymous if for all $\lambda \in \mathcal{N}$, 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 $\lambda \in \mathcal{N}$, 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 [31, 40]. 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 $\lambda \in \mathcal{N}$, 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:

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

An alternative definition is also possible for cases where an identifying mechanism to uniquely match messages/signatures to their completed signing transcript exists. Such an identifying mechanism must not be publicly computable. For instance, it can only be information-theoretically feasible by requiring a special trapdoor information.

In this alternative definition, we only require that the adversary outputs a message, the signature it has forged and the identity of the honest group member to whom the forgery is attributed (as opposed to

outputting $l+1$ signatures after l completed interactions with the group member). However, this approach is not general and requires that there exists a way to check if a particular message/signature pair corresponds to a particular transcript which in some cases might not even be information-theoretically possible (e.g. when the signing protocol is information-theoretically blind).

4.5 Blindness

A dynamic group blind signature scheme GBS is blind if the adversary is unable to tell which message it is signing. Blindness also implies that the adversary cannot link a signature to the protocol run via which it was obtained. In blind signatures [31], 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. We can consider a strong definition where we allow the adversary to use different group signing keys in producing the two 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 captured by this definition. We give this definition in Figure 2.

On the other hand, one can restrict the adversary to using the same group signing key (possibly a corrupt one) in producing both challenge signatures. Note here that this definition would require the ability to open signatures to verify whether or not the adversary signed both challenge signatures with the same key. Hence, this approach cannot be directly applied in the case of CPA-anonymity, i.e. the case where it is not possible to open signatures. Despite this restriction, the adversary can still win the game if it can successfully link its view of the interactions to the final signatures .

In the definition, 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 cases where the adversary can ask open queries at the different stages with the exception that the adversary must not be able to ask open queries on either of the two challenge signatures. Otherwise, blindness can be trivially broken. Also, we require that if either of the 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 $\lambda \in \mathcal{N}$, 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 [32] 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 Variant of the CL Signature Scheme

We introduce a new variant of the CL signature scheme [13] which we refer to as (NCL). The unforgeability of the NCL signature scheme is proved 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 $\mathsf{sk}_{\mathsf{NCL}} \leftarrow (x, y) \in \mathbb{Z}_p \times \mathbb{Z}_p$ and $\mathsf{pk}_{\mathsf{NCL}} \leftarrow (X, Y) = (G_2^x, G_2^y) \in \mathbb{G}_2^2$.
- $\mathsf{NCLSign}(\mathsf{sk}_{\mathsf{NCL}}, (M_1, M_2))$: To sign a message $(M_1, M_2) \in \mathbb{G}_1 \times \mathbb{G}_2$ where $M_1 \leftarrow G_1^m$ and $M_2 \leftarrow G_2^m$, 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 $A \leftarrow G_1^a$, $B \leftarrow A^y$, $C \leftarrow M_1^{ay}$, $D \leftarrow A^x \cdot M_1^{axy}$. Output $\sigma = (A, B, C, D) \in \mathbb{G}_1^4$.
- $\mathsf{NCLVerify}(\mathsf{pk}_{\mathsf{NCL}}, (M_1, M_2), \sigma)$: Output true if and only if $A \neq 0 \ \wedge \ \hat{e}(A, Y) = \hat{e}(B, G_2) \ \wedge \ \hat{e}(B, M_2) = \hat{e}(C, G_2) \ \wedge \ \hat{e}(D, G_2) = \hat{e}(A \cdot C, X) \ \wedge \ \hat{e}(G_1, M_2) = \hat{e}(M_1, G_2)$.

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

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

Signing Group Elements. The above signature scheme allows us to sign group elements without revealing the exponent m. We will use this signature scheme for the Join/Issue protocol in our group blind signature scheme where the Issuer can sign signers' public keys without learning their secret keys which permits concurrent joining.

Note that we could alternatively, for instance, have used a variant of the original CL signature scheme [13] whose unforgeability reduces to the original LRSW assumption and required that each signer attaches an interactive Σ proof of knowledge of the underlying secret key which would require rewinding the signers to extract their secret keys from the Σ proof of knowledge to break the LRSW assumption. However, the latter approach is impractical as it requires rewinding and hence yields inefficient Join protocol which limits the number of users that can join concurrently.

5.2 Groth-Sahai(GS) Proofs

In [29, 30] 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 [24] 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 GS proof system are:

- Pairing Product Equation(PPE): For $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ we have $f(P, Q) = \hat{e}(P, Q)$.
- Multi-Scalar Multiplication(MSME1) in $\mathbb{G}_1$: For $P \in \mathbb{G}_1$, $y \in \mathbb{F}_p$ we have $f(P, y) = P^y$.
- Multi-Scalar Multiplication(MSME2) in $\mathbb{G}_2$: For $x \in \mathbb{F}_p$, $Q \in \mathbb{G}_2$ we have $f(x, Q) = Q^x$.
- Quadratic Equation(QE) in $\mathbb{F}_p$: For $x, y \in \mathbb{F}_p$ we have $f(x, y) = x \cdot y$.

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 (GSSetup, GSProve, GSVerify, GSExtract, GSSimSetup, GSSimProve). GSSetup takes as input the description of bilinear groups $\mathcal{P}$ and outputs a binding reference string $\mathsf{crs}_{\mathsf{GS}}$ and a trapdoor information xk which allows for witness extraction. GSProve takes as input a set of equations, the string $\mathsf{crs}_{\mathsf{GS}}$ and a witness and produces a proof π for the satisfiability of the equations. For clarity, we will underline the elements of the witness to distinguish them from public constants. We will write $\mathsf{GSProve}_{ZK}$ if the proof to be produced is zero-knowledge while we will write $\mathsf{GSProve}_{WI}$ if it is only witness indistinguishable. GSVerify takes as input a set of equations, a CRS, a proof π 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 $\mathsf{crs}_{\mathsf{GS}}$, a valid proof π 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, $\mathsf{crs}_{\mathsf{GS}Sim}$, and a trapdoor key sd that allows to simulate proofs. GSSimProve takes as input a simulated CRS, $\mathsf{crs}_{\mathsf{GS}Sim}$, and the simulation key sd and produces a simulated proof π'. The security of the proof system requires that strings $\mathsf{crs}_{\mathsf{GS}}$ and $\mathsf{crs}_{\mathsf{GS}Sim}$ 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, $\mathsf{crs}_{\mathsf{GS}}$, and a valid proof π, the algorithm GSVerify always accepts the proof.
- **Perfect Soundness:** On a correctly generated CRS, $\mathsf{crs}_{\mathsf{GS}}$, 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 $\mathsf{crs}_{\mathsf{GS}}$ output by GSSetup is computationally indistinguishable from the CRS $\mathsf{crs}_{\mathsf{GS}Sim}$ output by GSSimSetup. We also have for any PPT adversary $\mathcal{A}$ that is given $\mathsf{crs}_{\mathsf{GS}Sim}$ and is allowed to choose any statement y and two distinct witnesses w_0 and w_1 for the statement y that

$$Pr\left[\begin{array}{l} b \leftarrow \{0,1\}; \pi \leftarrow \mathsf{GSProve}(\mathsf{crs}_{\mathsf{GS}Sim}, w_b, y); b' \leftarrow \mathcal{A}(\pi) \\ : b = b' \wedge (w_0, y) \in R \wedge (w_1, y) \in R \end{array}\right] = \frac{1}{2} + \nu(\lambda).$$

