Fully-Anonymous Short Dynamic Group Signatures Without Encryption

David Derler and Daniel Slamanig

IAIK, Graz University of Technology, Austria
{david.derler|daniel.slamanig}@tugraz.at

Abstract. Group signatures are an important privacy-enhancing tool which allow members of a group to anonymously produce signatures on behalf of the group. Ideally, group signatures are dynamic and thus allow to dynamically enroll new members to a group. For such schemes Bellare et al. (CT-RSA’05) proposed a strong security model (BSZ model) that preserves anonymity of a group signature even if an adversary can see arbitrary key exposures or arbitrary openings of other group signatures. All previous constructions achieving this strong security notion follow the so called sign-encrypt-prove (SEP) paradigm. In contrast, all known constructions which avoid this paradigm and follow the alternative “without encryption” paradigm introduced by Bichsel et al. (SCN’10), only provide a weaker notion of anonymity (which can be problematic in practice). Until now, it was not clear if constructions following this paradigm, while also being secure in the strong BSZ model, even exist. In this paper we positively answer this question by providing a novel approach to dynamic group signature schemes following this paradigm, which is a composition of structure preserving signatures on equivalence classes (Asiacrypt’14) and other standard primitives. Our results are interesting for various reasons: We can prove our construction following this “without encryption” paradigm secure in the strong BSZ model without requiring random oracles. Moreover, when opting for an instantiation in the ROM, the so obtained scheme is extremely efficient. It outperforms existing constructions following the SEP paradigm and being secure in the BSZ model regarding computational efficiency by some orders of magnitude and even yields shorter signatures. Regarding constructions providing a weaker anonymity notion than BSZ, we surprisingly outperform the popular short BBS group signature scheme (Crypto’04) and even obtain shorter signatures.

1 Introduction

Group signatures, initially introduced by Chaum and van Heyst [CvH91], allow a group manager to set up a group so that every member of this group can later anonymously sign messages on behalf of the group. Thereby, a dedicated authority (called opening authority) can open a given group signature to determine the identity of the actual signer. Group signatures were first rigorously formalized for static groups by Bellare et al. in [BMW03]. In this setting, all members are determined at setup and are also given their honestly generated keys at setup. This model was later extended to the dynamic case by Bellare et al. in [BSZ05] (henceforth denoted as BSZ model), where new group members can be dynamically enrolled to the group. Further, it separates the role of the issuer and the opener such that they can operate independently. Moreover, the BSZ model requires a strong anonymity notion, where anonymity of a group signature is preserved even if the adversary can see arbitrary key exposures and arbitrary openings of other group signatures.

The authors have been supported by EU H2020 project PRISMACLOUD, grant agreement no.644962.
A widely used paradigm to construct group signatures is the sign-encrypt-proof (SEP) paradigm [CS97]. Here, a signature is essentially an encrypted membership certificate together with a signature of knowledge, where the signer demonstrates knowledge of some signed value in the ciphertext [ACJT00, BBS04, NS04, BSZ05, DP06, BW07, BW06, Gro07, LPY15]. As an alternative to this paradigm, Bichsel et al. in [BCN10] proposed an elegant design paradigm for group signatures which does not require encryption to produce signatures. Essentially, they use a signature scheme which supports (1) randomization of signatures so that multiple randomized versions of the same signature are unlinkable, and (2) efficiently proving knowledge of a signed value. In their construction, on joining the group, the issuer uses a such signature scheme to sign a commitment to the user’s secret key. The user can then produce a group signature for a message by randomizing the signature and computing a signature of knowledge for the message, which demonstrates knowledge of the signed secret key. Bichsel et al. proposed an instantiation based on the randomizable pairing-based Camenisch-Lysyanskaya (CL) signature scheme [CL04] (whose EUF-CMA security is based on the interactive LRSW assumption). Recently, Pointcheval and Sanders [PS16] proposed another randomizable signature scheme (whose EUF-CMA security is proven in the generic group model), which allows to instantiate the approach due to Bichsel et al. more efficiently. For completeness, we also mention a very recent construction of group signatures from lattice assumptions by Libert et al. [LLM+16] following this paradigm.

However, a drawback of existing constructions following the “without encryption” paradigm is that they rely on a security model that is weaker than the BSZ model [BSZ05]. In particular, anonymity only holds for users whose keys do not leak, which essentially means that once a user key leaks, all previous signatures of this user can potentially be attributed to this user. Furthermore, the model in [BCN10] assumes that the opening authority and the issuing authority are one entity, meaning that the issuer can identify all signers when seeing group signatures. This can be highly problematic in practical applications of group signatures. Finally, we also want to mention the model which is used to prove the security (and in particular anonymity) of the popular BBS group signature scheme due to Boneh et al. [BBS04]. This model is a relaxation of the BSZ model, and in particular weakens anonymity so that the adversary can not request openings for signatures. Henceforth, we refer to this anonymity notion as CPA-full anonymity, whereas we use CCA2-full anonymity to refer to anonymity in the sense of BSZ.

Contribution. In this paper we propose a novel approach to construct group signatures following the paradigm of Bichsel et al., i.e., without including a ciphertext in the group signature. In particular, our approach is a composition of structure preserving signatures on equivalence classes (SPS-EQ) [HS14], EUF-CMA secure digital signatures, IND-CCA2 secure encryption, non-interactive zero-knowledge proofs of knowledge, and signatures of knowledge. Although these tools may sound quite heavy, we obtain surprisingly efficient group signatures, which can be proven secure in the strongest model for dynamic group signatures, i.e., the BSZ model. In doing so, we obtain the first construction following the “without encryption” paradigm which achieves this strong security notion (i.e., CCA2-full anonymity). Thus, we can positively answer the question whether such constructions are possible at all. When opting for an instantiation of our construction in the random oracle model, and comparing it with existing constructions having security proofs in the strong BSZ model, we outperform them in terms of computational efficiency by some orders of magnitude. As an additional bonus, our scheme even yields shorter signatures when compared to existing constructions. Moreover, when compared to the popular BBS group signature scheme [BBS04] (which only achieves CPA-full anonymity), we surprisingly obtain significantly better computational efficiency and even shorter signatures. Finally, when compared to existing instantiations in the vein of Bichsel et al. (who provide a substantially weaker anonymity notion and do not separate the issuer and the opener), we obtain compara-
ble computational efficiency, while the stronger anonymity comes at the cost of slightly larger signatures.

2 Preliminaries

In this section, we provide some preliminaries and recall the required primitives.

Notation. Let \( x \xleftarrow{} X \) denote the operation that picks an element \( x \) uniformly at random from a finite set \( X \). A function \( \epsilon : \mathbb{N} \rightarrow \mathbb{R}^+ \) is called negligible if for all \( c > 0 \) there is a \( k_0 \) such that \( \epsilon(k) < 1/k^c \) for all \( k > k_0 \). In the remainder of this paper, we use \( \epsilon \) to denote such a negligible function. We write \( y \xleftarrow{} A(x) \) to denote that the output of algorithm \( A \) on input \( x \) is assigned to \( y \) and write \( y \xleftarrow{} A(x; r) \) to make the random coins \( r \) of \( A \) explicit.

Let \( G_1 = \langle P \rangle, G_2 = \langle \hat{P} \rangle, \) and \( G_T \) be groups of prime order \( p \). A bilinear map \( e : G_1 \times G_2 \rightarrow G_T \) is a map, where it holds for all \((P, Q, a, b) \in G_1 \times G_2 \times \mathbb{Z}_p^* \) that \( e(aP, bQ) = e(P, Q)^{ab} \), and \( e(P, \hat{P}) \neq 1 \), and \( e \) is efficiently computable. We assume the Type-3 setting, where \( G_1 \neq G_2 \) and no efficiently computable isomorphism \( \psi : G_2 \rightarrow G_1 \) is known.

Definition 1 (Bilinear Group Generator). Let \( \text{BGGen} \) be an algorithm which takes a security parameter \( \kappa \) and generates a bilinear group \( \text{BG} = (p, G_1, G_2, G_T, e, P, \hat{P}) \) in the Type-3 setting, where the common group order \( p \) of the groups \( G_1, G_2 \) and \( G_T \) is a prime of bitlength \( \kappa \), \( e \) is a pairing and \( P \) and \( \hat{P} \) are generators of \( G_1 \) and \( G_2 \), respectively.

Based on this, we provide the required cryptographic hardness assumptions.

