Dynamic Group Signatures with Verifier-Local Revocation

Callum London, Daniel Gardham, Constantin Cătălin Drăgan
Surrey Centre for Cyber Security, University of Surrey, Guildford, United Kingdom
{c.london, daniel.gardham, c.dragan}@surrey.ac.uk

***Abstract*—Group Signatures are fundamental cryptographic primitives that allow users to sign a message on behalf of a predefined set of users, curated by the group manager. The security properties ensure that members of the group can sign anonymously and without fear of being framed. In dynamic group signatures, the group manager has finer-grained control over group updates while ensuring membership privacy (i.e., hiding when users join and leave). The only known scheme that achieves standard security properties and membership privacy has been proposed by Backes et al. CCS 2019. However, they rely on an inefficient revocation mechanism that re-issues credentials to all active members during every group update, and users have to rely on a secure and private channel to join the group.**

In this paper, we introduce a dynamic group signature that supports verifier local revocation, while achieving strong security properties, including membership privacy for users joining over a public channel. Moreover, when our scheme is paired with structure-preserving signatures over equivalence class it enjoys a smaller signature size compared to Backes et al. Finally, as a stand-alone contribution we extend the primitive Asynchronous Remote Key Generation (Frymann et al. CCS 2020) with trapdoors and introduce new security properties to capture this new functionality, which is fundamental to the design of our revocation mechanism.

***Index Terms*—dynamic group signatures, membership privacy, verifier-local revocation**

1. Introduction

Group signatures, introduced by Chaum and Van Heyst [1], allow a group manager to delegate signing capabilities to a set of group members (i.e., signers). They are often used in commercial settings where employees sign on behalf of the company without needing to bear public responsibility, e.g., anonymous electronic cash payment [2], [3]. This is ensured by *unforgeability* [4], [5] where only members of that group can produce valid signatures, *anonymity* [5], [6], [7] such that the signature does not reveal the identity of the signer, and by *traceability* [5], [6], [7] where the group manager can prove that a signature was produced by a specific user. A specific example of group signatures in an organisational setting is DocuSign's *Signing Groups* feature [8], but has seen uses in other domains such as vehicle-to-vehicle communication [9], [10].

Unforgeability is often enhanced to a stronger notion called *non-frameability* [5], [6] where honest signers cannot be framed to have produced a signature even if all other signers and group manager being corrupted. When attackers have access to leaked signer keys, then anonymity is modeled as *full anonymity* [4], [11], while limiting the adversary by hiding traced signatures gives the weakened definition of *selfless anonymity* [12], [13]. Any scheme that satisfies full-anonymity also satisfies anonymity. *Full traceability* [4], [11] models colluding signers and in [4] it has been shown to imply traceability and non-frameability.

Depending on how groups membership is established, we have *static* [1], [4], [12], [14], [11]—fixed and set at setup, and *dynamic* [6], [7]—users can join and leave the group. *Partial dynamic* [5], [13] group signatures are a special case of dynamic, where the role of group manager is divided into 2 entities - one for managing membership (e.g., join and leave) and another for tracing signatures. To strengthen privacy of dynamic group signatures Backes et al. [7] have formalised *membership privacy* to model an adversary's ability to identify when users join or leave the group.

In the context of (partial) dynamic group signatures the joining phase can be done over *public channel* [15], [16], [13] or over secure and *private channel* [5], [7].

To enable appropriate removal of users from groups, dynamic group signatures rely on well-established revocation techniques, e.g., revocation lists (i.e., RL - *public revocation list* [6] and VLR - *verified-local revocation* [12], [14], [13]), and *secret credentials* [7]. Schemes based on secret credentials incorporate a huge overhead as it relies on re-issuing credentials to all signers (except those revoked), Schemes based on RL rely on the signer having up-to-date revocation lists and showing that their signing key has not been revoked (e.g., using zero-knowledge proofs), while for VLR the verifier must have up-to-date information and check some anonymised signer token is not in the revocation list. Using *epochs* that limit frequency of re-issuing credentials or size of revocation list can help mitigate some of the overhead complexity.

Backes et al. [7] have shown how to design generic group signatures that satisfy membership privacy, anonymity, traceability and non-frameability. However, their design requires a *private channel* to perform the join and relies on *inefficient secret credential revocation* methods.

Open problem: can generic dynamic group signatures be designed while ensuring: (1) a more efficient revocation

mechanism, (2) a join phase over public channels, and (3) maintaining the same security guarantees: anonymity, traceability, non-frameability and membership privacy?

We note that such a primitive would find uses in, for example, Direct Anonymous Attestation (which is a type of group signature) used within Trusted Platform Modules (TPMs). Kumar et al. [17] propose a DAA-based subscription service that uses the local revocation mechanism of the verifier to shift computational overhead away from the low-compute TPM and to the verification server. Membership privacy would enhance the privacy guarantees in such context by preventing the identification of users as they join or leave the subscription service. Another example is for the use in privacy-enhancing crowd sensing [18] which involves individuals using mobile devices (such as smartphones or wearables) to collectively gather and share data to analyze or predict processes of common interest. Again, a verifier-local revocation mechanism is used to reduce computational burden on smart devices and onto the verifying data-collection server. Strong dynamic privacy properties prevent a user from being associated with its contributed data, and on the other hand, the tracing ability of the group manager can be used to identify those maliciously contributing false data, which is known problem in crowd sensing at scale [19].

1.1. Contributions

In this paper, we answer the open problem in the affirmative and present DGS-VLR—a generic dynamic group signature that supports verifier local revocation, and offers an efficient group membership update procedure. We achieve this using a novel two-step approach that uses unlinkable 'intermediate' keys that can be revoked without compromising the membership privacy of the group member. We give a generic construction based on Asynchronous Remote Key Generation [20], Signatures with Flexible Public Keys [11], Digital signatures [21], and a Non-interactive Zero-Knowledge Proof (NIZK) which we instantiate with a Structure-Preserving Signature over Equivalence Classes (SPS-EQ) [22].

This is comparable with the state-of-the-art by Backes et al. [7] as we achieve the same strong security properties (anonymity, traceability, non-frameability and membership privacy) while reducing the complexity of the group update procedure to be linear in the number of additions/revocations, instead of linear in total group members. Moreover, we provide group-based instantiations of the building blocks that allows our construction to achieve smaller signature sizes (cf. Table 1 and Table 2), In particular, our signature has size: $11\mathbb{G}_1 + 5\mathbb{G}_2 + 512\texttt{bits}$. An additional property of our scheme is that it allows users to join groups over public channels without interactive communication with the group manager.

Furthermore, to enable our construction, we extend Asynchronous Remote Key Generation with trapdoors that allow linking of keys under restricted conditions. This novel functionality is fundamental to the design of our efficient revocation mechanism and could be of independent interest in other privacy-preserving primitives.

1.2. Related Work

We compare our scheme with other known and state-of-the-art group signatures in Table 1. We focus on the **Type** of scheme (i.e., Dynamic, partial Dynamic or Static), if they satisfy **Membership Privacy** (i.e., Yes, No), over what type of channel the **Join** procedure is performed (i.e., private, public, or non-available), the method and cost of **Revocation** (e.g., RL - public revocation list, VLR - verified-local revocation, and secret credential), the standard **Security** properties (e.g., anonymity, traceability, non-frameability), and the **Size of the Signature**.

The cost of revocation for r user revocations (i.e., calling r times the Revoke algorithm) for secret credentials is large as it need to re-issue credentials to all n active members during each revocation - $\mathcal{O}(r \cdot n)$. Schemes based on revocation lists (i.e., RL and VLR) need to only compute information for the revoked signer - $\mathcal{O}(r)$. However, VLR may not be preferred over RL as the verifier needs to maintain an up-to-date revocation list, typically of size $\mathcal{O}(r)$.

Bellare et. al. [4] introduced a static group signature that satisfy stronger notions of anonymity and traceability (i.e., full-anonymity and full-traceability) that imply existing definitions of anonymity, traceability, unforgeability and non-frameability. This has been extended in [5] to partial dynamic group signatures where two different entities have roles for joining and tracing, while only satisfying anonymity, traceability and non-frameability. Both [4] and [5] do not consider any mechanism for revocation. Additionally, they do not provide any instantiation for their primitives (i.e., what type of one-time signature they use) as such it's not possible to estimate the size of the signature.

Boneh and Shacham [12] introduced a static group signature with a verification mechanism called verifier-local revocation (VLR) where the verifier assumes responsibility for checking revocation. However, their scheme only satisfies a weaker notion of anonymity, called selfless anonymity, where the adversary doesn't have access to the signer's key. This has been extended to full-anonymity in Ishida et al. [14] via the Sign-then-Encrypt paradigm. Both [12] and [14] have an implicit traceability due to the group manager generating signing keys and revocation tokens for users during joining phase. The signature size for [12] is small, while for [14] it can't be estimated (e.g., they rely on a complex NIZK and do not show how to instantiate it).

Bootle et. al. [6] introduces a unified model for dynamic group signatures and their security properties (i.e., full anonymity, anonymity, selfless anonymity, full traceability, traceability, and non-frameability). Our scheme DGS-VLR relies on the same definitions for anonymity, traceability and non-frameability. Additionally, Bootle et. al. show that the schemes proposed by Libert et al. [15], Libert et al. [23] and Nakanishi et al. [16] satisfy their new security models and are anonymous, traceable and non-frameable. The only significant difference between [15] and [23] is a drastic

Table 1. COMPARISON OF GROUP SIGNATURE WHERE "-" MEANS NON-AVAILABLE, VLR = VERIFIER LOCAL REVOCATION, RL = REVOCATION LIST, FA = FULL ANONYMITY, SA = SELFLESS ANONYMITY, A = ANONYMITY, T = TRACEABLE, NF = NON-FRAMEABLE, U = UNFORGEABLE, FT = FULL TRACEABILITY. WE USE r AS THE NUMBER OF REVOKED USERS AND n THE SIZE OF THE GROUP.