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

$$\begin{aligned} &Pr\left[\begin{array}{c} (\mathsf{crs}_{\mathsf{GS}Sim}, sd) \leftarrow \mathsf{GSSimSetup}(\mathcal{P}); (w, y) \leftarrow \mathcal{A}(\mathsf{crs}_{\mathsf{GS}Sim}, sd); \\ \pi \leftarrow \mathsf{GSProve}(\mathsf{crs}_{\mathsf{GS}Sim}, w, y) : \mathcal{A}(\pi) = 1 \end{array}\right] \\ =&Pr\left[\begin{array}{c} (\mathsf{crs}_{\mathsf{GS}Sim}, sd) \leftarrow \mathsf{GSSimSetup}(\mathcal{P}); (w, y) \leftarrow \mathcal{A}(\mathsf{crs}_{\mathsf{GS}Sim}, sd); \\ \pi_{sim} \leftarrow \mathsf{GSSimProve}(\mathsf{crs}_{\mathsf{GS}Sim}, sd, y) : \mathcal{A}(\pi_{sim}) = 1 \end{array}\right], \end{aligned}$$

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

Randomizable GS Proofs. As noted by [4], 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 GSRandomize which takes as input a CRS $\mathsf{crs}_{\mathsf{GS}}$ and a proof π and outputs a proof π' which is a randomized version of the proof π.

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

5.3 Blind Signatures

Blind signatures as introduced by Chaum [16] 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 [31, 40] consists of two properties: blindness and unforgeability. Intuitively, blindness says that an adversarial signer cannot learn the message being signed or be able 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 [20, 2], the authors presented a blind signature scheme whose unforgeability reduces to the AWFCDH and q-ADHSDH assumptions. The blind signature scheme is given in Figure 3.

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.

$\underline{\mathsf{Setup}(1^\lambda)}$:
- $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, G_1, G_2) \leftarrow \mathsf{BGrpSetup}(1^\lambda)$.
- $\mathcal{P} \leftarrow (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, G_1, G_2)$.
- $(\mathsf{crs}_{\mathsf{GS}}, xk) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
- $F, K, T \leftarrow \mathbb{G}_1$.

$\underline{\mathsf{KeyGen}(\mathcal{P})}$:
- Choose $s \leftarrow \mathbb{Z}_p^*$ and set $S_1 \leftarrow G_1^s$, $S_2 \leftarrow G_2^s$.
- $\mathsf{sk} \leftarrow s$, $\mathsf{pk} \leftarrow (S_1, S_2)$.

$\underline{\mathsf{Verify}(\mathsf{pk}, m, \Sigma)}$:
- Parse Σ as $(\Omega_1, \Omega_2, \Omega_3)$.
- Return 1 if $\forall i \in [1,3]$: $\mathsf{GSVerify}(\mathsf{crs}_{\mathsf{GS}}, \Omega_i) = 1$. Else Return 0.

$\mathsf{Obtain}(\mathsf{pk}, m) \backslash \mathsf{Sign}(\mathsf{sk})$

Obtain $\longrightarrow$ Sign: Choose $q \leftarrow \mathbb{Z}_p^*$ and compute $M_i \leftarrow G_i^m$, $Q_i \leftarrow G_i^q$ for $i = 1, 2$ and $Co \leftarrow T^q \cdot M_1$.
Compute $\Psi_1 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}}, (M_1, M_2), \left\{\hat{e}(\underline{M_1}, G_2)\hat{e}(G_1^{-1}, \underline{M_2}) = 1\right\}\right)$.
$\Psi_2 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}}, (Q_1, Q_2), \left\{\hat{e}(\underline{Q_1}, G_2)\hat{e}(G_1^{-1}, \underline{Q_2}) = 1\right\}\right)$.
$\Psi_3 = \mathsf{GSProve}_{WI}\left(\mathsf{crs}_{\mathsf{GS}}, (M_1, Q_2), \left\{\hat{e}(T, \underline{Q_2})\hat{e}(\underline{M_1}, G_2) = \hat{e}(Co, G_2)\right\}\right)$.
Send $(Co, \Psi_1, \Psi_2, \Psi_3)$ to Sign.

Sign $\longrightarrow$ Obtain: If $\mathsf{GSVerify}(\mathsf{crs}_{\mathsf{GS}}, \Psi_i) \neq 1$ for any $i \in [1,3]$ Then Abort().
Choose $r, c \leftarrow \mathbb{Z}_p^*$,and compute $H \leftarrow (K \cdot T^r \cdot Co)^{\frac{1}{s+c}}$, $R_1 \leftarrow G_1^r, R_2 \leftarrow G_2^r, C_1 \leftarrow F^c, C_2 \leftarrow G_2^c$.
Send $\sigma \leftarrow (H, R_1, R_2, C_1, C_2)$ to Obtain.

Obtain: Compute $R_i' \leftarrow R_i \cdot Q_i$ 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 $\hat{e}(H, S_2 \cdot C_2) \neq \hat{e}(K \cdot M_1, G_2)\hat{e}(T, R_2')$.
Compute $\Omega_1 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}}, (R_1', R_2'), \left\{\hat{e}(\underline{R_1'}, G_2)\hat{e}(G_1^{-1}, \underline{R_2'}) = 1\right\}\right)$.
$\Omega_2 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}}, (C_1, C_2), \left\{\hat{e}(\underline{C_1}, G_2)\hat{e}(F^{-1}, \underline{C_2}) = 1\right\}\right)$.
$\Omega_3 = \mathsf{GSProve}_{WI}\left(\mathsf{crs}_{\mathsf{GS}}, (H, C_2, R_2'), \left\{\hat{e}(\underline{H}, S_2 \cdot \underline{C_2})\hat{e}(T^{-1}, \underline{R_2'}) = \hat{e}(K \cdot M_1, G_2)\right\}\right)$.
Output $\Sigma \leftarrow (\Omega_1, \Omega_2, \Omega_3)$.

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

A Note on the Message Space of the Blind Signature.