Discrete Logarithm Assumption (DL). Let \( G = \langle P \rangle \) be a group of prime order \( p \), such that \( \log_2 p = \kappa \). Then, for all PPT adversaries \( A \) there exists a negligible function \( \epsilon(\cdot) \) such that:

\[
\Pr \left[ a \xleftarrow{\$} \mathbb{Z}_p, \ c \xleftarrow{\$} A(P, aP) : \ a = c \right] \leq 1/2 + \epsilon(\kappa).
\]

Decisional Diffie-Hellman Assumption (DDH). Let \( G = \langle P \rangle \) be a group of prime order \( p \), such that \( \log_2 p = \kappa \). Then, for all PPT adversaries \( A \) there exists a negligible function \( \epsilon(\cdot) \) such that:

\[
\Pr \left[ b \xleftarrow{\$} \{0,1\}, r, s, t \xleftarrow{\$} \mathbb{Z}_p, \ b^* \xleftarrow{\$} A(P, rP, sP, b \cdot (rs) + (1 - b) \cdot tP) : \ b = b^* \right] \leq 1/2 + \epsilon(\kappa).
\]

Symmetric External Diffie-Hellman Assumption (SXDH). Let \( \text{BG} \) be a bilinear group generated by \( \text{BGGen} \). Then, the SXDH assumption states that the DDH assumption holds in \( G_1 \) and \( G_2 \).

Additionally, we introduce a plausible assumption in the Type-3 bilinear group setting.

Computational co-Diffie-Hellman Inversion Assumption (co-CDHI): Let \( \text{BG} \xleftarrow{} \text{BGGen}(\kappa) \). The co-CDHI assumption states that for all PPT adversaries \( A \) there exists a negligible function \( \epsilon(\cdot) \) such that:

\[
\Pr \left[ a \xleftarrow{\$} \mathbb{Z}_p, \ C \xleftarrow{\$} A(\text{BG}, aP, 1/a\hat{P}) : \ C = 1/aP \right] \leq \epsilon(\kappa).
\]

Digital Signature Schemes. Subsequently, we recall a definition of digital signature schemes.

Definition 2 (Digital Signatures). A digital signature scheme \( \text{DSS} \) is a triple \((\text{KeyGen}, \text{Sign}, \text{Verify})\) of PPT algorithms, which are defined as follows:
KeyGen(1^\kappa) : This algorithm takes a security parameter \kappa as input and outputs a secret (signing) key sk and a public (verification) key pk with associated message space \mathcal{M} (we may omit to mention the message space \mathcal{M}).

Sign(sk, m) : This algorithm takes a secret key sk and a message m \in \mathcal{M} as input and outputs a signature \sigma.

Verify(pk, m, \sigma) : This algorithm takes a public key pk, a message m \in \mathcal{M} and a signature \sigma as input and outputs a bit b \in \{0, 1\}.

Besides correctness we require existential unforgeability under adaptively chosen message attacks (EUF-CMA) [GMR88]. Subsequently, we recall formal definitions of these properties.

Definition 3 (Correctness). A DSS is correct, if for all \kappa, all (sk, pk) \leftarrow KeyGen(1^\kappa) and all m \in \mathcal{M} it holds that Pr[Verify(pk, m, Sign(sk, m))] = 1 = 1.

Definition 4 (EUF-CMA). A DSS is EUF-CMA secure, if for all PPT adversaries \mathcal{A} there is a negligible function \epsilon(\cdot) such that

\Pr[(sk, pk) \leftarrow KeyGen(1^\kappa), (m^*, \sigma^*) \leftarrow \mathcal{A}^{Sign}(sk) : Verify(pk, m^*, \sigma^*) = 1 \land m^* \notin Q^{Sign}] \leq \epsilon(\kappa),

where \mathcal{A} has access to an oracle \mathcal{O}^{Sign} that allows to execute the Sign algorithm and the environment keeps track of all message queried to \mathcal{O}^{Sign} via Q^{Sign}.

Public Key Encryption. We also require public key encryption, which we recall below.

Definition 5. A public key encryption scheme PKE is a triple (KeyGen, Enc, Dec) of PPT algorithms, which are defined as follows:

KeyGen(1^\kappa) : This algorithm takes a security parameter \kappa as input and outputs a keypair (sk, pk). We assume that the message space \mathcal{M} is implicit defined by pk.

Enc(pk, m) : This algorithm takes a public key pk and a message m \in \mathcal{M} as input and outputs a ciphertext c or \perp.

Dec(sk, c) : This algorithm takes a secret key sk and a ciphertext c as input and outputs a message m \in \mathcal{M} or \perp.

We require a PKE to be correct and IND-CCA2 secure, which are formally defined as follows.

Definition 6 (Correctness). A PKE scheme is correct if it holds for all \kappa, for all (sk, pk) \leftarrow KeyGen(1^\kappa), and for all messages m \in \mathcal{M} that Pr[Dec(sk, Enc(pk, m)) = m] = 1.

Definition 7 (IND-CCA2 Security). A PKE scheme is IND-CCA2 secure, if for all PPT adversaries \mathcal{A} there exists a negligible function \epsilon(\cdot) such that

\Pr[(sk, pk) \leftarrow KeyGen(1^\kappa), (m_0, m_1, st) \leftarrow \mathcal{A}^{Enc}(sk), b \leftarrow \{0, 1\}, c \leftarrow Enc(pk, m_b), b^* \leftarrow \mathcal{A}^{Dec}(sk)(c, st) : b = b^* \land c \notin Q^{Dec}] \leq 1/2 + \epsilon(\kappa),

where Q^{Dec} denotes the list of queries to the decryption oracle.
Non-Interactive Zero-Knowledge Proof Systems. Now, we recall a standard definition of non-interactive zero-knowledge proof systems (NIZK). Therefore, let $L_R$ be an NP-language with witness relation $R : L_R = \{ x \mid \exists w : R(x, w) = 1 \}$.

**Definition 8 (Non-Interactive Zero-Knowledge Proof System).** A NIZK is a tuple of algorithms $(\text{Setup}, \text{Proof}, \text{Verify})$, which are defined as follows:

- **Setup($1^\kappa$)**: This PPT algorithm takes a security parameter $\kappa$ as input, and outputs a common reference string $\text{crs}$.
- **Proof($\text{crs}, x, w$)**: This algorithm takes a common reference string $\text{crs}$, a statement $x$, and a witness $w$ as input, and outputs a proof $\pi$.
- **Verify($\text{crs}, x, \pi$)**: This PPT algorithm takes a common reference string $\text{crs}$, a statement $x$, and a proof $\pi$ as input, and outputs a bit $b \in \{0, 1\}$.

If, in addition, Proof runs in polynomial time we talk about a non-interactive argument system. A NIZK is required to be complete, sound, and adaptively zero-knowledge. Subsequently, we recall formal definition of those properties (adapted from [BGI14]).

**Definition 9 (Completeness).** A NIZK is complete, if for every adversary $A$ it holds that

$$\Pr\left[ \text{crs} \leftarrow \text{Setup}(1^\kappa), (x, w) \leftarrow A(\text{crs}), \pi \leftarrow \text{Proof}(\text{crs}, x, w) : \text{Verify}(\text{crs}, x, \pi) = 1 \land (x, w) \in R \right] = 1. \right.$$

**Definition 10 (Soundness).** A NIZK is sound, if for every PPT adversary $A$ there is a negligible function $\epsilon(\cdot)$ such that

$$\Pr\left[ \text{crs} \leftarrow \text{Setup}(1^\kappa), (x, \pi) \leftarrow A(\text{crs}) : \text{Verify}(\text{crs}, x, \pi) = 1 \land x \notin L_R \right] \leq \epsilon(\kappa).$$

A NIZK is called perfectly sound, if $\epsilon = 0$.

**Definition 11 (Adaptive Zero-Knowledge).** A NIZK is adaptively zero-knowledge, if there exists a PPT simulator $S = (S_1, S_2)$ such that for every PPT adversary $A$ there is a negligible function $\epsilon(\cdot)$ such that

$$\left| \Pr\left[ \text{crs} \leftarrow \text{Setup}(1^\kappa) : A^P(\text{crs} \cdot \cdot \cdot)(\text{crs}) = 1 \right] - \Pr\left[ (\text{crs}, \tau) \leftarrow S_1(1^\kappa) : A^{S(\text{crs} \cdot \cdot \cdot)}(\text{crs}) = 1 \right] \right| \leq \epsilon(\kappa),$$

where $\tau$ denotes a simulation trapdoor. Thereby, $P$ and $S$ return $\bot$ if $(x, w) \notin R$ or $\pi \leftarrow \text{Proof}(\text{crs}, x, w)$ and $\pi \leftarrow S_2(\text{crs}, \tau, x)$, respectively, otherwise.