Scheme	Type	Membership Privacy	Join	Revocation		Security	Signature Size
				Type	Cost		
Bellare et. al. [4]	Static	No	-	-	-	fA, fT, NF	-
Bellare et. al. [5]	partial Dynamic	No	private	-	-	A, T, NF	-
Boneh et al. [12]	Static	No	-	VLR	$\mathcal{O}(r)$	sA, T, U	$2\mathbb{G}_1 + 5\mathbb{Z}_q$
Ishida et al. [14]	Static	No	-	VLR	$\mathcal{O}(r)$	fA, T, U	-
Libert et al [15] (SD)	partial Dynamic	No	public	RL	$\mathcal{O}(r)$	A, T, NF	$180\mathbb{G}_1$
Nakanishi et al [16]	Dynamic	No	public	RL	$\mathcal{O}(r)$	A, T, NF	$10\mathbb{G}_1 + 3\mathbb{G}_2 + 30\mathbb{Z}_q$
Bichsel et. al. [13]	Dynamic	No	public	VLR	$\mathcal{O}(r)$	sA, T, NF	$4\mathbb{G}_1 + 5\mathbb{Z}_q$
Backes et. al. [11]	Static	No	-	-	-	fA, fT, NF	$20\mathbb{G}_1 + 5\mathbb{G}_2$
Backes et. al. [7]	Dynamic	Yes	private	secret credential	$\mathcal{O}(r \cdot n)$	A, T, NF	$28\mathbb{G}_1 + 15\mathbb{G}_2 + 512\texttt{bits}$
This Work w/ NIZK	**Dynamic**	**Yes**	**public**	**VLR**	$\mathcal{O}(r)$	**A, T, NF**	$(4n+21)\mathbb{G}_1 + 19\mathbb{G}_2 + 512\texttt{bits}$
This Work w/ SPS-EQ	**Dynamic**	**Yes**	**public**	**VLR**	$\mathcal{O}(r)$	**A, T, NF**	$11\mathbb{G}_1 + 5\mathbb{G}_2 + 512\texttt{bits}$

reduction in the size of the public key. All three schemes [15], [23], [16] use revocation lists with [16] producing a smaller signature size compared to [15], [23].

An alternative approach to partial-dynamic group signatures is proposed by Bichsel et. al. [13] relies on re-randomisable signatures. They show their scheme is non-frameable, traceable, and selfless-anonymous. This has been extended by Backes et. al. [11] that used Signatures with Flexible Public Keys (FPK) to obtain static group signatures with smaller signature size while ensuring full-anonymity and full-traceability. Another extension by Backes et. al. [7] focused on introducing membership privacy for when users join/leave the group (cf. Table 1) and ensuring anonymity, traceability and non-frameability. However, revocation requires re-issuing credentials to all group users, for each member that is revoked, i.e. has complexity $\mathcal{O}(r \cdot n)$.

In this work, we achieve strong security properties: anonymity, traceability, non-frameability and membership privacy, whilst also supporting new more efficient revocation methods whose complexity is only linear in the number of revoked users, i.e. $\mathcal{O}(r)$.

2. Preliminaries

In this section we define the relevant cryptographic building block that we used in our construction our DGS-VLR scheme: Asynchronous Remote Key Generation [20], Signatures with Flexible Public Keys [11], and Non-interactive Zero-Knowledge [24].

2.1. Asynchronous Remote Key Generation

An Asynchronous Remote Key Generation (ARKG) protocol [20] allows a party to create a public key on behalf of another, without exchanging any private information. The issuing party uses an existing long-term public key pk belonging to the receiver to derive a new public key pk'. In a one-round communication over a public channel the issuer sends some credentials cred, from which the reciever is able to derive a corresponding secret key sk', using knowledge of their secret key sk. There are two security properties, *pk-unlinkability* that ensures that pk and pk' cannot be linked, and *sk-security* which ensures that only a party with knowledge of sk can derive a valid sk'. There are 4 flavours of this property that captures an adversary's abilities: in this paper we make use of variant *malicious strong* key secrecy. ARKG has already seen uses in privacy-preserving delegatable cryptography by enabling proxy signatures with unlinkable warrants for use in the WebAuthn ecosystem [25], whilst the original work focused on discrete-log key pairs, there have been extensions to pairings [26] and even for lattices [27], [28], [29].

We introduce a variant of the ARKG scheme where pk and pk' are linked under specific conditions, i.e., using a secret token τ generated as an additional output by the DerivePK algorithm. We introduce a new algorithm ChkLink to link two public keys and define a new security property, *linking soundness*, over this new algorithm.

Definition 1 (Asynchronous Remote Key Generation [20]). An ARKG scheme is a tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{DerivePK}, \mathsf{DeriveSK}, \mathsf{ChkKeys}, \mathsf{ChkLink})$ defined as:

- $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(\lambda)$ takes as input the security parameter λ and outputs a set of public parameters pp. The public parameters are implicit input to all other algorithms
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{KeyGen}()$ on input pp samples a private-public key pair $(\mathsf{sk}, \mathsf{pk})$.
- $(\mathsf{pk}', \mathsf{cred}, \tau) \leftarrow \mathsf{DerivePK}(\mathsf{pk}, \mathsf{aux})$ produces a new public key pk', along with credentials cred and a link τ, from pk with some (potentially empty) auxiliary input aux. For simplicity, we abuse notation and use $\mathsf{DerivePK}(\mathsf{pk})$ when no aux is provided.
- $\mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{sk}, \mathsf{cred})$ computes deterministically sk together with the information provided by cred to derive a new private key sk' corresponding to pk'. It may fail, in which case it outputs $\perp$ instead.
- $\{0,1\} \leftarrow \mathsf{ChkKeys}(\mathsf{sk}', \mathsf{pk}')$ checks whether the given pair $(\mathsf{sk}', \mathsf{pk}')$ forms a valid private-public key pair, outputting 1 if so and 0 otherwise.
- $\{0,1\} \leftarrow \mathsf{ChkLink}(\mathsf{pk}, \mathsf{pk}', \mathsf{cred}, \tau)$ decides whether the given link τ proves that pk' and cred were derived from pk, outputting 1 if so and 0 otherwise.

Correctness ensures with overwhelming probability a valid new key pair is generated and they can be linked using the

token τ, if the protocol steps were followed. More precisely, $1-\mathsf{negl}(\lambda) \leq \Pr\left[\mathsf{ChkKeys}(\mathsf{sk}', \mathsf{pk}') = 1\right]$ and $1-\mathsf{negl}(\lambda) \leq \Pr\left[\mathsf{ChkLink}(\mathsf{pk}, \mathsf{pk}', \mathsf{cred}, \tau) = 1\right]$ given

$$\begin{aligned} \mathsf{pp} &\leftarrow\!\$\ \mathsf{Setup}(\lambda) \\ (\mathsf{sk}, \mathsf{pk}) &\leftarrow\!\$\ \mathsf{KeyGen}() \\ (\mathsf{pk}', \mathsf{cred}, \tau) &\leftarrow \mathsf{DerivePK}(\mathsf{pk}) \\ \mathsf{sk}' &\leftarrow \mathsf{DeriveSK}(\mathsf{sk}, \mathsf{cred}). \end{aligned}$$

The security properties *pk-unlinkability* and *sk-security* are adapted from the original work [20] (with considerations of the improved definitions of follow up works [29], [28]) to support the additional output τ and algorithm ChkLink. We highlight that this does not change the intuition and goals, as the adversary does not have access to τ in either experiment. Intuitively, for *pk-unlinkability* the adversary cannot distinguish between a fresh key generation process and a key derived using ARKG, while for *sk-secrecy* the adversary can't compute the secret key matching the derived public key. The experiments are presented in the full version of this paper. We have the following advantages negligible in λ for an efficient adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\mathsf{pku}}_{\mathcal{A},\mathsf{ARKG}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{pku}}_{\mathcal{A},\mathsf{ARKG}} = 1\right] - \frac{1}{2}\right| \text{ and}$$

$$\mathsf{Adv}^{\mathsf{sks}}_{\mathcal{A},\mathsf{ARKG}}(\lambda) = \Pr\left[\mathsf{Exp}^{\mathsf{sks}}_{\mathcal{A},\mathsf{ARKG}} = 1\right]$$

In accordance with the adapted syntax definition, we introduce an additional property, namely the soundness of a proof of linking. This captures the probability that a derived key pk' is created from pk with link τ, implies that the ChkLink algorithm succeeds.

Definition 2 (Linking Soundness)**.** ARKG satisfies *linking soundness* if for any PPT adversary $\mathcal{A}$ the following advantage is negligible in λ:

$$\mathsf{Adv}^{\mathsf{l\text{-}sound}}_{\mathcal{A},\mathsf{ARKG}}(\lambda) = \Pr\left[\begin{array}{l} \mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(\lambda) \\ (\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}() \\ (\mathsf{pk}', \mathsf{cred}, \tau) \leftarrow \mathcal{A}(\mathsf{pk}) \\ b \leftarrow \mathsf{ChkLink}(\mathsf{pk}, \mathsf{pk}', \mathsf{cred}, \tau) \\ \mathsf{sk}' \leftarrow \mathsf{DeriveSK}(\mathsf{sk}, \mathsf{cred}) \\ b' \leftarrow \mathsf{ChkKeys}(\mathsf{sk}', \mathsf{pk}') \\ \textbf{return } b \wedge \neg b' \end{array}\right]$$

2.2. Signatures with Flexible Public Keys

Signatures with Flexible Public Keys (SFPK) [11] are reminiscent of standard digital signatures but allow a signer to change their public key while still being able to sign. This is achieved by changing keys within an equivalence class, and importantly, signatures verify against all keys in the equivalence class. An initial key pair $(\mathsf{sk}, \mathsf{pk})$ may be generated with a trapdoor τ that can be used to check whether another key pk' lies in the same class denoted $\mathsf{pk}' \in [\mathsf{pk}]$.

Definition 3 (Signatures with Flexible Public Keys [11]). An SFPK scheme is a tuple $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TrapGen}, \mathsf{ChgPK}, \mathsf{ChgSK}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{ChkRep})$ consisting of the following algorithms.