Note that although when defining the different algorithms of the blind signature scheme we express the message m as an exponent in $\mathbb{Z}_p$, it is adequate for the user to know the following two one-way functions of the exponent $M_1 \leftarrow G_1^m \in \mathbb{G}_1$ and $M_2 \leftarrow G_2^m \in \mathbb{G}_2$ where the well-formedness of such a pair can be verified by checking if $\hat{e}(M_1, G_2) = \hat{e}(G_1, M_2)$. Therefore, we never require knowledge of the exponent m itself and it is sufficient for the user to know a well-formed pair.

6 Techniques Used

In this section, we describe the techniques our constructions rely on.

Some previous constructions, e.g. [37], relied on using divertible proofs of knowledge whose interactive variants require many rounds of interaction and therefore require that the signer (e.g. the bank) remains online throughout the signing process which is impractical considering that, for example, a bank would have many daily transactions and it is resource-exhausting to remain online for a long time to complete each transaction. Alternatively, some schemes, e.g. [34, 37], relied on the Fiat-Shamir transformation to eliminate such an interaction but at the expense of relying on random oracles to prove their security.

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 and given a proof π, we can compute a new fresh proof $\pi' = \mathsf{GSRandomize}(\mathsf{crs}_{\mathsf{GS}}, \pi)$ even without knowing the witnesses used in π. The new proof π' is unlinkable to the original proof π.
- GS proofs are independent of any public terms (i.e. public monomials) in the equations being proved.[1] Thus, given a witness-indistinguishable GS proof, π, (i.e. one for an equation with a non-trivial public

[1] This observation was also independently noted by [21].

right-hand side τ), we can later (even without knowing the original witnesses in the proof) transform π into a related witness-indistinguishable/zero-knowledge proof π' 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.

Pairing product equations are of the form $\prod_{i=1}^{n} \hat{e}(X_i, Y_i) = \tau$. We can construct GS NIZK proofs for these types of equations if either τ is trivial (i.e. $\tau = 1$) or if we can factor τ by finding P_i, Q_i such that $\tau = \prod_{i=1}^{n} \hat{e}(P_i, Q_i)$. 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 τ. This will become clearer when we present our construction.

– 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.

7 Construction I

For efficiency reasons, in this construction we will use the NCL signature scheme from Section 5.1 to generate membership certificates upon joining. In Section 8, we show that this could be replaced by any other structure-preserving signature scheme. We start with an intuition for understanding our construction and then present the construction in detail in Figures 4 and 5.

7.1 Overview of the Construction

We now provide intuition for understanding our construction. The group public key gpk contains the public key for the NCL signature scheme (i.e. $\mathsf{pk}_{\mathsf{NCL}}$), the public values used in the blind signature scheme and two GS reference strings $\mathsf{crs}_{\mathsf{GS}_1}$ and $\mathsf{crs}_{\mathsf{GS}_2}$, which are used in constructing GS proofs used in the first round and the second round of the signing protocol respectively. Note here we can use the same reference string for both phases but we believe that separating the strings provides extra functionality such as being able to prevent 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.

The Issuer gets the secret signing key $\mathsf{sk}_{\mathsf{NCL}}$ for the NCL signature scheme described in Section 5.1. 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 $(\mathsf{sk}_i, \mathsf{pk}_i)$. 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. One can use any secure signature scheme for producing such a signature and we do not specify a particular scheme, for instance, either the automorphic signature scheme used in the signing protocol (since it is capable of signing its own keys) or the NCL signature scheme (since it is capable of signing group elements) can be used for this purpose. Since in our concrete construction we will use the NCL scheme for this purpose, WLOG we will assume that the SKg algorithm takes as input the same bilinear group description used in the GKg this is mainly for simplicity rather than generating a separate description for the NCL scheme used for this purpose.

The Issuer first verifies the signature sig_i and if it is valid, he issues a signature on pk_i using the NCL signature scheme and his secret issuing key it. 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 signing protocol in our construction $\langle \mathsf{Obtain}, \mathsf{Sign} \rangle$ is an interactive one and consists of two rounds between an external user who knows a secret message m (or simply a pair (M_1, M_2) where $M_i = G_i^m \in \mathbb{G}_i$ for $i = 1, 2$) and an anonymous group member Signer_i who possesses a secret group signing key $\mathbf{gsk}[i]$. In the first round, the user commits to the message using Pedersen commitment $Co = T^q \cdot G_1^m$ for some random $q \leftarrow \mathbb{Z}_p^*$ and sends the commitment Co along with GS proofs (Ψ_1, Ψ_2, Ψ_3) to prove that the commitment Co is indeed to the message $M_1 = G_1^m$ and that the message and the randomness pairs are well-formed.

In the second round of the protocol, Signer_i verifies the three GS proofs and if they are all 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. So the signer commits to the components (A, B, C, D) representing his membership certificate cert_i, his public verification key pk_i and the components (H, C_1, C_2) of the signature σ (which could identify the signer if they were sent in the clear). 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 $(\Omega_1, \Omega_2, \Omega_3)$ which prove that the signer has a valid certificate he got from the Issuer on his public verification key pk_i, GS proof Ω_4 to prove that his public verification key pk_i is well-formed, GS proofs (Ω_5, Ω_6) which prove that the signature was produced by the owner of public key pk_i, and the components R_1, R_2 of σ.

The user first checks that the six GS proofs Ω_i for $i \in [1, 6]$ are all valid. If so, the user first re-randomizes those proofs using the algorithm GSRandomize. The new proofs are unlinkable to the original ones. Note here that the proof Ω_6 is for the equation $\hat{e}(H, S_2 \cdot C_2) = \hat{e}(K \cdot Co, G_2)\hat{e}(T, R_2)$, and that Co and R_2 are public at this stage and hence not part of the witness for that proof.

Now, since the user has constructed the commitment, he knows the message M_1 and the randomness Q_1 used in the commitment. Therefore, the user updates the components R_1 and R_2 to include the randomness used in the commitment (by computing $R'_1 \leftarrow R_1 \cdot Q_1$ and $R'_2 \leftarrow R_2 \cdot Q_2$) and then decomposes the commitment and transforms the proof Ω_6 into a witness-indistinguishable/zero-knowledge proof based on the message M_1 instead of the commitment Co by adding the value R'_2 to the witness of proof Ω_6. The proof Ω_6 is now for the equation $\hat{e}(H, S_2 \cdot C_2)\hat{e}(T^{-1}, R'_2) = \hat{e}(K \cdot M_1, G_2)$, where H, S_2, C_2 and R'_2 are parts of the witness. In addition, the user produces a new GS zero-knowledge proof Ω_7 to prove that R'_1 and R'_2 hide the same exponent. The final group blind signature Σ is seven GS proofs Ω_i for $i = 1, \ldots, 7$.

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 GS 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 $\mathbf{ssk}[i]$.