A NIZK is called is perfect adaptively zero-knowledge, if $\epsilon = 0$. Furthermore, we require our NIZK to be a proofs of knowledge, which are defined as follows (adapted from [BGI14]):

**Definition 12 (Proof of Knowledge).** A NIZK is called a proof of knowledge if there exists a PPT extractor $E = (E_1, E_2)$ such that for every adversary $A$ there is a negligible function $\epsilon_1(\cdot)$ such that

$$\left| \Pr[\text{crs} \leftarrow \text{Setup}(1^\kappa) : 1 \leftarrow A(\text{crs})] - \Pr[(\text{crs}, \tau) \leftarrow E_1(1^\kappa) : 1 \leftarrow A(\text{crs})] \right| \leq \epsilon_1(\kappa),$$

and for every PPT adversary $A$ there is a negligible function $\epsilon_2(\cdot)$ such that

$$\Pr\left[ (\text{crs}, \tau) \leftarrow E_1(1^\kappa), (x, \pi) \leftarrow A(\text{crs}), w \leftarrow E_2(\text{crs}, \tau, x, \pi) : \text{Verify}(\text{crs}, x, \pi) = 1 \land (x, w) \notin R \right] \leq \epsilon_2(\kappa).$$
Signatures of Knowledge. Signatures of knowledge (SoKs) as formalized in [CL06] are recalled below, where $L_R$ is as above. A signature of knowledge (SoK) for $L_R$ is defined as follows.

**Definition 13.** A SoK is a tuple of PPT algorithms $(\text{Setup}, \text{Sign}, \text{Verify})$, which are defined as follows:

- $\text{Setup}(1^\kappa)$: This algorithm takes a security parameter $\kappa$ as input and outputs a common reference string $\text{crs}$. We assume that the message space $\mathcal{M}$ is implicitly defined by $\text{crs}$.
- $\text{Sign}(\text{crs}, x, w, m)$: This algorithm takes a common reference string $\text{crs}$, a word $x$, a witness $w$, and a message $m$ as input and outputs a signature $\sigma$.
- $\text{Verify}(\text{crs}, x, m, \sigma)$: This algorithm takes a common reference string $\text{crs}$, a word $x$, a message $m$, and a signature $\sigma$ as input and outputs a bit $b \in \{0, 1\}$.

**Definition 14 (Correctness).** A SoK with respect to $L_R$ is correct, if there exists a negligible function $\epsilon(\cdot)$ such that for all $x \in L_R$, for all $w$ such that $(x, w) \in R$, and for all $m \in \mathcal{M}$ it holds that

$$\Pr[\text{crs} \leftarrow \text{Setup}(1^\kappa), \sigma \leftarrow \text{Sign}(\text{crs}, x, w, m) : \text{Verify}(\text{crs}, x, m, \sigma) = 1] \geq 1 - \epsilon(\kappa).$$

**Definition 15 (Simulatability).** A SoK with respect to $L_R$ is simulatable, if there exists a simulator $\mathcal{S} = (\text{SimSetup}, \text{SimSign})$ such that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that for all polynomials $f$, for all $\kappa$, for all auxiliary inputs $\text{aux} \in \{0, 1\}^{f(\kappa)}$ it holds that

$$\Pr[(\text{crs}), b \leftarrow \mathcal{A}^{\text{Sign}(\text{crs}, \ldots)}(\text{crs}, \text{aux}) : b = 1] - \Pr[(\text{crs}, \tau) \leftarrow \text{SimSetup}(1^\kappa), b \leftarrow \mathcal{A}^{\text{Sim}(\text{crs}, \tau, \cdot, \cdot, \cdot)}(\text{crs}, \text{aux}) : b = 1] \leq \epsilon(\kappa),$$

where $\text{Sim}(\text{crs}, \tau, x, w, m) := \text{SimSign}(\text{crs}, \tau, x, m)$ and $\text{Sim}$ only responds if $(x, w) \in R$.

**Definition 16 (Extraction).** A SoK with respect to $L_R$ is extractable, if in addition to $\mathcal{S}$ there exists an extractor $\text{Extract}$, such that for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\epsilon(\cdot)$ such that for all polynomials $f$, for all $\kappa$, for all auxiliary inputs $\text{aux} \in \{0, 1\}^{f(\kappa)}$ it holds that

$$\Pr[(\text{crs}, \tau) \leftarrow \text{SimSetup}(1^\kappa), (x, m, \sigma) \leftarrow \mathcal{A}^{\text{Sim}(\text{crs}, \tau, \cdot, \cdot, \cdot)}(\text{crs}, \text{aux}), \text{Verify}(\text{crs}, x, m, \sigma) = 0 \lor \text{Extract}(\text{crs}, \tau, x, m, \sigma), (x, m, w) \in Q^{\text{Sim}} \lor (x, w) \in R] \geq 1 - \epsilon(\kappa),$$

where $Q^{\text{Sim}}$ denotes the queries to the $\text{Sim}$ oracle (defined as in Definition 15).

**Structure Preserving Signatures on Equivalence Classes.** Subsequently, we briefly recall structure-preserving signatures on equivalence classes (SPS-EQ) as presented in [HS14, FHS14]. Therefore, let $p$ be a prime and $\ell > 1$; then $\mathbb{Z}_p^\ell$ is a vector space and one can define a projective equivalence relation on it, which propagates to $\mathbb{G}_1^\ell$ and partitions $\mathbb{G}_1^\ell$ into equivalence classes. Let $\sim_R$ be this relation, i.e., for $M, N \in \mathbb{G}_1^\ell : M \sim_R N \iff \exists s \in \mathbb{Z}_p^\ell : M = sN$. An SPS-EQ scheme now signs an equivalence class $[M]_R$ for $M \in (\mathbb{G}_1^\ell)^\ell$ by signing a representative $M$ of $[M]_R$. Let us recall the formal definition of an SPS-EQ scheme subsequently.

**Definition 17.** An SPS-EQ on $\mathbb{G}_1^\ell$ (for $i \in \{1, 2\}$) consists of the following PPT algorithms:

- $\text{BGGen}_R(1^\kappa)$: A bilinear-group generation algorithm, which on input of a security parameter $\kappa$ outputs an asymmetric bilinear group $\text{BG}$.
KeyGen\(_R\) (BG, \(\ell\)): An algorithm, which on input of an asymmetric bilinear group BG and a vector length \(\ell > 1\) outputs a key pair \((sk, pk)\).

Sign\(_R\) (M, sk): An algorithm, which given a representative \(M \in (G^*_1)^\ell\) and a secret key sk outputs a signature \(\sigma\) for the equivalence class \([M]_R\).

ChgRep\(_R\) (M, \(\sigma, \rho, pk\)): An algorithm, which on input of a representative \(M \in (G^*_1)^\ell\) of class \([M]_R\), a signature \(\sigma\) for \(M\), a scalar \(\rho\) and a public key pk returns an updated message-signature pair \((M', \sigma')\), where \(M' = \rho \cdot M\) is the new representative and \(\sigma'\) its updated signature.

Verify\(_R\) (M, \(\sigma, pk\)): An algorithm, which on input of a representative \(M \in (G^*_1)^\ell\), a signature \(\sigma\) and a public key pk outputs a bit \(b \in \{0, 1\}\).

VKey\(_R\) (sk, pk) is an algorithm, which given a secret key sk and a public key pk outputs a bit \(b \in \{0, 1\}\).

For security, we require the following properties.

**Definition 18 (Correctness).** An SPS-EQ scheme on \((G^*_1)^\ell\) is called correct if for all security parameters \(\kappa \in \mathbb{N}\), \(\ell > 1\), \(BG \leftarrow BGGen_R(1^\kappa)\), \((sk, pk) \leftarrow KeyGen_B(BG, \ell)\), \(M \in (G^*_1)^\ell\) and \(\rho \in \mathbb{Z}_p\): \(\Pr[\text{Verify}_R(sk, pk) = 1] = 1\) and \(\Pr[\text{Verify}_R(ChgRep_R(M, Sign_R(M, sk), \rho, pk)) = 1] = 1\).

For EUF-CMA security, outputting a valid message-signature pair, corresponding to an unqueried equivalence class, is considered to be a forgery.

**Definition 19 (EUF-CMA).** An SPS-EQ over \((G^*_1)^\ell\) is existentially unforgeable under adaptively chosen-message attacks, if for all PPT adversaries \(A\) with access to a signing oracle \(O\), there is a negligible function \(\epsilon(\cdot)\) such that:

\[
\Pr\left[\begin{array}{c}
BG \leftarrow BGGen_R(1^\kappa), \\
(sk, pk) \leftarrow KeyGen_R(BG, \ell), \\
(M^*, \sigma^*) \leftarrow A^{O,<sk>(pk)}
\end{array}\right] = \epsilon(\kappa),
\]

where \(Q\) is the set of queries that \(A\) has issued to the signing oracle \(O\).

Besides EUF-CMA security, an additional security property for SPS-EQ was introduced in [FHS15].