- $\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(\lambda)$ takes as input the security parameter λ and outputs a set of public parameters pp. These are an implicit input for all other algorithms.
- $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}()$ samples a private-public key pair.
- $(\mathsf{pk}', \tau) \leftarrow \mathsf{TrapGen}(\mathsf{pk})$ produces a public key pk' with trapdoor τ, associated with the equivalence class $[\mathsf{pk}]$ [1].
- $\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{pk}, r)$ returns an equivalent public key $\mathsf{pk}' \in [\mathsf{pk}]$, for a public key pk and salt r.
- $\mathsf{sk}' \leftarrow \mathsf{ChgSK}(\mathsf{sk}, r)$ takes a secret key sk and salt r, returning an updated secret key sk'.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$ uses the key sk to produce a signature σ on the input message m.
- $\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \sigma)$ outputs 1 if (m, σ) forms a valid message-signature pair against pk and 0 otherwise.
- $\{0,1\} \leftarrow \mathsf{ChkRep}(\tau, \mathsf{pk})$ outputs 1 if pk lies in the equivalence class associated with τ, and 0 otherwise.

Correctness of SFPK ensures the following properties hold

1) Algorithms KeyGen and TrapGen produce the same distribution over key-pairs.
2) With overwhelming probability the following two verification holds: $1 \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m))$ and $1 \leftarrow \mathsf{Verify}(\mathsf{pk}', m, \mathsf{Sign}(\mathsf{sk}', m))$, given valid key-pairs $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}(\lambda)$ and $\mathsf{pk}' \leftarrow \mathsf{ChgPK}(\mathsf{pk}, r)$; $\mathsf{sk}' \leftarrow \mathsf{ChgSK}(\mathsf{sk}, r)$.
3) With overwhelming probability the algorithm $\mathsf{ChkRep}(\tau, \mathsf{pk}')$ returns 1 iff $\mathsf{pk}' \in [\mathsf{pk}]$ for any $(\mathsf{pk}', \tau) \leftarrow\!\$\ \mathsf{TrapGen}(\lambda)$.

The security of SFPK [11] is based on *class-hiding* and *unforgeability*. For class-hiding an adversary cannot distinguish between public keys generated from different equivalence classes, allowing the key-change function to "hide" the class containing the original key. The unforgeability of SFPK captures an adversary's capability to create forgeries in the context of key-adaption. That is, the adversary needs to create a new public key $\mathsf{pk}^* \in [\mathsf{pk}]$ and a forgery for that key. The formal experiments are presented in the full version of this paper. We have the following advantages negligible in λ for an efficient adversary $\mathcal{A}$:

$$\mathsf{Adv}^{\mathsf{ch}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{ch}}_{\mathcal{A},\mathsf{SFPK}} = 1\right] - \frac{1}{2}\right| \text{ and}$$

$$\mathsf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) = \Pr\left[\mathsf{Exp}^{\mathsf{suf\text{-}cma}}_{\mathcal{A},\mathsf{SFPK}} = 1\right]$$

2.3. Digital Signatures

Digital signatures (DS) [21] are standard cryptographic primitives defined by three algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. They satisfy *correctness* where with overwhelming probability we have $1 \leftarrow \mathsf{Verify}(\mathsf{pk}, m, \mathsf{Sign}(\mathsf{sk}, m))$ for $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{KeyGen}(\lambda)$. Their security property ensures that they are resistant to existential forgeries under adversarially chosen messages.

1. This is a slight modification of [11] and we expand on this in Section 6.

2.4. Non-Interactive Zero-Knowledge Proofs

For a given relation R, a language is defined as $L_R := \{x : \exists w \ (x,w) \in R\}$. A zero-knowledge proof system allows a prover to prove knowledge of a witness w to the relation R, without leaking any information about it. Formally, it is defined below.

Definition 4 (Non-Interactive Zero-Knowledge Proof System [24])**.** A NIZK proof system for a relation R is a tuple of algorithms $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{SimSetup}, \mathsf{Sim})$ defined as follows.

$\rho \leftarrow\!\$\ \mathsf{Setup}(\lambda)$ takes as input the security parameter λ and outputs a common reference string ρ.

$\pi \leftarrow \mathsf{Prove}(\rho, x, w)$ takes a common reference string ρ, statement x and witness w, returning a proof π.

$\{0,1\} \leftarrow \mathsf{Verify}(\rho, x, \pi)$ outputs 1 if π is a valid proof with respect to the common reference string ρ and statement x, and 0 otherwise.

$(\rho', \tau) \leftarrow \mathsf{SimSetup}(\lambda)$ takes as input the security parameter λ and outputs a common reference string ρ' and a trapdoor τ.

$\pi' \leftarrow \mathsf{Sim}(\rho', \tau)$ takes a common reference string ρ', statement x and returns a simulated proof π'.

A NIZK proof system is *correct* if for any $\rho \leftarrow\!\$\ \mathsf{Setup}(\lambda)$ and $(x,w) \in R$ it follows that with overwhelming probability we have $1 \leftarrow \mathsf{Verify}(\rho, x, \mathsf{Prove}(\rho, x, w))$.

Soundness. A NIZK proof system is sound if the following advantage is negligible in the security parameter λ.

$$\mathsf{Adv}^{\mathsf{sound}}_{\mathcal{A},\mathsf{NIZK}}(\lambda) \ = \ \Pr\left[\begin{array}{l} \rho \leftarrow \mathsf{NIZK.Setup}(\lambda) \\ (x,\pi) \leftarrow\!\$\ \mathcal{A}(\rho) \\ \textbf{return } \mathsf{NIZK.Verify}(\rho, x, \pi) \\ \qquad\qquad \wedge \ x \notin L_R \end{array}\right]$$

Non-interactive Zero Knowledge. A NIZK proof system has the zero-knowledge property if the following advantage is negligible in the security parameter λ. The advantage $\mathsf{Adv}^{\mathsf{zk}}_{\mathcal{A},\mathsf{NIZK}}(\lambda)$ is defined as:

$$\left| \begin{array}{l} \Pr\left[\rho \leftarrow \mathsf{NIZK.Setup}(\lambda);\ \textbf{return } \mathcal{A}^{\mathsf{NIZK.Prove}(\rho,\cdot,\cdot)}(\rho)\right] \\ -\Pr\left[(\rho',\tau) \leftarrow \mathsf{SimSetup}(\lambda);\ \textbf{return } \mathcal{A}^{\mathsf{Sim}(\rho',\tau,\cdot)}(\rho')\right] \end{array} \right|$$

3. Syntax & Security Properties for DGS-VLR

In this section we give the syntax and properties of Dynamic Group Signatures with Verifier-Local Revocation. It is modeled based on the definition of Fully Dynamic Group Signatures [6] but with the requirement that the joining phase can be carried out on a public channel. Furthermore, the model [6] separates out the group manager and tracing authorities; however we consider the case that both aspects are managed by the group manager, a consequence of implementing verifier-local revocation. We adopt standard security properties, namely anonymity, traceability, non-frameability (inspired by [6]), and Membership Privacy (inspired by [7]), with minor adaptations to account for the new functionality of the group manager to revoke users from the scheme.

Definition 5 (Dynamic Group Signatures with Verifier Local Revocation). A Dynamic Group Signature with Verifier Local Revocation DGS-VLR consists of the following algorithms:

$\mathsf{pp} \leftarrow\!\$\ \mathsf{Setup}(\lambda)$ takes as input the security parameter λ and outputs a set of public parameters pp. As part of the public parameters we also initialise an empty data structure REG that maps user identities with their long term public keys. The parameter pp is an implicit input to every other algorithm.

$(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{info}) \leftarrow\!\$\ \mathsf{GKeyGen}()$ generates a group public key gpk and a group manager secret key gmsk. It also initializes the public group information info.

$(\mathsf{usk}_{id}, \mathsf{upk}_{id}) \leftarrow \mathsf{RegUser}(id)$, executed by the user, it samples a private-public key pair $(\mathsf{usk}_{id}, \mathsf{upk}_{id})$ given the user identity id. The registry REG is updated only if nothing is previously stored for this used, e.g., **if** $\mathsf{REG}[id] = \perp$ **then** $\mathsf{REG}[id] \leftarrow \mathsf{upk}_{id}$.

$(\mathsf{gmsk}, \mathsf{info}) \leftarrow \mathsf{AddUser}(\mathsf{gmsk}, \mathsf{info}, id)$ executed by the group manager, this algorithm adds a registered user id to the group, and updates the group manager secret key gmsk and public group information info.

$\mathsf{info} \leftarrow \mathsf{Revoke}(\mathsf{gmsk}, \mathsf{info}, id)$ revokes a group member id, updating the public group information info accordingly.

$\Sigma \leftarrow \mathsf{Sign}(\mathsf{info}, \mathsf{usk}_{id}, m)$ run by user id, it uses the public group information info, the user's secret key usk_{id} and a message m to produce a signature Σ on m.

$\{0,1\} \leftarrow \mathsf{Verify}(\mathsf{gpk}, \mathsf{info}, m, \Sigma)$ checks whether (m, Σ) forms a valid message-signature pair against the group public key and information $(\mathsf{gpk}, \mathsf{info})$ and a list of revocation tokens RL. For ease, we include RL in info. It outputs a bit $b \in \{0,1\}$ with 1 indicating acceptance and 0 rejection.

$(id, \pi) \leftarrow \mathsf{Open}(\mathsf{gpk}, \mathsf{gmsk}, \mathsf{info}, m, \Sigma)$ outputs the identity id of a user along with a proof π tracing the signature (m, Σ) to this user. In the case that Open fails to trace, it outputs $(\perp, \perp)$.

$\{0,1\} \leftarrow \mathsf{Judge}(\mathsf{gpk}, \mathsf{info}, id, \pi, m, \sigma)$ uses the group public key and information gpk, info to validate a proof of authorship π for some message-signature pair (m, σ) attributing it to a user id. It outputs a bit $b \in \{0,1\}$ with 1 indicating acceptance and 0 rejection.

Remark 1. The separation of the join/issue protocol into the RegUser and AddUser algorithms allows the possibility for the same user to be added to the group multiple times under a single registry. In particular this behaviour is useful for reinstating a revoked user without having to allocate them a new id, i.e. temporary revocation. Alternatively, enforcing AddUser to ignore existing or revoked group members recovers existing models [6], [7].

3.1. Correctness and Security Properties

We now state the formal definitions of correctness and the security properties.