7.2 Our Scheme

We now present our dynamic group blind signature scheme. We start with a scheme that achieves CPA-anonymity then we outline in Section 9 how to extend this scheme to make it fully anonymous. The scheme is shown in Figure 4 while the Obtain/Sign protocol is shown in Figure 5.

7.3 Efficiency Analysis

The final group blind signature Σ consists of seven GS proofs, Ω_i for $i \in [1, 7]$. Note that 2 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 $\Omega_i = (\underline{\theta_i}, \underline{\pi_i}) \in 2 \cdot \mathbb{G}_1^2 \times 2 \cdot \mathbb{G}_2^2$ 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 $20 \cdot |\mathbb{G}_1| + 24 \cdot |\mathbb{G}_2|$. The total number of GS commitments used in those proofs are 8 commitments in group $\mathbb{G}_1^2$ and 3 commitments in group $\mathbb{G}_2^2$ and therefore the total size of the signature is $36 \cdot |\mathbb{G}_1| + 30 \cdot |\mathbb{G}_2|$. Note that when signing, the group member does not have to re-create proofs Ω_1,Ω_2,Ω_3,Ω_4 and their corresponding commitments and instead just re-randomizes the original proofs.

7.4 Security Analysis

Theorem 2. *The scheme given in Figures 4 and 5 is a secure dynamic group blind signature scheme.*

Proof. Perfect correctness follows from the perfect completeness of GS proofs, the correctness of the NCL signature scheme used in the Join/Issue protocol and in producing the signature sig and the correctness of the blind signature scheme used in the Obtain/Sign protocol.

$\underline{\mathsf{GKg}(1^\lambda)}$
- $\mathcal{P} \leftarrow \mathsf{BGrpSetup}(1^\lambda)$.
- $F, K, T \leftarrow \mathbb{G}_1$.
- $(\mathsf{crs}_{\mathsf{GS}_1}, xk_1) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
- $(\mathsf{crs}_{\mathsf{GS}_2}, xk_2) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
- $((x, y), (X, Y)) \leftarrow \mathsf{NCLKeyGen}(\mathcal{P})$.
- $\mathsf{gpk} \leftarrow (\mathcal{P}, \mathsf{crs}_{\mathsf{GS}_1}, \mathsf{crs}_{\mathsf{GS}_2}, F, K, T, X, Y)$.
- $\mathfrak{ik} \leftarrow (x, y)$; $\mathfrak{ok} \leftarrow xk_2$.
- Return $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok})$.

$\underline{\mathsf{SKg}(\mathcal{P})}$:
- $(\mathbf{ssk}[i], \mathbf{spk}[i]) \leftarrow \mathsf{NCLKeyGen}(\mathcal{P})$.
- Return $(\mathbf{ssk}[i], \mathbf{spk}[i])$.

$\underline{\mathsf{GVf}(\mathsf{gpk}, m, \Sigma)}$:
- Parse Σ as $(\Omega_1, \Omega_2, \Omega_3, \Omega_4, \Omega_5, \Omega_6, \Omega_7)$.
- Return 1 if $\forall i \in [1, 7]$: $\mathsf{GSVerify}(\mathsf{crs}_{\mathsf{GS}_2}, \Omega_i) = 1$. Else Return 0.

$\underline{\mathsf{Open}(\mathsf{gpk}, \mathfrak{ok}, m, \Sigma)}$
- Parse $\mathfrak{ok}$ as xk_2.
- Parse gpk as $(\mathcal{P}, \mathsf{crs}_{\mathsf{GS}_1}, \mathsf{crs}_{\mathsf{GS}_2}, F, K, T, X, Y)$.
- $(\sigma, \mathsf{cert}, \mathsf{pk}) \leftarrow \mathsf{GSExtract}(\mathsf{crs}_{\mathsf{GS}_2}, xk_2, \Sigma)$.
- 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 $(0, (\sigma, \mathsf{cert}, \mathsf{pk}, \epsilon))$.

$\underline{\mathsf{Judge}(\mathsf{gpk}, i, \mathbf{spk}[i], m, \Sigma, \tau)}$
- Parse gpk as $(\mathcal{P}, \mathsf{crs}_{\mathsf{GS}_1}, \mathsf{crs}_{\mathsf{GS}_2}, F, K, T, X, Y)$.
- Parse τ as $(i, (H, C_j, R'_j), \mathsf{cert}, \mathsf{pk}, \mathsf{sig})$ for $j = 1, 2$.
- Parse pk as (S_1, S_2).
- If $i > 0$ and $\hat{e}(H, S_2 \cdot C_2) = \hat{e}(K \cdot G_1^m, G_2)\hat{e}(T, R'_2)$ 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 $\mathsf{NCLVerify}(\mathbf{spk}[i], \mathsf{pk}, \mathsf{sig}) = 1$ and $\mathsf{NCLVerify}((X, Y), \mathsf{pk}, \mathsf{cert}) = 1$ Then Return 1 Else Return 0.

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

Fig. 4. Our group blind signature scheme

Lemma 1. *The scheme is anonymous providing that* GS *proofs are hiding.*

Proof. We prove that

$$\mathsf{Adv}^{Anon}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{Hiding}_{\mathsf{GS},\mathcal{A}}(\lambda) + \nu(\lambda).$$

By the perfect zero-knowledge/witness-indistinguishability of GS proofs in the hiding setting we have that $\mathsf{Adv}^{Hiding}_{\mathsf{GS},\mathcal{A}}(\lambda) = 0$. Also, note that the adversary has a negligible advantage in telling apart a binding CRS from a hiding one and therefore this only negligibly changes $\mathcal{B}$ success probability and hence the scheme is anonymous.

We will use adversary $\mathcal{B}$ to construct an adversary $\mathcal{A}$ which can distinguish between witnesses of GS proofs in the hiding setting and hence break the zero-knowledge/witness-indistinguishabiliy properties of the GS proof system. We achieve a strong notion of anonymity where we allow $\mathcal{B}$ to learn the keys of the two signers which it will use for its challenge and therefore a signer would not be able to tell if the signature was produced by himself or by someone else.