**Definition 20 (Perfect Adaption of Signatures).** An SPS-EQ scheme \((BGGen_R, KeyGen_R, Sign_R, ChgRep_R, Verify_R, VKey_R)\) on \((G^*_1)^\ell\) perfectly adapts signatures if for all tuples \((sk, pk, M, \sigma, \rho)\) where it holds that \(VKey_R(sk, pk) = 1\), \(Verify_R(M, \sigma, pk) = 1\), \(M \in (G^*_1)^\ell\), and \(\rho \in \mathbb{Z}_p\), the distributions \((\rho M, Sign_R(\rho M, sk))\) and \(ChgRep_R(M, \sigma, \rho, pk)\) are identical.

An instantiation providing all above security properties is provided in [FHS14, FHS15].

### 3 Dynamic Group Signatures

Subsequently, we recall the established model for dynamic group signatures by Bellare et al. [BSZ05] (BSZ model), where we slightly adapt the notation to ours. We assume that each algorithm outputs a special symbol \(\perp\) on error.

GKeyGen\((1^\kappa)\): This algorithm takes a security parameter \(\kappa\) as input and outputs a triple \((gpk, ik, ok)\) containing the group public key gpk, the issuing key ik as well as the opening key ok.
oracles. Furthermore, the environment maintains the sets HU and GS.

In the following we recall the definitions of the oracles required by the security model. We introduce the user public keys, user private keys, and group signing keys. The list HU of honest users and GS of group users are initially empty and all list entries are implicitly available to the oracle. Finally, SI represents a variable that ensures the consistency of subsequent calls to CrptU and SndToI. All sets are initially empty and all list entries are initially set to ⊥. In the context of lists, we use upk, usk, etc. as shorthand for upk[i], usk[i], etc.

3.1 Oracles

In the following we recall the definitions of the oracles required by the security model. We assume that the keys (gpk, ik, ok) created in the experiments are implicitly available to the oracles. Furthermore, the environment maintains the sets HU, CU of honest and corrupted users, the set GS of message-signature tuples returned by the challenge oracle, the lists upk, usk, gsk of user public keys, user private keys, and group signing keys. The list upk is publicly readable and the environment also maintains the registration table reg. Finally, SI represents a variable that ensures the consistency of subsequent calls to CrptU and SndToI. All sets are initially empty and all list entries are initially set to ⊥. In the context of lists, we use upk, usk, etc. as shorthand for upk[i], usk[i], etc.

AddU(i) : This oracle takes an index i as input. If i ∈ CU ∪ HU it returns ⊥. Otherwise it runs

\( \text{RegGen}(1^n) \) and

\[ (\text{reg}, \text{gsk}_i) \leftarrow (\text{Issue}(gpk, ik, i, upk_i, \text{reg}) \leftrightarrow \text{Join}(gpk, usk_i, upk_i)) \].

Finally, it sets HU ← HU ∪ {i} and returns upk_i.

CrptU(i, upk_j) : This oracle takes an index i and user public key upk_j as input. If i ∈ CU ∪ HU it returns ⊥. Otherwise it sets CU ← CU ∪ {i}, SI ← i and upk_i ← upk_j.

SndToI(i) : This oracle takes an index i as input. If i ≠ SI it returns ⊥. Otherwise, it plays the role of an honest issuer when interacting with the corrupted user i. More precisely, it runs

\[ \text{reg} \leftarrow \text{Issue}(gpk, ik, i, upk_i, \text{reg}), \]

thereby interacting with the dishonest user who aims to join the group but does not necessarily follow the Join protocol.

SndToU(i) : This oracle takes an index i as input. If i ≠ HU it sets HU ← HU ∪ {i}, runs

\( \text{RegGen}(1^n) \). Then it plays the role of the honest user i when interacting with a corrupted issuer. More precisely, it runs

\[ \text{gsk}_i \leftarrow \text{Join}(gpk, usk_i, upk_i), \]
thereby interacting with the dishonest issuer who does not necessarily follow the Issue
protocol.

USK(i) : This oracle takes an index i as input and returns (gsk_i, usk_i).
RReg(i) : This oracle takes an index i as input and returns reg_i.
WReg(i, ρ) : This oracle takes an index i and a registration table entry ρ as input and sets the
entry for i in reg to ρ.

GSig(i, m) : This oracle takes an index i and a message m as input. If i ∈ HU or gsk_i = ⊥ it
returns ⊥ and σ ← Sign(gpk, gsk_i, m) otherwise.

Ch(b, i_0, i_1, m) : This algorithm takes a bit b, two indexes i_0 and i_1, and a message m as input.
If {i_0, i_1} ⊆ HU ∨ gsk_i_0 = ⊥ ∨ gsk_i_1 = ⊥ it returns ⊥. Otherwise, it computes σ ←
Sign(gpk, gsk_i_0, m), sets GS ← GS ∪ {(m, σ)} and returns σ.

Open(m, σ) : This oracle takes a message m and a signature σ as input. If (m, σ) ∈ GS or
Verify(gpk, m, σ) = 0 it returns ⊥. Otherwise, it returns (i, τ) ← Open(gpk, ok, reg, m, σ).

3.2 Security Notions

For security, dynamic group signatures are required to be correct, anonymous, traceable, and
non-frameable. We recall the formal definitions below.

Definition 21 (Correctness). A GSS is correct, if for all adversaries A it holds that
\[ \Pr \left[ (gpk, ik, ok) \leftarrow \text{GKeyGen}^\kappa, \right. \right.
\left. \begin{array}{c}
\sigma \leftarrow \text{Sign}(gpk, gsk_i, m),
\text{Verify}(gpk, m, \sigma) = 1 \wedge
\end{array} \right] \]

Definition 22 (Anonymity). A GSS is anonymous, if for all PPT adversaries A there is a
negligible function ε(·) such that
\[ \Pr \left[ (gpk, ik, ok) \leftarrow \text{GKeyGen}^\kappa, b \in \{0, 1\}, \right. \right.
\left. \begin{array}{c}
\sigma \leftarrow \text{Sign}(gpk, gsk_i, m),
\text{Judge}(gpk, m, \sigma, i, \text{upk}_i, \tau) = 1
\end{array} \right] \leq 1/2 + \epsilon(\kappa). \]

Definition 23 (Traceability). A GSS is traceable, if for all PPT adversaries A there is a
negligible function ε(·) such that
\[ \Pr \left[ (gpk, ik, ok) \leftarrow \text{GKeyGen}^\kappa, \right. \right.
\left. \begin{array}{c}
\sigma \leftarrow \text{Out}(gpk, ok),
\text{Judge}(gpk, m, \sigma, i, \text{upk}_i, \tau) = 0
\end{array} \right] \leq \epsilon(\kappa). \]

Definition 24 (Non-Frameability). A GSS is non-frameable, if for all PPT adversaries A there is a
negligible function ε(·) such that
\[ \Pr \left[ (gpk, ik, ok) \leftarrow \text{GKeyGen}^\kappa, \right. \right.
\left. \begin{array}{c}
\sigma \leftarrow \text{Sign}(gpk, gsk_i, m),
\text{Judge}(gpk, m, \sigma, i, \text{upk}_i, \tau) = 1
\end{array} \right] \leq \epsilon(\kappa), \]

where USK and SIG denote the queries to the oracles USK and Sign, respectively.
4 Construction

Before we present our construction in Scheme 1, we briefly revisit our basic idea. In our scheme, each group member chooses a secret key, which consists of a vector \((R, P) \in (G_1)^2\). When joining the group, a blinded version \(q \cdot (R, P)\) with \(q \in \mathbb{Z}_p^\ast\) of this key is signed by the group manager using an SPS-EQ, and, by the re-randomization property of SPS-EQ, the user thus obtains a signature on the unblinded key (using ChgRep with \(q^{-1}\)). Using this key, each group member can sign a message on behalf of a group by randomizing it’s secret key and the corresponding SPS-EQ signature and computing a signature of knowledge of the used randomizer. Very roughly, a signer then remains anonymous since it is infeasible to distinguish two randomized user secret keys under DDH in \(G_1\). The unforgeability of SPS-EQ ensures that each valid signature can be opened. Furthermore, it is hard to forge signatures of honest group members since it is hard to unblind a user secret key under co-CDHI and the signature of knowledge essentially ensures that we can extract such an unblinded user secret key from a successful adversary. To provide a means to open signatures, a user has to provide an encryption of a value \(R \in G_2\) such that \(e(R, P) = e(P, R)\) on joining.