Notation. The experiments make use of the following additional lists.

- HU: A list of honest registered users, populated by the registration oracle $HonestU$. For any $id \in HU$ we have a stored public key $\mathsf{REG}[id] \neq \perp$. The adversary can additionally create users outside of this ($id' \notin HU$) and populate $\mathsf{REG}[id']$ directly with some public key.
- BU: A list of corrupt users, whose keys have been compromised by the adversary calling $RevealU$. We have that for any $id \in BU$ implies $id \in HU$.
- AM: A list of active group members who are able to sign successfully. An active group member becomes inactive if revoked by $RevokeU$. Note $id \in AM$ implies $\mathsf{REG}[id] \neq \perp$, but does not imply $id \in HU$.
- Q_{sig}: list of queries (id, m) made as input to $SignU$.
- OBL: A list of message-signature pairs that the adversary has been challenged on. The adversary is prohibited from running the Open algorithm (via the oracle $OpenSig$) to identify the user that issued that signature.
- $Chl = \{id_0, id_1\}$: Stores a pair of challenge users chosen by the adversary in the anonymity, join, and leave experiments.

Oracles. We introduce the following oracles that are used in the security experiments.

- $\mathsf{upk}_{id} \leftarrow HonestU(id)$ simulates $\mathsf{RegUser}(id)$ to create a private-public key pair $(\mathsf{usk}_{id}, \mathsf{upk}_{id})$, and updates $HU \leftarrow HU \cup \{id\}$. Note that the registry is also updated $\mathsf{REG}[id] \leftarrow \mathsf{upk}_{id}$.
- $\mathsf{usk}_{id} \leftarrow RevealU(id)$ returns the signing key of an user with the given an identity that is not part of the challenge $id \in HU \setminus Chl$. We also update $BU \leftarrow BU \cup \{id\}$. Note all $id \in BU$ implies $id \in HU$.
- $CorruptU(id, \mathsf{pk})$ updates $\mathsf{REG}[id] \leftarrow \mathsf{pk}$ only if $\mathsf{REG}[id] = \perp$. This simulates the registration of a public key of a corrupted user (i.e, under the control of the adversary). Note $\mathsf{REG}[id] = \perp$ implies $id \notin HU$, but $\mathsf{REG}[id] \neq \perp$ does not imply $id \in HU$.
- $\mathsf{info} \leftarrow IssueU(id)$ simulates AddUser on the identity id, making them an active group member. This adds a new entry to info associated with the corresponding user even if they were already an active member. The list of active group members is updated $AM \leftarrow AM \cup \{id\}$.
- $\mathsf{info} \leftarrow ActivateU(id)$ simulates AddUser only for $id \notin AM$. Otherwise, it returns previously stored info. The list of active group members is updated $AM \leftarrow AM \cup \{id\}$ as for $IssueU$.
- $\mathsf{info} \leftarrow RevokeU(id)$ simulates Revoke to revoke user id from the group, and updates $AM \leftarrow AM \setminus \{id\}$. The user can later be added back to the group by using $IssueU(id)$ or $ActivateU(id)$.
- $\sigma \leftarrow SignU(\mathsf{info}, id, m)$ simulates Sign with usk_{id} and returns the signature σ for any $id \in HU \setminus Chl$. We also update $Q_{sig} \leftarrow Q_{sig} \cup \{(id, m)\}$.
- $id \leftarrow OpenSig(m, \sigma)$ simulates Open and returns the identity id without a proof, except if (m, σ) is in the opening ban list OBL.
- $\sigma \leftarrow Chall_{b,anon}(\mathsf{info}, m)$ runs Sign with usk_{id_b} for the challenge bit $b \in \{0, 1\}$ and challenge user identities $Chl = \{id_0, id_1\}$. It returns the signature σ only if both users have been honestly generated and are active members $Chl \subseteq HU$ and $Chl \subseteq AM$; otherwise it returns $\perp$. Notice that for the anonymity experiment the adversary can corrupt the users and have access to their signing keys, e.g., $Chl \subseteq BU$. The opening ban list is updated $OBL \leftarrow OBL \cup \{(m, \sigma)\}$.
- $\sigma \leftarrow Chall_{b,join}(\mathsf{info}, m)$ simulates Sign with usk_{id_b} for the challenge bit $b \in \{0, 1\}$ and challenge user identities $Chl = \{id_0, id_1\}$. Returns the signature σ only if both users haven't been corrupted $Chl \subseteq HU$ and $Chl \cap BU = \emptyset$. The following list is updated $OBL \leftarrow OBL \cup \{(m, \sigma)\}$.
- $Chall_{b,leave}(\mathsf{info}, m)$ simulates Sign with usk_{id_b} for the challenge bit $b \in \{0, 1\}$ and challenge user identities $Chl = \{id_0, id_1\}$. Returns the signature σ, unless either user is corrupt or has been used to query the signing oracle.

Correctness. This ensures that if the protocol steps are followed, for any message m, with overwhelming probability we have:

1) A valid group signature is created by a user if and only if they are an active group member, and
2) An honest judge will accept the proof regarding the identity of the signer, given an opening of a valid signature.

Formally, using the experiment in Figure 1 we have

$$1 - \mathsf{negl}(\lambda) \;\leq\; \Pr\left[\mathsf{Exp}^{\mathsf{corr}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}} = 1\right].$$

Anonymity. A DGS-VLR scheme is anonymous if an adversary cannot determine the identity of a user $id_b \in \{id_0, id_1\}$ given polynomially many challenge message-signature pairs (m, Σ) under the signing key usk_{id_b}. This is the case even if the adversary has access to all group members' signing keys and an opening oracle (restricted on the challenge pairs).

Definition 6 (Anonymity)**.** A dynamic group signature with verifier local revocation scheme is anonymous if the following advantage is negligible in λ, where $\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}$ is defined in Figure 1.

$$\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda) \;=\; \left|\Pr\left[\mathsf{Exp}^{\mathsf{anon}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}} = 1\right] - \frac{1}{2}\right|$$

Remark 2. Our anonymity experiment uses two challenges oracles to capture the notion of swapping the order in which the identities id_0 and id_1 are input to the challenge oracle (so the oracle effectively signs for $1 - b$ instead of b). While the experiment is typically defined using a single oracle with malleable input for challenge identities in the literature (e.g. [6], [7]), using distinct oracles leads to both an equivalent security definition and a more direct proof argument for our construction.

Traceability. A dynamic group signature scheme is traceable if an adversary cannot create a valid message-signature

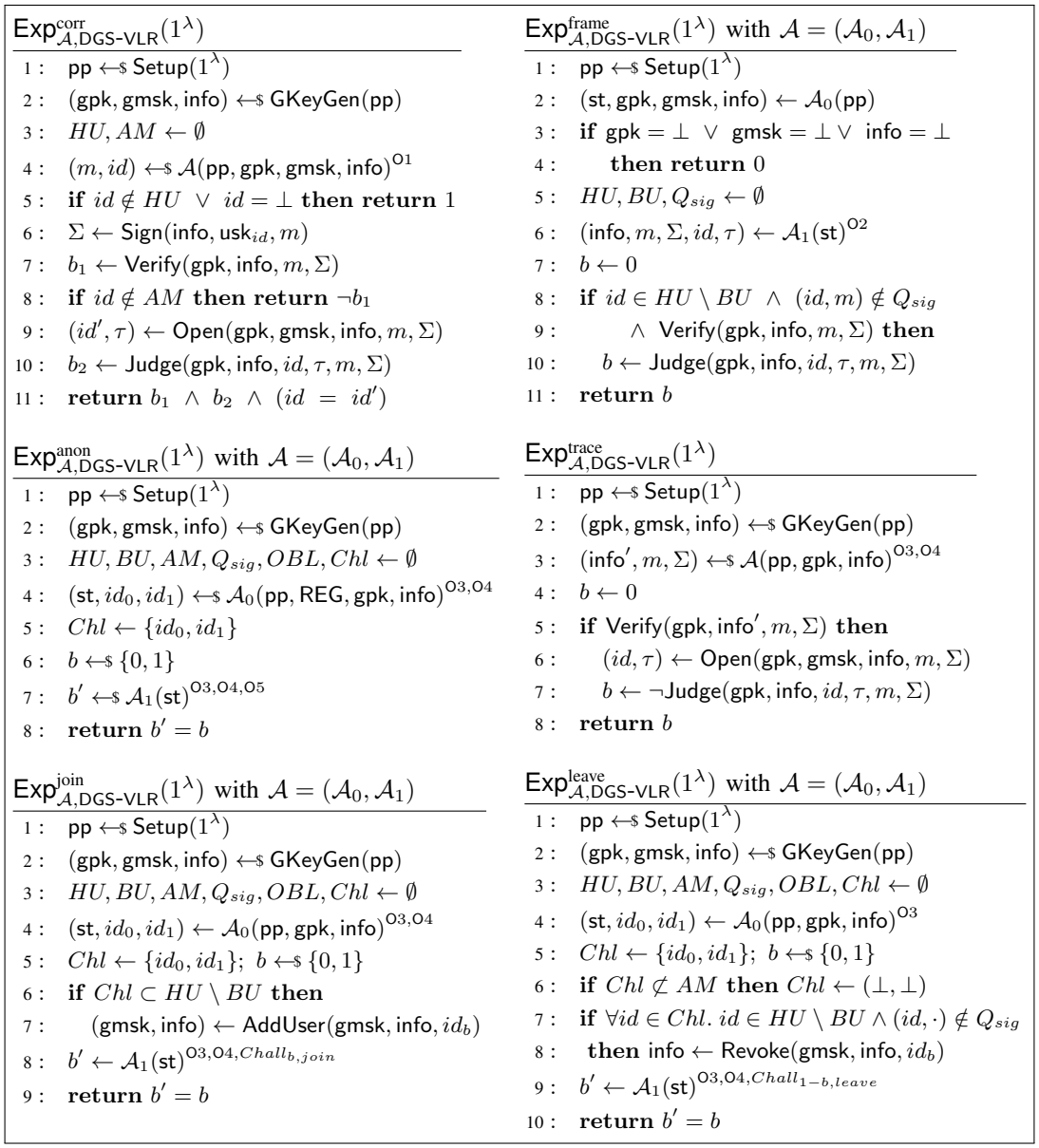

Figure 1. Group signature security games with oracles $\mathsf{O1} = \{HonestU, IssueU, RevokeU\}$, $\mathsf{O2} = \{HonestU, CorruptU, RevealU, SignU\}$, $\mathsf{O3} = \{HonestU, CorruptU, RevealU, ActivateU, RevokeU, SignU, OpenSig\}$, $\mathsf{O4} = \{IssueU\}$, and $\mathsf{O5} = \{Chall_{b,anon}, Chall_{1-b,anon}\}$.

pair (m, Σ) that does not open to a valid identity id and proof π for an existing group member. Note that our definition slightly departs from existing work (e.g. [6], [7]) as we do not separate the role of tracing authority and group manager.