The algorithm $\mathcal{A}$ is shown in Figure 6(a). Adversary $\mathcal{A}$ 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. The success probability of adversary

$\mathsf{Obtain}(\mathsf{gpk}, m) \backslash \mathsf{Sign}(\mathbf{gsk}[i])$	
$\underline{\mathsf{Obtain} \rightarrow \mathsf{Sign}}$:	Set $M_1 \leftarrow G_1^m$, $M_2 \leftarrow G_2^m$. Choose $q \leftarrow \mathbb{Z}_p^*$ and set $Q_1 \leftarrow G_1^q$, $Q_2 \leftarrow G_2^q$ and $Co \leftarrow T^q \cdot M_1$.
	Compute: $\Psi_1 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}_1}, (M_1, M_2), \{\hat{e}(\underline{M_1}, G_2)\hat{e}(G_1^{-1}, \underline{M_2}) = 1\}\right)$.
	$\Psi_2 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}_1}, (Q_1, Q_2), \{\hat{e}(\underline{Q_1}, G_2)\hat{e}(G_1^{-1}, \underline{Q_2}) = 1\}\right)$.
	$\Psi_3 = \mathsf{GSProve}_{WI}\left(\mathsf{crs}_{\mathsf{GS}_1}, (M_1, Q_2), \{\hat{e}(T, \underline{Q_2})\hat{e}(\underline{M_1}, G_2) = \hat{e}(Co, G_2)\}\right)$.
	Send $(Co, \Psi_1, \Psi_2, \Psi_3)$ **to** Sign.
$\underline{\mathsf{Sign} \rightarrow \mathsf{Obtain}}$:	If $\mathsf{GSVerify}(\mathsf{crs}_{\mathsf{GS}_1}, \Psi_i) \neq 1$ for $i \in [1,3]$ Then Abort().
	Choose $r, c \leftarrow \mathbb{Z}_p^*$ and set $H \leftarrow (K \cdot T^r \cdot Co)^{\frac{1}{s+c}}$, $R_1 \leftarrow G_1^r$, $R_2 \leftarrow G_2^r$, $C_1 \leftarrow F^c$ and $C_2 \leftarrow G_2^c$.
	Set $\sigma \leftarrow (H, R_1, R_2, C_1, C_2)$ and Parse pk_i as (S_1, S_2) and cert_i as (A, B, C, D).
	Compute $\Omega_1 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}_2}, (A, B), \left\{\hat{e}(\underline{A}, Y)\hat{e}(\underline{B}, G_2^{-1}) = 1\right\}\right)$.
	$\Omega_2 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}_2}, (B, C, S_2), \left\{\hat{e}(\underline{C}, G_2^{-1})\hat{e}(\underline{B}, \underline{S_2}) = 1\right\}\right)$.
	$\Omega_3 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}_2}, (A, C, D), \left\{\hat{e}(\underline{D}, G_2)\hat{e}(\underline{A}, X^{-1})\hat{e}(\underline{C}, X^{-1}) = 1\right\}\right)$.
	$\Omega_4 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}_2}, (S_1, S_2), \left\{\hat{e}(\underline{S_1}, G_2)\hat{e}(G_1^{-1}, \underline{S_2}) = 1\right\}\right)$.
	$\Omega_5 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}_2}, (C_1, C_2), \left\{\hat{e}(\underline{C_1}, G_2)\hat{e}(F^{-1}, \underline{C_2}) = 1\right\}\right)$.
	$\Omega_6 = \mathsf{GSProve}_{WI}\left(\mathsf{crs}_{\mathsf{GS}_2}, (H, C_2, S_2), \left\{\hat{e}(\underline{H}, \underline{S_2} \cdot \underline{C_2}) = \hat{e}(K \cdot Co, G_2)\hat{e}(T, R_2)\right\}\right)$.
	Send $(R_1, R_2, \zeta \leftarrow (\Omega_1, \Omega_2, \Omega_3, \Omega_4, \Omega_5, \Omega_6))$ **to** Obtain.
$\underline{\mathsf{Obtain}}$:	If $\mathsf{GSVerify}(\mathsf{crs}_{\mathsf{GS}_2}, \Omega_i) \neq 1$ for any $i \in [1,6]$ or $\hat{e}(R_1, G_2) \neq \hat{e}(G_1, R_2)$ Then Abort().
	Set $R_i' \leftarrow R_i \cdot Q_i$ for $i = 1, 2$.
	Compute $\Omega_i = \mathsf{GSRandomize}(\mathsf{crs}_{\mathsf{GS}_2}, \Omega_i)$ for all $i \in [1,6]$.
	Transform Ω_6 to $\Omega_6 = \mathsf{GSProve}_{WI}\left(\mathsf{crs}_{\mathsf{GS}_2}, (H, C_2, S_2, R_2')\right.$ $\left., \left\{\hat{e}(\underline{H}, \underline{S_2} \cdot \underline{C_2})\hat{e}(T^{-1}, \underline{R_2'}) = \hat{e}(K \cdot M_1, G_2)\right\}\right)$.[2]
	$\Omega_7 = \mathsf{GSProve}_{ZK}\left(\mathsf{crs}_{\mathsf{GS}_2}, (R_1', R_2'), \{\hat{e}(\underline{R_1'}, G_2)\hat{e}(G_1^{-1}, \underline{R_2'}) = 1\}\right)$.
	Output $\Sigma \leftarrow (\Omega_i,$ **for** $i \in [1,7])$.

[2] The transformation is done without knowledge of H, C_2 and S_2. Also, the proof can be easily made zero-knowledge.

Fig. 5. The Obtain/Sign protocol in our group blind signature scheme

$\mathcal{A}$ is given by $\mathsf{Adv}^{Hiding}_{\mathsf{GS},\mathcal{A}}(\lambda)$ which is defined as

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

where the experiment $\mathsf{Exp}^{Hiding-b}_{\mathsf{GS},\mathcal{A}}$ is defined in Figure 6(b).

Adversary $\mathcal{A}$ starts out by creating the group public key gpk, where $\mathcal{A}$ gets $\mathsf{crs}_{\mathsf{GS}_2}$ from its environment, chooses the Issuer's key $\mathsf{i\mathfrak{k}}$ and generates the rest of gpk itself. Note that the GS reference string $\mathsf{crs}_{\mathsf{GS}_1}$ is independent of $\mathsf{crs}_{\mathsf{GS}_2}$ and the proofs generated using $\mathsf{crs}_{\mathsf{GS}_1}$ are independent of those generated using $\mathsf{crs}_{\mathsf{GS}_2}$, hence $\mathcal{A}$ could choose $\mathsf{crs}_{\mathsf{GS}_1}$ to be a binding CRS if it needs a guarantee that $\mathcal{B}$ indeed knows the message it is using in the challenge. Adversary $\mathcal{A}$ calls $\mathcal{B}$ with input $(\mathsf{gpk}, \mathsf{i\mathfrak{k}} = (x, y))$.

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

- CH_b: Adversary $\mathcal{A}$ will use the group signing keys of group 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. Note that 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 $\mathcal{A}$ then forwards the details $(\mathbf{gsk}[i_0], \mathbf{gsk}[i_1], Co, \sigma_0, \sigma_1, R_1, R_2)$ to its Prove oracle and receives a proof (ζ). $\mathcal{A}$ then forwards (ζ, R_1, R_2) to $\mathcal{B}$.
- Open: In our initial construction, we cannot answer this query. This is because GS proofs are not simulation-sound, and hence we cannot extract while simulating the proofs. We show in Section 9, how to make our scheme CCA-anonymous where we can answer Open queries in the anonymity game.