\[
\begin{align*}
\text{GKeyGen}(1^n) & : \text{Run } \text{BG} \leftarrow \text{BGGen}(1^n), (\text{sk}_G, \text{pk}_G) \leftarrow \text{KeyGen}_G(\text{BG}, 2), (\text{sk}_0, \text{pk}_0) \leftarrow \text{PKE.KeyGen}(1^n), \\
\text{crs}_G & \leftarrow \text{NIZK.Setup}(1^n), \text{crs}_R \leftarrow \text{NIZK.Setup}(1^n), \text{crs}_G \leftarrow \text{SoK.Setup}(1^n), \text{choose } T \leftarrow \mathbb{G}_1, \text{set } \text{gpk} \leftarrow (\text{pk}_R, \text{pk}_0, \text{crs}_G, \text{crs}_R, T), \text{ik} \leftarrow \text{sk}_R, \text{ok} \leftarrow \text{sk}_0 \text{ and return } (\text{gpk}, \text{ik}, \text{ok}) \\
\text{UKeyGen}(1^n) & : \text{Return } (\text{usk}_i, \text{upk}_i) \leftarrow \text{DSS.KeyGen}(1^n).
\end{align*}
\]

\[
\begin{align*}
\text{Join}(1^n)(\text{gpk}, \text{uski}, \text{upki}) & : \text{Choose } q, r \leftarrow \mathbb{Z}_p^\ast, \text{set } (U, Q) \leftarrow (r \cdot q, P \cdot q), \text{ and compute } \hat{C}_i \leftarrow \text{PKE.Enc}(\text{pk}_0, r \cdot \hat{P}; \omega), \sigma_i \leftarrow \text{DSS.Sign}(\text{uski}, \hat{C}_i), \pi_i \leftarrow \text{NIZK.Proof}(\text{crs}_G, (U, Q, \hat{C}_i), \text{pk}_O), (r, q, \omega, \bot), \text{ where } \pi_i \text{ attests membership of } (U, Q, \hat{C}_i, \text{pk}_0) \text{ in the NP relation } R_i: \\
(\text{upk}_i, \text{upki}) \rightarrow (\text{gpk}, \text{uski}, \text{upki}) \leftarrow \text{DSS.KeyGen}(1^n).
\end{align*}
\]

\[
\begin{align*}
\text{Issue}(\text{gpk}, \text{ik}, \text{upki}, \text{reg}) & : \text{Receive } M_i \leftarrow ((U, Q), \hat{C}_i, \sigma_i, \pi_i) \text{ and } \text{st} \leftarrow (\text{gpk}, \text{uski}, \text{upki}, q, U, Q). \\
\text{Sign}(\text{gpk}, \text{gsk}, m) & : \text{Choose } \rho \leftarrow \mathbb{Z}_p^\ast, \text{compute } \sigma_1 \leftarrow \text{ChgRep}_G(\text{gsk}, \rho, \text{pk}_G), \sigma_2 \leftarrow \text{SoK.Sign}(\text{crs}_G, \sigma_1[1][2], \rho, m) \text{ and return } \sigma \leftarrow (\sigma_1, \sigma_2). \text{Here SoK is with respect to the language associated to the NP relation } R_5: (\sigma_1[1][2], \rho) \in R_5 \iff \sigma_1[1][2] = \rho \cdot P. \\
\text{Verify}(\text{gpk}, \text{gsk}, m, \sigma) & : \text{Return } \text{Verify}_R(\sigma_1, \text{pk}_R) \land \text{SoK.Verify}(\text{crs}_G, \sigma_1[1][2], m, \sigma_2).
\end{align*}
\]

\[
\begin{align*}
\text{Open}(\text{gpk}, \text{ok}, \text{reg}, m, \sigma) & : \text{Obtain } (\hat{C}_i, \sigma_i) \text{ and } \text{rP} \leftarrow \text{PKE.DEC}(\text{sk}_0, \hat{C}_i) \text{ from reg such that } e(\sigma_1[1][1], P) = e(\sigma_1[1][2], \text{rP}). \text{Compute } \tau_0 \leftarrow \text{NIZK.Proof}(\text{crs}_G, (\hat{C}_i, \sigma_i, \text{pk}_0, \text{upk}_i), (\text{sk}_0, \text{rP}, \bot)), \text{ where } \tau_0 \text{ attests membership in the NP relation } R_0: \\
((\hat{C}_i, \sigma_i, \text{pk}_0, \text{upk}_i), (\text{sk}_0, \text{rP}, t)) \in R_0 \iff (\text{rP} = \text{PKE.DEC}(\text{sk}_0, \hat{C}_i)) \land \text{pk}_0 \equiv \text{sk}_0 \land \\
\text{DSS.Verify}(\text{upk}_i, \hat{C}_i, \sigma_i) = 1 \land e(\sigma_1[1][1], \text{rP}) = e(\sigma_1[1][2], \text{rP})) \lor T = t \cdot P.
\end{align*}
\]

\[
\begin{align*}
\text{Judge}(\text{gpk}, m, \sigma, i, \text{upki}, \tau) & : \text{Given } \tau = (\tau_0, \hat{C}_i, \sigma_i), \text{ return whatever } \text{NIZK.Verify}(\text{crs}_G, (\hat{C}_i, \sigma_i, \text{pk}_0, \text{upki}), \tau_0) \text{ returns.}
\end{align*}
\]

Scheme 1: Fully Secure Dynamic Group Signature Scheme
Correctness of Scheme 1 is straightforward and can be verified by inspection. The remaining properties are proven subsequently.

**Theorem 1.** If \( \text{NIZK} \) is adaptively zero knowledge, \( \text{SoK} \) is simulatable and extractable, \( \text{PKE} \) is IND-CCA2 secure, SPS-EQ perfectly adapts signatures, and the DDH assumption holds in \( \mathbb{G}_1 \), then Scheme 1 is anonymous.

**Proof (Anonymity).** We prove anonymity by showing that the output distributions of the Ch oracle are (computationally) independent of the bit \( b \). To do so, we assume that the adversary makes \( q_{\text{Ch}} \leq \text{poly}(\kappa) \) queries to \( \text{Ch} \), \( q_0 \leq \text{poly}(\kappa) \) queries to \( \text{Open} \), and \( q_{\text{SndToU}} \leq \text{poly}(\kappa) \) queries to \( \text{SndToU} \).

**Game 0:** The original anonymity game.

**Game 1:** As Game 0, but we run \( (\text{crs}_0, \tau_0) \leftarrow \text{NIZK.Setup}(1^\kappa) \) upon running \( \text{GKeyGen} \) and store the trapdoor \( \tau_0 \). Then, we simulate all calls to \( \text{NIZK.Proof} \) executed in Join using the simulator, i.e., without a witness.

**Transition - Game 0 \( \rightarrow \) Game 1:** A distinguisher between Game 0 and Game 1 is an adversary against adaptive zero knowledge of \( \text{NIZK} \), and, therefore, the probability to distinguish Game 0 and Game 1 is negligible, i.e., \( |\Pr[S_1] - \Pr[S_0]| \leq \epsilon_{\text{NIZK}}(\kappa) \).

**Game 2:** As Game 1, but we run \( (\text{crs}_0, \tau_0) \leftarrow \text{NIZK.Setup}(1^\kappa) \) upon running \( \text{GKeyGen} \) and store the trapdoor \( \tau_0 \). Then, we simulate all calls to \( \text{NIZK.Proof} \) in \( \text{Open} \) using the simulator, i.e., without a witness.

**Transition - Game 1 \( \rightarrow \) Game 2:** A distinguisher between Game 0 and Game 1 is an adversary against adaptive zero knowledge of \( \text{NIZK} \), and, therefore, the probability to distinguish Game 1 and Game 2 is negligible, i.e., \( |\Pr[S_2] - \Pr[S_1]| \leq \epsilon_{\text{NIZK}}(\kappa) \).

**Game 3:** As Game 2, but we run \( (\text{crs}_s, \tau_s) \leftarrow \text{SoK.SimSetup}(1^\kappa) \) instead of \( \text{crs}_s \leftarrow \text{NIZK.Setup}(1^\kappa) \) upon running \( \text{GKeyGen} \) and store the trapdoor \( \tau_s \). Then, we simulate all calls to \( \text{SoK.Sign} \) using the simulator, i.e., without a witness.

**Transition - Game 2 \( \rightarrow \) Game 3:** A distinguisher between Game 2 and Game 3 is an adversary against simulatability of \( \text{SoK} \). Therefore, the distinguishing probability is negligible, i.e., \( |\Pr[S_3] - \Pr[S_2]| \leq \epsilon_{\text{Sim}}(\kappa) \).