Definition 7 (Traceability)**.** A dynamic group signature scheme is traceable if the following advantage is negligible in λ, where $\mathsf{Exp}^{\mathsf{trace}}_{\mathcal{A},\text{DGS-VLR}}$ is defined in Figure 1.

$$\mathsf{Adv}^{\mathsf{trace}}_{\mathcal{A},\text{DGS-VLR}}(\lambda) \ = \ \Pr\left[\mathsf{Exp}^{\mathsf{trace}}_{\mathcal{A},\text{DGS-VLR}} = 1\right]$$

Non-Frameability. A dynamic group signature scheme is non-frameable if an adversary cannot create a valid message-signature pair (m, Σ) that opens to an identity id of an honest user for which the adversary does not know the user secret key, even if the adversary can corrupt the group manager.

Definition 8 (Non-Frameability)**.** A dynamic group signature scheme is non-frameable if the following advantage is negligible in λ, where $\mathsf{Exp}^{\mathsf{frame}}_{\mathcal{A},\text{DGS-VLR}}$ is defined in Figure 1.

$$\mathsf{Adv}^{\mathsf{frame}}_{\mathcal{A},\text{DGS-VLR}}(\lambda) \ = \ \Pr\left[\mathsf{Exp}^{\mathsf{frame}}_{\mathcal{A},\text{DGS-VLR}} = 1\right]$$

Membership Privacy. Introduced by Backes et al. [7], membership privacy captures strong privacy properties for dynamic group signatures. Intuitively, it ensures privacy as groups members are added and revoked. We adopt these

notions for our scheme, and say a DGS-VLR provides membership privacy if it provides both join and leave privacy.

- The scheme provides join privacy if an adversary cannot distinguish which of two chosen honest users id_0, id_1 has been issued (regardless of whether each user is already active).
- The scheme provides leave privacy if an adversary cannot distinguish which of two chosen active honest users id_0, id_1 has been revoked.

In both games the adversary cannot request signatures on either user after the challenge is given, instead they may receive signatures anonymously via a challenge oracle for the activated user id_b in the join game and the non-revoked user id_{1-b} in the leave game. In particular, the adversary cannot use the opening oracle on these signatures.

Definition 9 (Join Privacy)**.** A dynamic group signature with VLR provides join-privacy if the following advantage is negligible in λ, where $\mathsf{Exp}^{\mathsf{join}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}$ is defined in Figure 1.

$$\mathsf{Adv}^{\mathsf{join}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda) \;=\; \left|\Pr\left[\mathsf{Exp}^{\mathsf{join}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}} = 1\right] - \frac{1}{2}\right|$$

Definition 10 (Leave Privacy)**.** A dynamic group signature with VLR provides leave-privacy if the following advantage is negligible in λ, for $\mathsf{Exp}^{\mathsf{leave}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}$ defined in Figure 1.

$$\mathsf{Adv}^{\mathsf{leave}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda) \;=\; \left|\Pr\left[\mathsf{Exp}^{\mathsf{leave}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}} = 1\right] - \frac{1}{2}\right|$$

Remark 3. The consequences of our modeling choices as discussed in Remark 1 must be carefully considered for the membership privacy experiments. For the join game, no problems arise as the change in info is the same regardless of whether id_b has been issued previously before the challenge is declared. However, for the leave game when the challenge is declared and id_b is revoked, all group information linked to id_b is impacted simultaneously. The adversary may link the amount of affected information to the number of times id_b was issued before the challenge was declared. Thus the experiment must limit each identity to only one issuing in the first phase. This purpose is fulfilled by the $ActivateU$ oracle, while the $IssueU$ oracle is prohibited for $\mathcal{A}_0$ in the leave game. Furthermore, the leave game requires that both challenge users are active members, however since one of them is revoked before the guess phase, the $Chall_{b,leave}$ oracle will not be able to check this condition directly. This is addressed by setting both challenge users to $(\perp, \perp)$ if the check fails before revocation. Then, the oracle will reject considering these users dishonest.

4. Generic Construction of DGS-VLR

In this section we detail our construction for DGS-VLR, and illustrate all our algorithms in Figure 2.

The group manager executes Setup to generate the public parameters by running the setup algorithms for an asynchronous remote key generation ARKG and a signature with flexible public keys SFPK. It instantiates a group by creating keys for a digital signature DS, as well as initialising both a list of group members pkl and a revocation list RL as empty. These are output as a group information info. The process for building the group can be summerised by the following six steps.

- Users register to be considered for the group.
- Group manager selects group members from the list of users in the registry.
- Group manager creates group signing keys for each group member via ARKG's DerivePK algorithm.
- Users identify their group signing keys using the group information (i.e. via public channels).
- Users re-randomise their obtained signing key and use it to create a signature via SFPK.
- Group manager may revoke a user by publishing a SFPK trapdoor that identifies all possible randomisations of their signing key.

A user id registers by first generating an ARKG key pair $(\mathsf{usk}, \mathsf{upk})$ and declaring ownership of the public key upk via the registry $\mathsf{REG}[id] \leftarrow \mathsf{upk}$. The user is added to the group by the group manager by executing the $\mathsf{AddUser}$ algorithm. This algorithm runs our modified $\mathsf{ARKG.DerivePK}$ algorithm on upk to create a new user key pk (and corresponding derivation information cred, τ_{cred}). The pk-unlinkability property of ARKG ensures that these two keys cannot be linked, even with knowledge of cred. The keys can however be linked with knowledge of τ_{cred}, which is stored by the group manager so that they may later open signatures to id. Furthermore, the algorithm creates an additional token τ_{pk} by executing $\mathsf{SFPK.TrapGen}$. This is stored by the group manager to enable linking in the case that the user is later revoked, by running the revoke algorithm, which appends τ_{pk} to the revoke list RL. This is published, and is authenticated with a digital signature σ_{info}, over pkl and RL, which verifies against the group managers key gpk.

To create a group signature, a user runs the Sign algorithm, which takes as input the group information info, its long-term user key usk and the message m. It begins by parsing pkl and identifying which $\mathsf{pk}, \mathsf{cred}$ correspond to its key usk. It does this by checking the MACs stored in cred. Once found, it runs $\mathsf{ARKG.DeriveSK}$ to derive sk that corresponds to the derived public key pk. It is important to notice that the derived key space to which pk belongs is carefully chosen to coincide with the key space for the SFPK scheme, thus it can randomise this key using $\mathsf{SFPK.ChgPK}$, to create a key pk' along with a corresponding key sk' using $\mathsf{SFPK.ChgSK}$. The message m is signed with $\mathsf{SFPK.Sign}$ using the key sk'. The class-hiding property of SFPK ensures that the randomised key pk' cannot be linked to pk without knowledge of τ_{pk}, and since the keys are randomised for every new signature, it implies that signatures also cannot be linked. Finally, the user creates a zero-knowledge proof Π that the key pk' indeed is the randomisation of a valid key, i.e. a pk that is in pkl. More precisely, the relation for

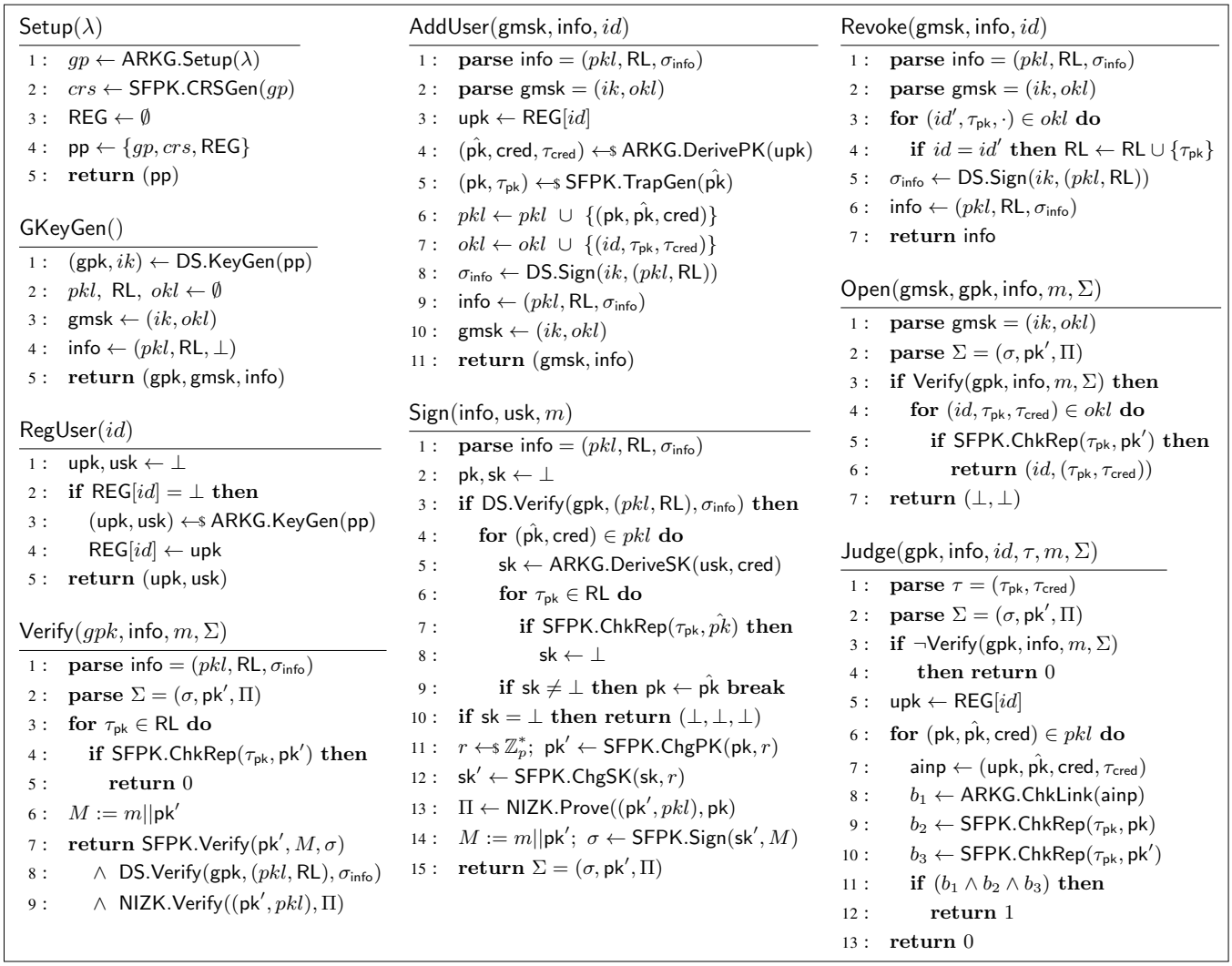

Figure 2. All algorithms for the group signature DGS-VLR.