When the scheme is made fully anonymous, to realize the predicate IdentifyTrans, we can use the extraction key xk_1 associated with the the GS reference string $\mathsf{crs}_{\mathsf{GS}_1}$ to extract the message from the GS proofs adversary

$\mathcal{B}$ sends in the first round of the signing protocol. We can also use the Opener's key to extract the components of the signature σ. This allows us to check if a message/signature pair resulted from the interaction with the CH_b oracle.

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

Algorithm: $\mathcal{A}(\mathcal{P}, \mathsf{crs}_{\mathsf{GS}_2} : \mathsf{Prove}_b(\cdot,\cdot,\cdot))$
- $x, y \leftarrow \mathbb{Z}_p^*$; $X \leftarrow G_2^x$; $Y \leftarrow G_2^y$.
- $F, K, T \leftarrow \mathbb{G}_1$.
- $(\mathsf{crs}_{\mathsf{GS}_1}, xk_1) \leftarrow \mathsf{GSSetup}(\mathcal{P})$.
- $\mathsf{gpk} \leftarrow (\mathcal{P}, \mathsf{crs}_{\mathsf{GS}_1}, \mathsf{crs}_{\mathsf{GS}_2}, F, K, T, X, Y)$; $\mathfrak{ik} \leftarrow (x, y)$.
- $b^* \leftarrow \mathcal{B}^{\langle\cdot,\cdot\rangle, \mathsf{CH}_b(\cdot,\cdot,\cdot))^1}(\mathsf{gpk}, \mathfrak{ik} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{ModifyReg}(\cdot,\cdot), \mathsf{SSK}(\cdot))$.
- Return b^*.

Experiment: $\mathsf{Exp}^{Hiding-b}_{\mathsf{GS},\mathcal{A}}(\lambda)$
- $\mathcal{P} \leftarrow \mathsf{BGrpSetup}(1^\lambda)$.
- $(\mathsf{crs}_{\mathsf{GS}}, sd) \leftarrow \mathsf{GSSimSetup}(\mathcal{P})$.
- $(y, w_0, w_1, \mathsf{state}_{\mathsf{find}}) \leftarrow \mathcal{A}_{\mathsf{find}}(\mathsf{crs}_{\mathsf{GS}}, \mathcal{P})$.
- 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. 6. Adversary $\mathcal{A}$ against the hiding property of the GS proof system (left) and security experiment for GS proof system hiding property (right)

Lemma 2. *The scheme is traceable providing that the* DH-LRSW *assumption holds and that* GS *proofs are sound.*

Proof. We prove that

$$\mathsf{Adv}^{Trace}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DH\text{-}LRSW}}_{\mathcal{A}_1}(\lambda) + \mathsf{Adv}^{Soundness}_{\mathsf{GS},\mathcal{A}_2}(\lambda).$$

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. In the soundness setting, we have that GS proofs are perfectly sound and therefore $\mathsf{Adv}^{Soundness}_{\mathsf{GS},\mathcal{A}_2}(\lambda) = 0$. So we have $\mathsf{Adv}^{Trace}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DH\text{-}LRSW}}_{\mathcal{A}_1}(\lambda)$. Using adversary $\mathcal{B}$ that wins the traceability game, we can construct an adversary $\mathcal{A}_1$ that breaks the DH-LRSW assumption. Since we have that $\mathsf{Adv}^{\mathsf{DH\text{-}LRSW}}_{\mathcal{A}_1}(\lambda) \leq \nu(\lambda)$, we also have that $\mathsf{Adv}^{Trace}_{\mathsf{GBS},\mathcal{B}}(\lambda) \leq \nu(\lambda)$ and therefore the scheme is traceable. Note that in this experiment, the Opener uses the proof system's extraction key xk_2 to open signatures 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 elements (X, Y) from the DH-LRSW problem it is trying to solve. Adversary $\mathcal{A}_1$ will call $\mathcal{B}$ as an oracle, it does this by running the GKg algorithm and initiating GS proofs in the binding setting (i.e. $\mathsf{crs}_{\mathsf{GS}_2}$ is chosen to be a binding CRS), except that the Issuer's public key (X, Y) is set to be its own input.

When calling $\mathcal{B}$, adversary $\mathcal{A}_1$ needs to pass as input gpk and $\mathfrak{ok}$, obtained from running GKg. All $\mathcal{B}$'s queries are answered by adversary $\mathcal{A}$ 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 DH-LRSW 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 $\mathcal{A}_1$ will extract the certificate components (A, B, C, D) and the verification key $pk = (S_1, S_2)$ from the signature Σ and returns $((S_1, S_2)(A, B, C, D))$ as its answer in the DH-LRSW game.

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

Lemma 3. *The scheme is non-frameable if the blind signature scheme is unforgeable (i.e. If the* AWFCDH *and* q-ADHSDH *assumptions hold and* GS *proofs are sound) and the* NCL *signature scheme (used for producing* sig *upon joining the group) is unforgeable.*

Proof. We prove that

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

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 when instantiated using a binding CRS, we have that $\mathsf{Adv}^{Soundness}_{\mathsf{GS},\mathcal{A}_3}(\lambda) = 0$.

Also, the signature schemes $\mathsf{BS-DS}$ used in the signing protocol and the NCL 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 $\mathsf{Adv}^{Unforg}_{\mathsf{BS-DS},\mathcal{A}_1}(\lambda) \leq \nu(\lambda)$, $\mathsf{Adv}^{Unforg}_{\mathsf{NCL},\mathcal{A}_2}(\lambda) \leq \nu(\lambda)$ 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 [20]. Since the GS proofs are perfectly sound in the binding setting, unforgeability of the blind signature scheme reduces to producing a forgery on a new message that the adversary did not query the sign oracle for the underlying signature scheme $\mathsf{BS-DS}$ on.

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.

Adversary $\mathcal{A}_1$ 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. $\mathsf{crs}_{\mathsf{GS}_2}$ is chosen to be a binding CRS). Adversary $\mathcal{A}_1$ calls $\mathcal{B}$ on input gpk, the Issuer's key $\mathfrak{ik} = (x, y)$, and the Opener's key $\mathfrak{ok} = xk_2$.

Let j be the identity of t^{th} signer $\mathcal{B}$ creates, where $t \in \{1, 2, \ldots, n(\lambda)\}$. 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 by adversary $\mathcal{A}_1$ 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 $\mathcal{A}_1$ has guessed. Otherwise, $\mathcal{A}_1$ uses the extraction key xk_2 to extract the extra signature $\sigma^* = (H^*, R_1^*, R_2^*, C_1^*, C_2^*)$ 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 NCL 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. Adversary $\mathcal{A}_2$ 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. $\mathsf{crs}_{\mathsf{GS}_2}$ is chosen to be a binding CRS). Adversary $\mathcal{A}_2$ calls $\mathcal{B}$ on input gpk, the Issuer's key $\mathfrak{ik} = (x, y)$, and the Opener's key $\mathfrak{ok} = xk_2$. 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 by adversary $\mathcal{A}_2$ 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.