**Game 4:** As Game 3, but instead of computing \( (\text{sk}_0, \text{pk}_0) \leftarrow \text{PKE.KeyGen}(1^\kappa) \) in \( \text{GKeyGen} \), we obtain \( \text{pk}_0 \) from an IND-CCA2 challenger and set \( \text{sk}_0 \leftarrow \emptyset \). The environment additionally maintains a secret list \( \text{GSK} \) and upon each call to the \( \text{SndToU} \) oracle it sets \( \text{GSK}[i] \leftarrow \text{gsk}_i \). Furthermore, we simulate the open algorithm as follows.

\[ \text{Open}(\text{gpk}, \text{ok}, \text{reg}, m, \sigma) : \text{Obtain } \rho \leftarrow \text{Extract}(\text{crs}_s, \tau_s, \sigma_1[1][2], m, \sigma_2), \text{gsk}_i^s \leftarrow \text{ChgRep}_{\text{GSK}}(\sigma_1, \rho^{-1}, \text{pk}_\text{R}), \text{and index } i \text{ such that } \text{GSK}[i] = \text{gsk}_i^s. \text{ If } i \neq \bot \text{ and the entry for } i \text{ in the registration table was not overwritten by the adversary, compute a (simulated) proof } \tau \text{ and return } (i, \tau). \text{ Otherwise we submit } \hat{C}_1 \text{ to the decryption oracle provided by the IND-CCA2 challenger and return whatever the original open oracle would return.} \]

If the extractor fails at some point, we choose \( b \in \{0, 1\} \) and return \( b \).

**Transition - Game 3 \( \rightarrow \) Game 4:** By the extractability of the \( \text{SoK} \), one can extract a witness \( \rho \) in every call to \( \text{Open} \) with overwhelming probability \( (1 - \epsilon_{\text{Ext}}(\kappa))^\infty \). Thus, we have that \( \Pr[S_4] = 1/2 + (\Pr[S_3] - 1/2) \cdot (1 - \epsilon_{\text{Ext}}(\kappa))^\infty \).

**Game 5:** As Game 4, but we compute the ciphertext \( \hat{C}_1 \) in the Join algorithm (executed within the \( \text{SndToU} \) oracle) as \( \hat{C}_1 \leftarrow \text{PKE.Enc}(\text{pk}, \hat{P}) \), i.e., with a constant message that is independent of the user.

**Transition - Game 4 \( \rightarrow \) Game 5:** A distinguisher between Game 4 and Game 5 is a distinguisher for the IND-CCA2 game of the \( \text{PKE} \), i.e., \( |\Pr[S_5] - \Pr[S_4]| \leq q_{\text{SndToU}} \cdot \epsilon_{\text{CCA2}}(\kappa) \).

\(^1\) For compactness, we collapsed the \( q_{\text{SndToU}} \) game changes into a single game change and note that one can straightforwardly unroll this to \( q_{\text{SndToU}} \) game changes where a single ciphertext is exchanged in each game.
Game 6: As Game 5, but the environment obtains and stores a DDH instance \((aP, bP, cP)\) in \(G_1\). Additionally, we further modify the Join algorithm (executed within \(\text{SndToU}\)) as follows. Instead of choosing \(r \in \mathbb{Z}_p^\star\), we use the random self reducibility of DDH to obtain an independent DDH instance \((uP, vP, wP)\), choose \(q \in \mathbb{Z}_p^\star\), compute \((u_i, Q_i) \leftarrow (q \cdot uP, q \cdot P)\). Furthermore, the environment maintains a secret list \(\text{DDH}\) and upon each issue it sets \(\text{DDH}[i] \leftarrow (uP, vP, wP)\).

Transition - Game 5 → Game 6: The output distributions in Game 5 and Game 6 are identical, i.e., \(\text{Pr}[S_6] = \text{Pr}[S_5]\).

Game 7: As Game 6, but all calls \(\text{ChgRep}_R(M, \rho, \pk_R)\) are replaced by \(\text{Sign}_R(\rho \cdot M, \sk_R)\).

Transition - Game 6 → Game 7: Under perfect adaption of signatures, the output distributions in Game 6 and Game 7 are identical, i.e., \(\text{Pr}[S_7] = \text{Pr}[S_6]\).

Game 8 \(j\) (\(1 \leq j \leq q_{Ch}\)): As Game 7, but we modify the Ch oracle as follows. For the first \(j\) queries, instead of running \(\sigma_1 \leftarrow \text{Sign}_R(\rho \cdot gsk_i[1], \sk_R)\), we choose \(R \leftarrow G_1\), and compute \(\sigma_1 \leftarrow \text{Sign}_R(\rho \cdot (R, P), \sk_R)\).

Transition - Game 7 → Game 8 \(j\): A distinguisher between Game 7 and Game 8 \(j\) is a DDH distinguisher. To show this, we present an implementation of the Ch oracle, that—depending on the validity of the DDH instance \((aP, bP, cP)\)—interpolates between Game 7 and Game 8 \(j\). That is, in the first query we obtain the tuple \((uP, vP, wP) \leftarrow \text{DDH}[i]_b\) and compute \(\sigma_1\) as \(\sigma_1 \leftarrow \text{Sign}_R((uP, vP), \sk_R)\). Then, if the initial DDH instance \((aP, bP, cP)\) is valid, we have a distribution as in Game 7, whereas we have a distribution as in Game 8 \(j\) otherwise. The success probability of a distinguisher between Game 7 and Game 8 \(j\) is thus negligible, i.e., \(\Pr[|S_8] - \Pr[|S_7]| \leq \epsilon_{DDH}(\kappa)\).

Transitions - Game 8 \(j\) → Game 8 \(j+1\) (\(1 \leq j \leq q_{Ch}\)): The answers of the Ch oracle for the first \(j\) queries are already random in Game 8 \(j\). Then, it is easy to show that a distinguisher for Game 8 \(j\) and game 8 \(j+1\) is a DDH distinguisher, i.e., by embedding \((uP, vP, wP) \leftarrow \text{DDH}[i]_b\) in the answer of the Ch query \(j+1\) using the same strategy as above. Summing up, we have \(|\Pr[|S_8] - \Pr[|S_8]| \leq (q_{Ch} - 1) \cdot \epsilon_{DDH}(\kappa)\).

In Game 8q0, the simulation is independent of the bit \(b\), i.e., \(\Pr[|S_8] = 1/2\); what remains is to obtain a bound on the success probability in Game 0. We have that \(\Pr[|S_0] \leq \Pr[|S_3] + \epsilon_{ZK_0}(\kappa) + \epsilon_{ZK_0} + \epsilon_{SIM}(\kappa)\), that \(\Pr[|S_3] \leq 1/2 + \epsilon_{SndToU} \cdot \epsilon_{CCAg}(\kappa) + q_{Ch} \cdot \epsilon_{DDH}(\kappa)\), and that \(\Pr[|S_0] \leq \Pr[|S_3] \cdot 1/2 + \epsilon_{ZK_0}(\kappa) + \epsilon_{ZK_0} + \epsilon_{SIM}(\kappa)\). Combining those equations yields \(\Pr[|S_0] \leq \Pr[|S_3] \cdot 1/2 + 1/2 + \epsilon_{ZK_0}(\kappa) + \epsilon_{ZK_0} + \epsilon_{SIM}(\kappa)\).

Theorem 2. If SPS-EQ is EUF-CMA secure, NIZK is a proof of knowledge, and the DL assumption holds in \(G_1\), then Scheme 1 is traceable.

Proof (Traceability). We show that traceability holds by showing that an efficient adversary \(\mathcal{A}'\) against traceability can efficiently be turned into an efficient adversary

1. \(\mathcal{A}'\) against the discrete logarithm problem in \(G_1\), or
2. \(\mathcal{A}'\) against EUF-CMA security of the underlying SPS-EQ.

We do so by presenting efficient reductions \(\mathcal{R}_{DL}\) and \(\mathcal{R}'\), respectively, that simulate the environment for \(\mathcal{A}'\) and interact with the challengers from respective external security games.

1. Here, \(\mathcal{R}_{DL}\) obtains a DL instance \(tP\) for the group \(G_1\), sets \(T \leftarrow tP\), sets up the NIZK used in Join in extraction mode, i.e., via \((\text{crs}_j, \tau_j) \leftarrow \text{NIZK} \cdot \text{E}_1(\kappa)\) and performs the remaining \(\text{GKeyGen}\) as in the original game. It starts \(\mathcal{A}'\) on \((\text{gpk}, \text{sk})\). At the end of each successful \(\text{SndToI}\), it extracts \((\cdot, \cdot, \cdot, \cdot) \leftarrow \text{E}_2(\text{crs}_j, \tau, (U_i, Q, \hat{C}_i, \text{pk}_0), \tau_i)\). If \(t \cdot P \neq T\) it aborts as we are in the second case, and outputs \(t\) as a solution to the DL problem in \(G_1\) otherwise.
If the adversary $A^r$ eventually outputs a forgery $(m, \sigma)$, but did never engage in such an aforementioned $SndTol$ execution, then $R^{DL}$ aborts because we are in the second case.