the proof Π is given by:

$$\mathcal{R} := \{(\text{pk}', pkl), (\text{pk}); \quad \exists r \ s.t. \ \text{ChgPK}(\text{pk}, r) = \text{pk}' \wedge \text{pk} \in pkl\}$$

The group signature Σ is a tuple consisting of the SFPK signature σ, the verification key pk$'$ and the proof Π.

Verification is a public process which takes as input a candidate signature Σ, a message m and group information info. The verifier parses revocation tokens τ_{pk} from the revocation list RL, and for each one, runs SFPK.ChkRep against the verification key pk$'$ that comprises part of Σ. If the algorithm returns 1, then a users trapdoor has been added to RL and is thus revoked, and verification fails. Else, the SFPK signature σ is verified against pk$'$ on m, and the validity of the proof Π is verified. If all stages are successful, then the algorithm outputs 1, else it returns 0.

To trace a signature, the group manager runs the Open algorithm by checking for which SFPK trapdoor τ_{pk} matches the verification key pk$'$. It then outputs the corresponding identity id. This step can be publicly checked in the Judge algorithm, by verifying a signature, and then checking that the group manager's provided τ_{cred} links a user's long-term key upk to some intermediate key pk in info. Then it checks that the SFPK trapdoor τ_{pk} is a valid trapdoor for pk by running SFPK.ChkRep, and finally checks that the ephemeral verification key pk$'$ belongs to the same equivalence class by once more running SFPK.ChkRep.

As observed in Remark 1, it is possible to issue multiple public keys to the same group member, while each token τ_{pk} only revokes the individual key class [pk]. This provides the ability to reinstate a previously revoked user by issuing them a new public key.

5. Security Analysis

In this section we prove that our construction of DGS-VLR (Figure 2), is correct and satisfies Anonymity, Trace-

ability, Non-frameability, Join Privacy and Leave Privacy.

Correctness (sketch). Our scheme being correct intuitively relies on the correctness of the underlying building blocks. Furthermore, we require that the keys generated by ARKG are compatible with the instantiation of SFPK. If this is the case, then ARKG can be used to create valid keys for SFPK, and thus SFPK will not fail to produce an output. Correctness of the NIZK ensures that, since the signer has a valid witness (i.e. their derived secret key and its randomisation), then it can produce a valid proof Π.

Theorem 1 (Anonymity)**.** *The dynamic group signature scheme* DGS-VLR *defined in Figure 2 achieves the anonymity security property if* DS *is EUF-CMA secure,* NIZK *has the zero-knowledge property and* SFPK *is both class-hiding and SUF-CMA secure. More precisely, an adversary $\mathcal{A}$ wins the anonymity game defined in Definition 6 with advantage:*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda) \;\leqslant\; & \mathsf{Adv}^{\mathsf{zk}}_{\mathcal{A},\mathsf{NIZK}}(\lambda) + \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{DS}}(\lambda) \\ & + n(n-1)\ \mathsf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) \\ & + n(n-1)\ \mathsf{Adv}^{\mathsf{ch}}_{\mathcal{A},\mathsf{SFPK}}(\lambda),\end{aligned}$$

where n is the size of the group.

Proof. Denote by ε^{anon}_k the probability $\left|\Pr[\mathsf{Anon_k}] - \frac{1}{2}\right|$ where $k \in \{1,\ldots,5\}$. Notice that

$$\varepsilon^{\mathsf{anon}}_1 \;:=\; \mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda).$$

We introduce the following sequence of games, starting from the anonymity experiment.

$\mathsf{Anon_1}$ Original anonymity game.

$\mathsf{Anon_2}$ Modify the $SignU$ oracle such that instead of checking $\mathsf{DS.Verify}(\mathsf{gpk}, \sigma_{\mathsf{info}})$, check that the value of info given matches some previous output of either $IssueU$ or $RevokeU$ tracked by a list IL.

$\mathsf{Anon_3}$ Replace the prover used to generate the NIZK proof in each signature with a simulator.

$\mathsf{Anon_4}$ Extend line 5 as follows. Choose random $i, j \leftarrow\!\!\$\ \{1,\ldots,n\}$, if $(id_0, id_1) \neq (i, j)$ then abort.

$\mathsf{Anon_5}$ In line 7 replace $\mathcal{A}$'s challenge oracles $Chall_{b,anon}, Chall_{1-b,anon}$ with two copies of $Chall_{0,anon}$.

We denote by $\varepsilon^{\mathsf{anon}}_k$ the probability $\left|\Pr[\mathsf{Anon_k}] - \frac{1}{2}\right|$ where $k \in \{1,\ldots,5\}$. Notice that

$$\varepsilon^{\mathsf{anon}}_1 \;:=\; \mathsf{Adv}^{\mathsf{anon}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda).$$

Consider an adversary $\mathcal{A}$ that wins $\mathsf{Anon_1}$ but not $\mathsf{Anon_2}$. By the correctness of DS, every $\mathsf{info} \in IL$ will contain σ_{info} such that $\mathsf{DS.Verify}(\mathsf{gpk}, \sigma_{\mathsf{info}}) = 1$ so $SignU$ will behave identically in $\mathsf{Anon_1}$ and $\mathsf{Anon_2}$ for such queries. But $\mathcal{A}$ achieves a different outcome in $\mathsf{Anon_2}$ than in $\mathsf{Anon_1}$ and hence it must make at least one query to $SignU$ with some input value $\mathsf{info} \notin IL$. Furthermore, in the case that DS.Verify rejects some σ_{info}, the output of $SignU$ is simply $(\perp, \perp, \perp)$ and does not depend on the input id. In particular, when $SignU$ is called via $Chall_{b,join}$ its output does not depend on the challenge bit b. Therefore $\mathcal{A}$ must make at least one query to $SignU$ with input $\mathsf{info} \notin IL$ such that $\mathsf{DS.Verify}(\mathsf{gpk}, (pkl, \mathsf{RL}), \sigma_{\mathsf{info}}) = 1$.

We construct an adversary $\mathcal{B}$ against the euf-cma game for DS that simulates $\mathsf{Anon_1}$ for $\mathcal{A}$ while also storing IL in its state. In order to simulate $IssueU$ and $RevokeU$ $\mathcal{B}$ uses the euf-cma signing oracle. For each query to $SignU$ $\mathcal{B}$ checks whether $\mathsf{info} \notin IL$ and $\mathsf{DS.Verify}(\mathsf{gpk}, (pkl, \mathsf{RL}), \sigma_{\mathsf{info}}) = 1$. If such a query is made $\mathcal{B}$ terminates the simulation of $\mathsf{Anon_1}$ and adds info to IL. Once $\mathsf{Anon_1}$ terminates as above, or otherwise, $\mathcal{B}$ submits IL to the euf-cma game, winning in the former case. Recall that this is precisely the case that $\mathcal{A}$ wins the simulated instance of $\mathsf{Anon_1}$ without winning the corresponding instance of $\mathsf{Anon_2}$.

$$\left|\varepsilon^{\mathsf{anon}}_1 - \varepsilon^{\mathsf{anon}}_2\right| \;\leqslant\; \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{DS}}(\lambda)$$

An adversary that can distinguish $\mathsf{Anon_2}$ from $\mathsf{Anon_3}$ can distinguish the simulator from the prover, breaking the zero-knowledge property of NIZK.

$$\left|\varepsilon^{\mathsf{anon}}_2 - \varepsilon^{\mathsf{anon}}_3\right| \;\leqslant\; \mathsf{Adv}^{\mathsf{z\text{-}k}}_{\mathcal{A},\mathsf{NIZK}}(\lambda)$$

Next we consider $\mathsf{Anon_4}$. The probability that the adversary chooses the required values for i, j to avoid $\mathsf{Anon_4}$ being aborted is $1/n(n-1)$, so its advantage is adjusted by the same factor.

$$\varepsilon^{\mathsf{anon}}_4 \;=\; \frac{1}{n(n-1)}\varepsilon^{\mathsf{anon}}_3$$

Consider an adversary $\mathcal{A}$ against the problem of distinguishing between $\mathsf{Anon_4}$ and $\mathsf{Anon_5}$. If $\mathcal{A}$ uses either usk_i or usk_j to produce a signature against info and then submits the result to $OpenSig$, the behaviour of $OpenSig$ will be identical in both games. We may therefore assume without loss of generality that $\mathcal{A}$ does not make such queries. However, if $\mathcal{A}$ instead submits a signature that depends on some output of either challenge oracle, that is it depends on the underlying key pk that the oracle is using, and also if this submission is not in OBL and successfully verifies (otherwise $OpensSig$ would simply reject the query), then the behaviour of $OpenSig$ could depend on which game they are playing. Suppose that $\mathcal{A}$ does not make such a query, then $\mathcal{A}$ can be used to construct an adversary $\mathcal{B}$ against the class-hiding game as follows.