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 security of the DH-LRSW assumption, we have that $\mathsf{Adv}^{Unforg}_{\mathsf{NCL},\mathcal{A}_2}(\lambda) \leq \nu(\lambda)$.

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 4. *The scheme is blind providing that the underlying blind signature scheme is blind.*

Proof. We prove that

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

By the perfect zero-knowledge/witness-indistinguishability 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 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 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 Pedersen commitments which are perfectly hiding and have identical distributions. Moreover, if we slightly modify the equation being proved in proof Ψ_3 in the first round of the signing protocol where we add G_2^{-1} in the third pairings to the set of witnesses, and also adding a proof for the multi-scalar multiplication equation $G_2^{-1} \cdot G_2 = 1$, 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 Pedersen commitment (i.e. adversary $\mathcal{A}_2$ that wins the security game in Figure 7) 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 $\mathsf{crs}_{\mathsf{GS}_1}$ and $\mathsf{crs}_{\mathsf{GS}_2}$ to hiding strings, the proofs $\mathcal{B}$ sees are distributed uniformly and hence

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

Fig. 7. Security experiment for Pedersen commitment scheme's hiding property

again 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 6(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.

8 Constructions based on Non-Interactive Assumptions

In the construction in Section 7, we used the new NCL signature scheme for issuing certificates for new group members. This choice was merely based on efficiency reasons. The NCL signature scheme could be replaced by any other structure-preserving signature scheme [2]. For instance, we can use the asymmetric automorphic signature scheme (AFPV) from [20] which has the extra beneficial property of being able to sign its own public keys. Although this will result in a less efficient construction than the NCL based one, we get the bonus of basing the security of the construction on non-interactive complexity assumptions.

The construction is straightforward. We just note here that since the certification scheme can issue certificates on its own public keys, there is no need for the new group member to choose a new pair of secret/public keys for signing and instead he just uses the same key pair for signing. To protect against a corrupt Issuer, we ask that each member signs his own public key when joining.

Replacing the NCL signature scheme used for certification with the asymmetric automorphic signature scheme from [20] yields group blind signatures of size $42 \cdot |\mathbb{G}_1| \times 38 \cdot |\mathbb{G}_2|$. This is less efficient than the NCL based construction. However, unlike the NCL based construction, the security of this construction is based merely on non-interactive complexity assumptions. Furthermore, since the equations for verifying the certificates and signatures are the same, deploying batch verification techniques [23, 9] for Groth-Sahai proofs could improve the efficiency of the verification algorithm.

9 Achieving Full Anonymity & Other Instantiations

Achieving Full Anonymity: Since GS proofs are not simulation-sound and hence when instantiated in the hiding setting we can no longer apply the 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 [42, 43] 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. [27, 39].

Other Instantiations: Since we are interested in settings deployed over asymmetric pairings, we can alternatively instantiate GS proofs under the symmetric decisional linear (SDLIN) assumption presented in [24].

10 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

The author would like to thank Nigel Smart and Bogdan Warinschi for many helpful discussions 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. 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`.
4. 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.
5. 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.
6. M. Bellare and P. Rogaway. Random oracles are practical: A Paradigm for Designing Efficient Protocols. In *ACM-CCS 1993*, ACM, pp. 62–73.
7. 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.
8. 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.
9. 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`.
10. D. Boneh and X. Boyen. Short signatures without random oracles. In *Eurocrypt 2004*, Springer LNCS 3027, 56–73, 2004.
11. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures. In *CRYPTO 2004*, Springer LNCS 3152, 227–242, 2004.
12. X. Boyen and B.Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC 2007*, Springer LNCS 4450, 1–15, 2007.
13. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, Springer LNCS 3152, 56–72, 2004.
14. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO 1997*, Springer LNCS 1294, 410–424, 1997.
15. R. Canetti, H. Krawczyk, and J. Nielsen. Relaxing Chosen-Ciphertext Security. In *CRYPTO 2003*, Springer LNCS 2729, 565–582, 2003.
16. D. Chaum. Blind signatures for untraceable payments. In *CRYPTO 1982*, Plenum Press, 199–203, 1983.
17. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT 1991*, Springer LNCS 547, 257–265, 1991.
18. 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`.
19. 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.
20. 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`.
21. 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`.
22. S. Galbraith, K. Paterson and N.P. Smart. Pairings for cryptographers. In *Discrete Applied Mathematics*, **156**, 3113–3121, 2008
23. 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.
24. E. Ghadafi, N.P. Smart and B. Warinschi. Groth-Sahai proofs revisited. In *PKC 2010*, Springer LNCS 6056, 177–192, 2010.
25. S. Goldwasser and S. Micali. Probabilistic encryption. In *Journal of Computer and System Sciences*, 28:2, 270–299, 1984.
26. M. Green and S. Hohenberger. Universally Composable Adaptive Oblivious Transfer. In *Asiacrypt 2008*, Springer LNCS 5350, 179–197, 2008.
27. J. Groth. Rerandomizable and Replayable Adaptive Chosen Ciphertext Attack Secure Cryptosystems. In *Theory of cryptography– TCC 2004*, Springer-Verlag LNCS 2951, 152–170, 2004.
28. J. Groth. Fully anonymous group signatures without random oracles. In *Asiacrypt 2007*, Springer LNCS 4833, 164–180, 2007.

29. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT 2008*, Springer LNCS 4965, 415–432, 2008.
30. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups (full version). `http://www.brics.dk/~jg/WImoduleFull.pdf`
31. A. Juels, M. Luby and R. Ostrovsky. Security of blind digital signatures. In *Advances in Cryptology – CRYPTO '97*, Springer LNCS 1294, 150–164, 1997.
32. 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.
33. A. Lysyanskaya, R. Rivest, A. Sahai and S. Wolf. Pseudonym systems. In *Selected Areas in Cryptography – SAC 1999*, Springer LNCS 1758, 184–199, 1999.
34. 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.
35. U. Maurer. Abstract models of computation in cryptography. In *Cryptography and Coding 2005*, Springer LNCS 3796, 1–12, 2005.
36. 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.
37. 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.
38. T. Okamoto and K. Ohta. Divertible Zero Knowledge Interactive Proofs and Commutative Random Self-Reducibility. In *EUROCRYPT 1989*, Springer LNCS 434, 134–149, 1990.
39. M. Prabhakaran and M. Rosulek. Rerandomizable RCCA encryption. In *Advances in Cryptology – CRYPTO 2007*, Springer-Verlag LNCS 4622, 517–534, 2007.
40. D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, **13(3)**, 361–396, 2000.
41. 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.
42. A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In FOCS 1999, 543–553, 1999.
43. A. Sahai. Simulation-Sound Non-Interactive Zero Knowledge. In Manuscript, 2001.
44. C.P. Schnorr. Efficient signature generation by smart cards. In *Journal of Cryptology*, **4**, 161–174, 1991.
45. J. T. Schwartz. Fast Probabilistic Algorithms for Verification of Polynomial Identities. In *J. ACM*, **27**, 701–717, 1980.
46. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Eurocrypt 1997*, Springer LNCS 1233, 256–266, 1997.
47. B. Waters. Efficient Identity-Based Encryption Without Random Oracles. In *EUROCRYPT 2005*, Springer LNCS 3494, 114–127, 2005.