2. Here, $R'$ obtains $BG$ and a public key $pk_R$ from the challenger in the EUF-CMA game of the SPS-EQ, sets up the NIZK used in Join in extraction mode, i.e., via $(crs_1, \tau_J) \leftarrow \text{NIZK} \cdot E_1(1^n)$ and performs the remaining $GKeyGen$ as in the original game. It starts $A^r$ on $(gpk, ok)$. Whenever a signature is required, $R'$ forwards the message to be signed to the signing oracle provided by the EUF-CMA challenger. Furthermore, at the end of each successful $SndTol$, it extracts $(r, q, \omega, \cdot) \mapsto E_2(crs_1, \tau, (U_i, Q, \hat{C}_j, pk_0), \pi_1)$. If $C_j \neq \text{PKE}.Enc(pk_o, rP; \omega) \lor U_i \neq r \cdot Q$, it aborts as we are in the first case. Otherwise, if $A^r$ eventually outputs $(m, \sigma)$, we know that no $SndTol$ oracle execution did abort and thus $\sigma$ contains an SPS-EQ signature $\sigma_1$ for some $(rP, P)$ such that $R'$ has never seen a corresponding $rP$, i.e., there is no entry $i$ in the registration table where $\hat{C}_j$ contains $rP$ s.t. $e(\sigma_1[1][1], P) = e(\sigma_1[1][2], rP)$ holds. Consequently, $R'$ outputs $\sigma_1$ as a forgery for the SPS-EQ scheme as a signature on an unqueried equivalence class.

The simulations of both reductions are negligibly close to an original game and the reductions (in particular the extractor) succeed with overwhelming probability. We note that (1) and (2) could be combined into a single reduction, thus circumventing the security loss of $1/2$, but we opted for a separate presentation for the sake of readability.

**Theorem 3.** If NIZK is an adaptively zero knowledge proof of knowledge, SoK is simulatable and extractable, DSS is EUF-CMA secure, PKE is perfectly correct, and the co-CDHI assumption holds, then Scheme 1 is non-frameable.

**Proof (Non-frameability).** We prove non-frameability using a sequence of games. Thereby we assume that the upper bound of users in the system is $q \leq \text{poly}(\kappa)$.

**Game 0:** The original non-frameability game.

**Game 1:** As Game 0, but we guess the index $i$ that will be attacked by the adversary. If the adversary attacks another index, we abort.

**Transition - Game 0 $\rightarrow$ Game 1:** The winning probability in Game 1 is the same as in Game 0, unless an abort event happens. We thus have $\Pr[S_0] = \Pr[S_0] \cdot \frac{1}{q}$.

**Game 2:** As Game 1, but we run $(crs_J, \tau_J) \leftarrow \text{NIZK} \cdot E_1(1^n)$ instead of $\tau_J \leftarrow \text{NIZK}.Setup(1^n)$ upon running $GKeyGen$ and store the trapdoor $\tau_J$. Then, we simulate all calls to $\text{NIZK.Proof}$ in Join using the simulator, i.e., without a witness.

**Transition - Game 1 $\rightarrow$ Game 2:** A distinguisher between Game 1 and Game 2 is an adversary against adaptive zero knowledge of NIZK, and, therefore, the probability to distinguish Game 1 and Game 2 is negligible, i.e., $\Pr[S_0] - \Pr[S_1] \leq \epsilon_{\text{2K}}(\kappa)$.

**Game 3:** As Game 2, but we run $(crs_0, \tau_0) \leftarrow \text{NIZK} \cdot E_1(1^n)$ instead of $\tau_0 \leftarrow \text{NIZK}.Setup(1^n)$ upon running $GKeyGen$ and store the trapdoor $\tau_0$.

**Transition - Game 2 $\rightarrow$ Game 3:** A distinguisher between Game 2 and Game 3 distinguishes an extraction CRS from an honest CRS, and, therefore, the probability to distinguish Game 2 and Game 3 is negligible, i.e., $\Pr[S_0] - \Pr[S_2] \leq \epsilon_{\text{ext}}(\kappa)$.

**Game 4:** As Game 3, but we setup the SoK in simulation mode, i.e., we run $(crs_S, \tau_S) \leftarrow \text{SoK}.SimSetup(1^n)$ instead of $\tau_S \leftarrow \text{SoK}.Setup(1^n)$ upon running $GKeyGen$ and store the trapdoor $\tau_S$. Then, we simulate all calls to $\text{SoK.Sign}$ using the simulator, i.e., without a witness.

**Transition - Game 3 $\rightarrow$ Game 4:** A distinguisher between Game 3 and Game 4 is an adversary against simulatability of SoK. Therefore, the distinguishing probability is negligible, i.e., $\Pr[S_0] - \Pr[S_3] \leq \epsilon_{\text{SIM}}(\kappa)$.

---

2 Note that the perfect correctness of the PKE and extractability of the proof of knowledge ensure that we are always in Case 2 if we do not abort.
Game 5: As Game 4, but we modify the Join algorithm (executed within the SndToU oracle) when queried for user with index $i$ as follows. We obtain a co-CDHI instance $(aP, 1/aP)$ for $BG$, choose $r \leftarrow Z_p$, set $(U_i, Q) \leftarrow (r \cdot P, aP)$, and compute $\hat{C}_j \leftarrow \text{PKE.Enc}(\hat{P}_k, r \cdot 1/aP)$ and store $r$. On successful execution we set $\text{gsk}_i \leftarrow ((U_i, Q), \sigma')$ (note that $\pi_1$, as well as the signatures in the $\text{GSig}$ oracle are already simulated, i.e., the discrete log of $Q$ is not required to be known to the environment).

Transition - Game 4 $\rightarrow$ Game 5: Since $r$ is uniformly random, we can write it as $r = r'a$ for some $r' \in Z_p$. Then it is easy to see that the game change is only conceptual, i.e., \( \Pr[S_5] = \Pr[S_6] \).

Game 6: As Game 5, but for every forgery output by the adversary, we extract $\rho \leftarrow \text{Extract}(\text{crs}_S, S, \sigma_1[1][2], m, \sigma_2)$ and abort if the extraction fails.

Transition - Game 5 $\rightarrow$ Game 6: By the extractability of the SoK, one can extract a witness $\rho$ with overwheleming probability $1 - \epsilon_{\text{ext}}(\kappa)$. Thus, we abort with probability $\epsilon_{\text{ext}}(\kappa)$ and \( \Pr[S_6] = \Pr[S_5] \cdot (1 - \epsilon_{\text{ext}}(\kappa)) \).

Game 7: As Game 6, but we further modify the Join algorithm when queried for user with index $i$ as follows (executed within the SndToU oracle). Instead of choosing $(\text{usk}_i, \text{upk}_i) \leftarrow \text{UKeyGen}(1^\kappa)$, we engage with an EUF-CMA challenger, obtain $\text{upk}_i$ and set $\text{usk}_i \leftarrow \emptyset$. If any signature is required, we obtain it using the oracle provided by the EUF-CMA challenger.

Transition Game 6 $\rightarrow$ Game 7: This change is only conceptual, i.e., \( \Pr[S_7] = \Pr[S_6] \).

Game 8: As Game 7, but we obtain a DL instance $tP$ for the group $G_1$ and set $T \leftarrow tP$.

Transition Game 7 $\rightarrow$ Game 8: This change is only conceptual, i.e., \( \Pr[S_8] = \Pr[S_7] \).

At this point we have three possibilities if $A$ outputs a valid forgery.

1. If a signature for $\hat{C}_j$ was never requested, $A$ is an EUF-CMA forger for the DSS and $(\hat{C}_j, \sigma_1)$. Then $\Pr[S_8] \leq \epsilon_1(\kappa)$.
2. Otherwise, we know that $\hat{C}_j$ is honestly computed by the environment and—by the perfect correctness of $\text{PKE}$—thus contains $r/aP$, which leaves us two possibilities:
   (a) If $e(\sigma_1[1][1], \hat{P}) = e(\sigma_1[1][2], r/aP)$, $A$ is an adversary against co-CDHI, since we can obtain $(((r \cdot 1/aP, P), \sigma')) \leftarrow \text{ChgRep}_{\text{K}}(\sigma_1, \rho^{-1}, \text{pk}_\sigma)$ and use $r$ to output $r^{-1} \cdot (r \cdot 1/aP) = 1/aP$. That is, $\Pr[S_8] \leq \epsilon_{\text{co-CDHI}}(\kappa)$.
   (b) If $e(\sigma_1[1][1], \hat{P}) \neq e(\sigma_1[1][2], r/aP)$, $A$ honestly performed the proof $\pi_\sigma$ with respect to the $T$-part of the OR statement. We can thus obtain $(\cdot, t) \leftarrow \text{NIZK.E2}(\text{crs}_S, \sigma_1, \hat{C}_j, \text{pk}_\sigma, \text{upk}_i, \pi_\sigma)$ and output $t$ as a solution to the DL problem with overwhelming probability $1 - \epsilon_{\text{ext}}(\kappa)$. In this case $\Pr[S_8] \leq \frac{\epsilon_{\text{ext}}(\kappa)}{1 - \epsilon_{\text{ext}}(\kappa)}$.