Upon receiving the challenge $(\mathsf{pp}, (\mathsf{sk}_0, \mathsf{pk}_0), (\mathsf{sk}_1, \mathsf{pk}_1), \mathsf{pk}')$ from the class-hiding game, $\mathcal{B}$ generates the group keys for $\mathsf{Anon_4}$ by calling $\mathsf{GKeyGen}(\mathsf{pp})$. In the choose phase $\mathcal{B}$ simulates the oracles for $\mathcal{A}$ according to the normal protocol of DGS-VLR except that on the i-th and j-th calls to $HonestU$, $\mathcal{B}$ takes $(\mathsf{usk}_i, \mathsf{upk}_i) \leftarrow (\mathsf{sk}_0, \mathsf{pk}_0)$ and $(\mathsf{usk}_j, \mathsf{upk}_j) \leftarrow (\mathsf{sk}_1, \mathsf{pk}_1)$ respectively. These calls also fix the salt generated in $\mathsf{ARKG.DerivePK}$ such that $ck = 1$, ensuring both key pairs remain unchanged in this step. As an additional change, $\mathcal{B}$ simulates $Chall_{0,anon}$ using sk_0 as it would $SignU(\cdot, id_0, \cdot)$ and $Chall_{b,anon}$ using sk' to sign via its oracles $\mathcal{O}_{\mathsf{chSign1}}, \mathcal{O}_{\mathsf{chSign2}}$. Note that it does not matter that $\mathcal{B}$ cannot simulate $OpenSig$ for signatures against keys in

the classes $[\mathsf{pk}_0]$ or $[\mathsf{pk}_1]$ that it did not create itself, as by assumption $\mathcal{A}$ never makes such queries.

Then $\mathcal{B}$ sends this game instance to $\mathcal{A}$, which outputs a bit $b' \in \{0,1\}$ with $b' = 0$ indicating that the given instance represents Anon_5 and $b' = 1$ indicating Anon_4. Now if $b = 0$ then we have $\mathsf{sk}' = \mathsf{sk}_0$ matching the protocol of Anon_5, whereas if $b = 1$ then $\mathsf{sk}' = \mathsf{sk}_1$ matching the protocol of Anon_4. Thus by forwarding the bit b' to its own game, $\mathcal{B}$ wins in the case that $\mathcal{A}$ was able to identify the game instance correctly. Now suppose instead that $\mathcal{A}$ makes some query to $OpenSig$ depending on the underlying key pk of one of the challenge oracles. Then $\mathcal{A}$ can be used to construct another adversary $\mathcal{C}$, this time against the $\mathsf{suf\text{-}cma}$ game for SFPK.

Upon receiving the challenge $(\mathsf{pp}, \mathsf{pk}, \tau)$ from the $\mathsf{suf\text{-}cma}$ game, $\mathcal{C}$ generates the group keys for Anon_4 by calling $\mathsf{GKeyGen}(\mathsf{pp})$. In the choose phase $\mathcal{C}$ simulates the oracles for $\mathcal{A}$ according to the normal protocol of DGS-VLR except that for $Chall_{b,anon}$ and $Chall_{1-b,anon}$ it uses its signing oracle $\mathcal{O}_{\mathsf{unfSign2}}$ with a random salt r, ignoring the user keys $\mathsf{upk}_i, \mathsf{upk}_j$.

When $\mathcal{A}$ makes its query to either $Chall_{b,anon}$ or $Chall_{1-b,anon}$, $\mathcal{C}$ terminates the simulation of Anon_4, extracts the SFPK message-signature pair from the query and submits this to its own game. Because $OpenSig$ would not normally reject this query, it must be distinct from any of the outputs of $Chall_{b,anon}$ or $Chall_{1-b,anon}$ i.e. from the outputs of $\mathcal{O}_{\mathsf{Sign2}}$. Furthermore, Verify must accept the query and in particular $\mathsf{SFPK.Verify}$ must accept the extracted message-signature pair so that $\mathcal{C}$ wins the $\mathsf{suf\text{-}cma}$ game.

Combining the above results from both possible cases for the behaviour of $\mathcal{A}$ arrives at the following.

$$|\varepsilon_4^{\mathsf{anon}} - \varepsilon_5^{\mathsf{anon}}| \leqslant \mathsf{Adv}^{\mathsf{ch}}_{\mathcal{B},\mathsf{SFPK}}(\lambda) + \mathsf{Adv}^{\mathsf{suf\text{-}cma}}_{\mathcal{C},\mathsf{SFPK}}(\lambda)$$

Finally, in Anon_5 both signatures are sampled identically so it is statistically impossible to distinguish them. Furthermore, the NIZK simulator does not depend on the value of b since it is not given a witness. Therefore $\mathcal{A}$ has a probability of exactly $\frac{1}{2}$ of winning Anon_5.

$$\varepsilon_5^{\mathsf{anon}} = 0$$

Now tracing back the changes in advantage recovers the theorem statement. □

Theorem 2 (Traceability)**.** *The dynamic group signature scheme* DGS-VLR *defined in Figure 2 achieves the traceability security property if* NIZK *is sound and* DS *is* $\mathsf{euf\text{-}cma}$ *secure. That is, an adversary* $\mathcal{A}$ *wins the traceability game defined in Definition 7 with advantage:*

$$\mathsf{Adv}^{\mathsf{trace}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda) \leqslant \mathsf{Adv}^{\mathsf{sound}}_{\mathcal{B},\mathsf{NIZK}}(\lambda) + \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{DS}}(\lambda).$$

Proof. We consider the following experiments:

Trace_1 Original traceability game.

Trace_2 Add the requirement that $\mathsf{info}^* \in IL$, where IL is the list of all values for info output on some adversarial query to $IssueU, ActivateU$ or $RevokeU$ oracles.

We start by defining $\varepsilon_1^{trace} := \mathsf{Adv}^{\mathsf{trace}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda)$. Consider an adversary $\mathcal{A}$ that wins Trace_1 but not Trace_2. Then $\mathcal{A}$ must output some $\mathsf{info}^* := (pkl^*, \mathsf{RL}^*, \sigma_{\mathsf{info}^*})) \notin IL$ such that $\mathsf{DS.Verify}(\mathsf{gpk}, (pkl^*, \mathsf{RL}^*), \sigma_{\mathsf{info}^*})$ accepts. Consider further an adversary $\mathcal{B}$ against the $\mathsf{euf\text{-}cma}$ game for DS that simulates Trace_1 for $\mathcal{A}$ using its $\mathsf{DS.Sign}$ oracle to generate each σ_{info}. By extracting σ_{info} from each $\mathsf{info} \in IL$ and submitting this list along with σ_{info^*}, $\mathcal{B}$ wins its own game.

$$\left|\varepsilon_1^{\mathsf{trace}} - \varepsilon_2^{\mathsf{trace}}\right| \leqslant \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{DS}}(\lambda)$$

We now show that an adversary $\mathcal{A}$ against Trace_2 may be used to construct an adversary $\mathcal{B}$ against the soundness of NIZK. This adversary first executes the trace game as described in Figure 1 against $\mathcal{A}$. After some time, it receives $\mathsf{info}^* := (pkl^*, \mathsf{RL}^*, \sigma_{\mathsf{info}^*})$ and a message-signature pair (m^*, Σ^*) with $\Sigma^* = (\mathsf{pk}^*, \sigma^*, \Pi^*)$ from the adversary. Then $\mathcal{B}$ simply extracts the proof Π^* and submits it to the soundness game for NIZK. We show that if $\mathcal{A}$ wins the traceability game, then $\mathcal{B}$ wins the soundness game.

If $\mathcal{A}$ wins its experiment, then the adversary output (m^*, Σ^*) must open to some $(id, (\tau_{\mathsf{pk}}, \tau_{\mathsf{cred}}))$ where $\mathsf{Judge}(\mathsf{gpk}, \mathsf{info}, \mathsf{REG}, id, (\tau_{\mathsf{pk}}, \tau_{\mathsf{cred}}), m^*, \Sigma^*) = 0$. Furthermore, we must have $\mathsf{Verify}(\mathsf{gpk}, \mathsf{info}^*, m^*, \Sigma^*) = 1$ and in particular the final condition that the proof Π^* verifies in line 9 of Verify must hold. Then $\mathcal{B}$ wins except if the relation $\exists \mathsf{pk} \in pkl^* : \mathsf{pk}^* \in [\mathsf{pk}]$ holds. Now since $\mathsf{info}^* \in IL$ and elements of pkl are never removed, this pk must lie in pkl. But then the condition $\mathsf{SFPK.ChkRep}(\tau_{\mathsf{pk}}, \mathsf{pk}^*) = 1$ on line 5 of Open must be met on some iteration i of the **for** loop containing it during $\mathcal{A}$'s experiment. We now argue this cannot be true.

Suppose that $\mathsf{SFPK.ChkRep}(\tau_{\mathsf{pk}_i}, \mathsf{pk}^*) = 1$ for $i \in \{1, \ldots, n\}$. Then line 6 returns $\tau_{\mathsf{pk}} = \tau_{\mathsf{pk}_i}$, causing the condition $\mathsf{SFPK.ChkRep}(\tau_{\mathsf{pk}}, \mathsf{pk}^*) = 1$ on line 9 of Judge to hold. Furthermore, from line 4 of Open it can be seen that $(id, \tau_{\mathsf{pk}}, \tau_{\mathsf{cred}}) \in okl$. As pkl and okl are only ever changed on lines 6 and 7 of AddUser respectively, there must also exist some corresponding $(\mathsf{pk}, \mathsf{cred}) \in pkl$. Now lines 4 and 5 of AddUser show that by the correctness of ARKG and the compatibility of ARKG with SFPK the condition $\mathsf{ARKG.ChkLink}(\mathsf{REG}[id], \mathsf{pk}, \mathsf{cred}, \tau_{\mathsf{cred}}) = 1$ on line 7 of Judge holds. Similarly, line 5 of AddUser together with correctness of SFPK implies that the condition $\mathsf{SFPK.ChkRep}(\tau_{\mathsf{pk}}, \mathsf{pk}) = 1$ on line 8 of Judge holds. But then with all conditions on lines 7 to 9 of Judge fulfilled, on line 11 it returns 1 thus contradicting that $\mathcal{A}$ won.

$$\varepsilon_2^{\mathsf{trace}} \leqslant \mathsf{Adv}^{\mathsf{sound}}_{\mathcal{B},\mathsf{NIZK}}(\lambda)$$

Now tracing back the changes in advantage recovers the theorem statement. □

Theorem 3 (Non-Frameability)**.** *The dynamic group signature scheme* DGS-VLR *defined in Figure 2 achieves the non-frameability security property if* SFPK *is* $\mathsf{euf\text{-}cma}$ *secure and* ARKG *has sk-security. That is, an adversary* $\mathcal{A}$

wins the non-frameability game defined in Definition 8 with advantage

$$\mathsf{Adv}^{\text{frame}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda) \leqslant n \cdot \big(\mathsf{Adv}^{\text{euf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda) + \mathsf{Adv}^{\text{sks}}_{\mathcal{A},\mathsf{ARKG}}(\lambda)\big).$$

Proof. We introduce the following sequence of games.