A *SXDH* Instantiation of GS Proof System

Here we show how GS proof system can be instantiated under the SXDH assumption. Let $\mathcal{P} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_1, G_2)$ 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. To generate the CRS, the trusted party generates, for $i = 1, 2$, $a_i, t_i \in \mathbb{F}_p^*$ at random and computes $H_i = G_i^{a_i}$, $U_i = G_i^{t_i}$, $V_i = H_i^{t_i}$, and $\mathcal{U}_{i,1} = (G_i, H_i) \in \mathbb{B}_i$.

- 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 $xk = (a_1, a_2)$.
- 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 $\{\mathcal{U}_1, \mathcal{U}_2\} \in \mathbb{B}_1^2 \times \mathbb{B}_2^2$ where $\mathcal{U}_i = \{\mathcal{U}_{i,1}, \mathcal{U}_{i,2}\}$. Under the SXDH assumption one cannot tell a binding key from a hiding key. To aid what follows, we first set $\mathcal{W}_i = \mathcal{U}_{i,2} \cdot (1, G_i) = (W_{i,1}, W_{i,2}) \in \mathbb{B}_i$.

GS **Commitments**.

- **Committing to Group Elements:** To commit to $X \in \mathbb{G}_i$, randomly choose $r_1, r_2 \leftarrow \mathbb{F}_p$ and compute $\mathrm{comm}_{\mathrm{i}}(X, r_1, r_2) = (1, X) \cdot \mathcal{U}_{i,1}^{r_1} \cdot \mathcal{U}_{i,2}^{r_2} \in \mathbb{B}_i$. GSExtract then extracts the witness from a commitment $c = (C_1, C_2) \in \mathbb{B}_i$ by computing $\frac{C_2}{C_1^{a_i}}$.

– **Committing to Exponents:** To commit to $x \in \mathbb{F}_p$, randomly choose $r \leftarrow \mathbb{F}_p$ and compute $\text{comm}_i(x, r) = \mathcal{W}_i^x \cdot \mathcal{U}_{i,1}^r \in \mathbb{B}_i$. $\mathsf{GSExtract}$ then extracts the witness from a commitment $c = (G_i^{c_1}, G_i^{c_2})$ by computing $c_2 - a_i c_1$. 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 $(\pi, \theta) \in \mathbb{B}_2 \times \mathbb{B}_1$ for the satisfiability of each equation. Proofs for linear equations are simpler and consists 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 [30].

B DH-LRSW in the Generic Group Model

Here we show that the DH-LRSW assumption holds in the generic group model [46, 35].

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* $\mathsf{O}_{\mathsf{DH\text{-}LRSW}}$*. 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 $\mathsf{O}_{\mathsf{DH\text{-}LRSW}}(\sigma_1, \sigma_2)$.
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.

The Adversary Output: 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

$$P_B = P_A \cdot Y \tag{1}$$
$$P_C = P_B \cdot P_{M_2} \tag{2}$$
$$P_D = X \cdot (P_A + P_C) \tag{3}$$
$$P_{M_1} = P_{M_2} \tag{4}$$

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

$$v_0 + \sum_{i=1}^{q} v_{1,i} \cdot A_i + \sum_{i=1}^{q} v_{2,i} \cdot A_i \cdot Y + \sum_{i=1}^{q} v_{3,i} \cdot X \cdot A_i \cdot (1 + r_i \cdot Y) \tag{5}$$

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

By (1) and (2) 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 (3) 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 (5) 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:

$$P_{1,j}(x, y, a_i) = P_{1,j'}(x, y, a_i) \tag{6}$$
$$P_{2,j}(x, y, a_i) = P_{2,j'}(x, y, a_i) \tag{7}$$
$$P_{T,j}(x, y, a_i) = P_{T,j'}(x, y, a_i) \tag{8}$$

Since the maximum degree of any polynomial in list $\mathcal{G}_1 \leq 3$, by applying [46][Lemma 1], we have that the probability of Equation (6) 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 (7) holding is $\leq \frac{1}{p}$. Finally, the probability of Equation (8) 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 $\epsilon \leq \frac{(q_G+q_P+4q_O+4)^2 \cdot 4}{p}$.

C Details of the Alternative Experiment for Anonymity

We give here the details of the alternative security experiment that is appropriate for the case where the signing protocol is of strong unforgeability.

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

- $(\mathsf{gpk}, \mathfrak{ik}, \mathfrak{ok}) \leftarrow \mathsf{GKg}(1^\lambda)$; $\mathsf{HSL} \leftarrow \emptyset$; $\mathsf{CSL} \leftarrow \emptyset$; $\mathsf{BSL} \leftarrow \emptyset$.
- $(i_0, i_1, \mathsf{state}_{\mathsf{find}}) \leftarrow \mathcal{A}_{\mathsf{find}}(\mathsf{gpk}, \mathfrak{ik} : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{ModifyReg}(\cdot,\cdot), \mathsf{Open}(\cdot,\cdot), \mathsf{SSK}(\cdot))$.
- $(\Sigma_b, m, \mathsf{state}_{\mathsf{obtain}}) \leftarrow \mathcal{A}^{\langle \cdot, \mathsf{CH}_b(i_0,i_1)\rangle^1}_{\mathsf{obtain}}(\mathsf{state}_{\mathsf{find}})$.
- $(j, \tau) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathfrak{ok}, \mathbf{reg}, m, \Sigma_b)$.
- If $j \neq i_b$ Then Return $\perp$.
- $b^* \leftarrow \mathcal{A}_{\mathsf{guess}}(\mathsf{state}_{\mathsf{obtain}}, \Sigma_b : \mathsf{CrptS}(\cdot,\cdot), \mathsf{SndToS}(\cdot,\cdot), \mathsf{ModifyReg}(\cdot,\cdot), \mathsf{Open}(\cdot,\cdot), \mathsf{SSK}(\cdot))$.
- Return b^*.

Fig. 8. Alternative experiment for the anonymity

In the experiment in Figure 8, the adversary in the guess phase is not allowed to query the Open oracle on the pair (m, Σ_b).