Independent of whether case (1), (2a), or (2b) holds, the success probability in Game 8 is negligible. Henceforth, we use $\epsilon_{\text{ext}}(\kappa) = \max\{\epsilon_1(\kappa), \epsilon_{\text{co-CDHI}}(\kappa), \frac{\epsilon_{\text{ext}}(\kappa)}{1 - \epsilon_{\text{ext}}(\kappa)}\}$, which yields the following bound for the success probability in Game 1: $\Pr[S_1] \leq \Pr[S_0] + \epsilon_{\text{ZK}}(\kappa) + \epsilon_{\text{ext}}(\kappa) + \epsilon_{\text{SIM}}(\kappa)$.

Furthermore, we know that $\Pr[S_0] = \frac{\epsilon_{\text{ext}}(\kappa)}{1 - \epsilon_{\text{ext}}(\kappa)}$ and $\Pr[S_0] = \Pr[S_1] \cdot q$. Taking all together we have that $\Pr[S_0] \leq q \cdot \left( \frac{\epsilon_{\text{ext}}(\kappa)}{1 - \epsilon_{\text{ext}}(\kappa)} + \epsilon_{\text{ZK}}(\kappa) + \epsilon_{\text{ext}}(\kappa) + \epsilon_{\text{SIM}}(\kappa) \right)$, which is negligible.

4.1 Instantiation in the ROM

To compare our approach to existing approaches regarding signature size and computational effort upon signature generation and verification, we present the sign and verification algorithms for an instantiation of our scheme with the SPS-EQ from [FHS14, FHS15], whose security holds in the generic group model, and signatures of knowledge obtained when applying the Fiat-Shamir [FS87] heuristic to $\Sigma$-protocols. Before we do so, we recall that the group signing key $\text{gsk}_i$ consists of a vector of two group elements $(R, P) \in (G_1)^2$ and an SPS-EQ signature.
\( \sigma \in G_1 \times G_1' \times G_2' \) on this vector. Randomization of a \( gsk_i \) with a random value \( \rho \in \mathbb{Z}_p^* \), i.e., ChgRep\(_R\), requires 4 multiplications in \( G_1 \) and 1 multiplication in \( G_2 \). Verification of an SPS-EQ signature on \( gsk_i \) requires 5 pairings. Subsequently, we show how Sign and Verify are instantiated in this setting, where \( H : \{0, 1\}^* \rightarrow \mathbb{Z}_p \) is modelled as a random oracle:

\[
\text{Sign}(gpk, gsk_i, m) : \text{Given gpk, gsk}_i = ((R, P), \sigma) \text{ and } m, \text{ compute } \rho \in \mathbb{Z}_p^*, \text{ compute } \sigma_1 = ((R', P'), \sigma') \leftarrow \text{ChgRep}_R(gsk_i, \rho, pk_R). \text{ Choose } \nu \in G_p^*, \text{ compute } N \leftarrow \nu P, c \leftarrow H(N|m), z \leftarrow \nu + c \cdot \rho, \text{ and set } \sigma_2 \leftarrow (c, z).
\]

\[
\text{Verify}(gpk, m, \sigma) : \text{Given gpk, m, and } \sigma = (\sigma_1, \sigma_2) = (((R', P'), \sigma), (c, z)), \text{ return 0 if } \text{Verify}_{\text{ps}}(\sigma_1, pk_R) = 0. \text{ Otherwise compute } N \leftarrow cP' - zP \text{ and check whether } c = H(N|m) \text{ holds. If so return 1 and 0 otherwise.}
\]

All in all, group signatures contain 4 elements in \( G_1 \), 1 element in \( G_2 \) and 2 elements in \( \mathbb{Z}_p \). Counting only the expensive operations, signing costs 5 multiplications in \( G_1 \) and 1 multiplication in \( G_2 \), and verification costs 2 multiplications in \( G_1 \) and 5 pairings.

We note that the proofs performed using the NIZK within Join and Open can straightforwardly be instantiated using standard techniques. Therefore, and since they are neither required within Sign nor Verify, we do not discuss their instantiation here.

## 5 Evaluation and Discussion

To underline the practical efficiency of our approach, we provide a comparison of our approach when instantiated in the ROM with other related schemes in the ROM. In particular we use two schemes who follow the approach of Bichsel et al., i.e., [BCN\(^+\)10, PS16], which are proven secure in a weaker model, and the well known BBS scheme [BBS04] (with and without precomputations) providing the weaker anonymity notion of CPA-ful anonymity. We note that we use the plain BBS scheme for comparison, which does not even provide non-frameability. The non-frameable version would be even more expensive. Moreover, we use the group signature scheme with the shortest known signatures [DP06] (with and without precomputations) being secure in the strong BSZ model and thus providing CCA2-full anonymity. In Table 1, we provide a comparison regarding signature size and computational costs. In Table 2 we provide a comparison of the estimated efficiency in a 254bit BN-pairing setting, based on performance.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Anon.</th>
<th>Signature Size</th>
<th>Signature Cost</th>
<th>Verification Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>[BCN(^+)10]</td>
<td>BCN(^+)</td>
<td>(3G_1 + 2G_p)</td>
<td>(1G_T + 3G_1)</td>
<td>(5P + 1G_T + 1G_1)</td>
</tr>
<tr>
<td>[PS16]</td>
<td>BCN(^+)</td>
<td>(2G_1 + 2G_p)</td>
<td>(1G_T + 2G_1)</td>
<td>(3P + 1G_T + 1G_1)</td>
</tr>
<tr>
<td>[BBS04]</td>
<td>CPA</td>
<td>(3G_1 + 6G_p)</td>
<td>(3P + 3G_T + 9G_1)</td>
<td>(5P + 4G_T + 8G_1)</td>
</tr>
<tr>
<td>[BBS04] (prec.)</td>
<td>CPA</td>
<td>(3G_1 + 6G_p)</td>
<td>(3G_T + 9G_1)</td>
<td>(4G_T + 8G_1)</td>
</tr>
<tr>
<td>This paper</td>
<td>CCA2</td>
<td>(1G_2 + 4G_1 + 2G_p)</td>
<td>(1G_2 + 5G_1)</td>
<td>(5P + 2G_1)</td>
</tr>
<tr>
<td>[DP06]</td>
<td>CCA2</td>
<td>(4G_1 + 5G_p)</td>
<td>(3P + 3G_T + 4G_1)</td>
<td>(5P + 4G_T + 7G_1)</td>
</tr>
<tr>
<td>[DP06] (prec.)</td>
<td>CCA2</td>
<td>(4G_1 + 5G_p)</td>
<td>(3G_T + 8G_1)</td>
<td>(1P + 3G_T + 2G_2 + 7G_1)</td>
</tr>
</tbody>
</table>

Table 1. Comparison of related group signature schemes regarding signature size, signing and verification cost, where, in terms of computational costs, we only count the expensive operations in \( G_1 \), \( G_2 \), and \( G_T \) as well as the pairings. The values for [BCN\(^+\)10] and [PS16] are taken from [PS16]. We use \( \text{BCN}\(^+\) \) to denote anonymity in the sense of [BCN\(^+\)10] and note that precomputation in [BBS04,DP06] requires to store extra elements in \( G_T \).
values on an ARM-Cortex-M0+ with drop-in hardware accelerator [UW14]. This processor is small enough to be suitable to be employed in smart cards or wireless sensor nodes [UW14].

Compared to [BBS04] and [DP06], we achieve shorter signatures and are more efficient with respect to signature generation and verification by some orders of magnitude. Recall, while [DP06] provides CCA2-full anonymity, the security guarantees provided by [BBS04] are even weaker than the ones provided by our scheme, i.e., it only provides CPA-full anonymity. Furthermore, for the schemes which do not achieve full anonymity and are proven secure in a weaker model while following a similar paradigm as we do, the efficiency in our scheme is in between [BCN+10] and [PS16] for verification. It seems that the stronger security notion in our scheme comes at the cost of slightly larger signatures when compared to [BCN+10,PS16].

Regarding signature generation, we want to emphasize that our scheme is the fastest among the schemes used for comparison. This is of particular importance since signature generation is most likely to be executed on a constrained device.

We note that if the weaker anonymity notion CPA-full anonymity [BBS04] is sufficient (recall that here the anonymity adversary does not have access to the Open oracle), one could replace the CCA2-secure encryption scheme with a CPA-secure encryption scheme. However, since the choice of the encryption scheme neither has an impact on the signature size nor the computational efficiency of signing and verification, opting for a weaker security notion only seems to make sense in very specific applications.

Finally, we mention two interesting open points and leave a rigorous investigation open as future work. Firstly, it would be interesting to investigate whether our scheme provides the notion of opening soundness introduced by Sakai et al. [SSE+12]. Furthermore, our construction paradigm also seems to be interesting when it comes to efficient standard model instantiations.

References