Frame_1 Original non-frameability game.

Frame_2 Extend line 6 as follows. Choose random $i \leftarrow\!\!\$\ \{1,\ldots,n\}$, if $id \neq i$ then abort.

Frame_3 We replace $\mathsf{ARKG.ChkLink}(\mathsf{upk},\mathsf{pk},\mathsf{cred},\tau_{\mathsf{cred}})$ on line 5 of Judge with $\mathsf{ARKG.ChkKeys}(\mathsf{pp},\mathsf{sk},\mathsf{pk})$ where $\mathsf{sk} \leftarrow \mathsf{ARKG.DeriveSK}(\mathsf{pp},\mathsf{usk}_i,\mathsf{cred})$.

Frame_4 In line 6 the adversary also outputs a secret key sk^* alongside a message and signature, i.e. it returns $(\mathsf{sk}^*, m, \sigma)$. Furthermore, Judge gains the additional check $\mathsf{ARKG.ChkKeys}(\mathsf{pp},\mathsf{sk}^*,\mathsf{pk})$ for the submitted key.

Frame_5 User i now also calls $\mathsf{SFPK.TrapGen}(\mathsf{upk}_i)$ when generating keys, obtaining a trapdoor τ_i and replacing upk_i with pk_i. The adversary receives τ_i and Judge now replaces the two $\mathsf{SFPK.ChkRep}$ checks on lines 6 and 7 with the single check $\mathsf{SFPK.ChkRep}(\tau_i,\mathsf{pk}')$. Additionally, on line 5 we further replace $\mathsf{ARKG.ChkKeys}(\mathsf{pp},\mathsf{sk},\mathsf{pk})$ with $\mathsf{ARKG.ChkKeys}(\mathsf{pp},\mathsf{usk}_i,\mathsf{pk})$. The value of cred is ignored.

Frame_6 User i now uses $\mathsf{SFPK.TrapGen}$ immediately instead of calling $\mathsf{ARKG.KeyGen}$ and $\mathsf{SFPK.TrapGen}$ sequentially, effectively replacing $\mathsf{ARKG.KeyGen}$ with $\mathsf{SFPK.KeyGen}$.

We denote by ε_k^{frame} the probability $\Pr[\mathsf{Frame}_k]$ where $k \in \{1,\ldots,6\}$, where

$$\varepsilon_1^{\text{frame}} := \mathsf{Adv}^{\text{frame}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda).$$

The probability that the adversary chooses the required valued for i to avoid Frame_2 being aborted is $1/n$, so its advantage is adjusted by the same factor.

$$\varepsilon_2^{\text{frame}} = \frac{1}{n}\varepsilon_1^{\text{frame}}$$

In Frame_2 if Judge accepts then in particular we must have that $\mathsf{ARKG.ChkLink}(\mathsf{upk},\mathsf{pk},\mathsf{cred},\tau_{\mathsf{cred}})$ accepts. Hence by the linking soundness property $\mathsf{ARKG.ChkKeys}(\mathsf{pp},\mathsf{sk},\mathsf{pk})$ accepts, so Judge will also accept in Frame_3. If it did not, the adversary would be able to win the Linking Soundness experiment, defined in Definition 2, by submitting the tuple $(\mathsf{pk},\mathsf{cred},\tau_{\mathsf{cred}})$ on line two of the definition. The embedding of the challenge parameters is straight forward; all ARKG related parameters are inherited from the Linking Soundness experiment and instantiated in the DGS-VLR , and all additional parameters (such as for SFPK) are created locally by the soundness adversary. Therefore we conclude:

$$\left|\varepsilon_2^{\text{frame}} - \varepsilon_3^{\text{frame}}\right| \leqslant \mathsf{Adv}^{\text{l-sound}}_{\mathcal{A},\mathsf{ARKG}}(\lambda)$$

Consider the following two mutually exclusive classifications for the adversary's behaviour in Frame_3. One possibility is that $\mathcal{A}$ views $\mathsf{ARKG.DerivePK}$ as a pseudo-random function with respect to the relationship between $\hat{\mathsf{pk}}$ and cred. In this case $\mathcal{A}$ does not distinguish the distribution of $\hat{\mathsf{pk}}$ under any choice of cred from that of upk and hence it would perform identically in Frame_5. The alternative is that $\mathcal{A}$ exploits $\mathsf{ARKG.DerivePK}$ by choosing cred to herd the output for $\hat{\mathsf{pk}}$ to a particular value, say to match some known key pair $(\mathsf{sk}^*,\mathsf{pk}^*) \leftarrow \mathsf{ARKG.KeyGen}$. Then sk^* may be extracted along with the usual forgery m,Σ to win Frame_4. The overall advantage for Frame_3 is therefore bounded according to the total advantage summed across the two branching games Frame_4 and Frame_5.

$$\varepsilon_3^{\text{frame}} \leqslant \varepsilon_4^{\text{frame}} + \varepsilon_5^{\text{frame}}$$

Let's focus on the latter case and consider such an adversary $\mathcal{A}$ against Frame_4. An adversary $\mathcal{B}$ against the sk-security game may be constructed from $\mathcal{A}$ as follows.

Upon receiving the challenge $(\mathsf{sk},\mathsf{pk})$ from the sk-security game, $\mathcal{B}$ forwards the public parameters pp to $\mathcal{A}_0$. In the play phase $\mathcal{B}$ simulates the oracles for $\mathcal{A}_1$ according to the normal protocol of DGS-VLR except that on the i-th call to $HonestU$, $\mathcal{B}$ takes $(\mathsf{usk}_i,\mathsf{upk}_i) \leftarrow (\mathsf{sk},\mathsf{pk})$.

At the end of the play phase $\mathcal{A}_1$ outputs a guess sk^*. In the case that Judge accepts $\mathcal{A}$'s guess, $\mathcal{B}$ additionally extracts $(\mathsf{pk}^*,\mathsf{cred}^*)$ from the accepting iteration of the **for** loop in Judge. By forwarding $(\mathsf{sk}^*,\mathsf{pk}^*,\mathsf{cred}^*)$ to the sk-security game, $\mathcal{B}$ is also able to win, yielding:

$$\varepsilon_4^{\text{frame}} \leqslant \mathsf{Adv}^{\text{sks}}_{\mathcal{A},\mathsf{ARKG}}(\lambda).$$

Now return to the first case and consider an adversary against Frame_5. Given that the ARKG keys are compatible with the SFPK instantiation (as required by correctness), the distribution of each $(\mathsf{pk}_{id},\mathsf{usk}_{id})$ in Frame_5 is identical to the distribution of each $(\mathsf{upk}_{id},\mathsf{usk}_{id})$ in Frame_6 making the two games indistinguishable.

$$\varepsilon_5^{\text{frame}} = \varepsilon_6^{\text{frame}}$$

Now let $\mathcal{A}$ be an adversary against Frame_6, we construct $\mathcal{B}$ against the euf-cma experiment for SFPK as follows.

Upon receiving the challenge $(\cdot,\mathsf{pk},\tau)$ from the euf-cma experiment, $\mathcal{B}$ simulates Frame_6 for $\mathcal{A}$ as normal except it replaces the key generation in the i-th call to $HonestU$ with the challenge $(\cdot,\mathsf{pk},\tau)$. It simulates signing for user i using its oracles $\mathcal{O}_{\mathsf{unfSign1}},\mathcal{O}_{\mathsf{unfSign2}}$ with uniformly random r. When $\mathcal{A}$ outputs its forgery $(m^*,\sigma^*,id^*,\tau^*)$, $\mathcal{B}$ submits (m^*,σ^*) to its own experiment and wins in the case that $\mathcal{A}$ wins Frame_6.

$$\varepsilon_6^{\text{frame}} \leqslant \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{A},\mathsf{SFPK}}(\lambda)$$

Now tracing back the changes in advantage recovers the theorem statement.

□

Theorem 4 (Join Privacy)**.** *The dynamic group signature scheme* DGS-VLR *defined in Figure 2 achieves the join-privacy security property if* ARKG *has pk-unlinkability and* DS *is* euf-cma *secure. That is, an adversary $\mathcal{A}$ wins the join-privacy game defined in Definition 9 with advantage:*

$$\mathsf{Adv}^{\text{join}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda) \leqslant n \cdot \mathsf{Adv}^{\text{pku}}_{\mathcal{A},\mathsf{ARKG}}(\lambda) + \mathsf{Adv}^{\text{euf-cma}}_{\mathcal{A},\mathsf{DS}}(\lambda).$$

Proof. Let us consider the series of games be defined as follows:

Join_1 Original join-privacy game.

Join_2 Modify the $SignU$ oracle such that instead of checking $\mathsf{DS.Verify}(\mathsf{gpk}, \sigma_{\mathsf{info}})$, check that the value of info given matches some previous output of either $IssueU$ or $RevokeU$ tracked by a list IL.

Join_3 Extend line 5 as follows. Choose random $i \leftarrow\!\!\$ \{1, \ldots, n\}$, if $id_b \neq i$ then abort.

Join_4 When AddUser is called with id_b on line 7, instead of running $\mathsf{ARKG.DerivePK}(\mathsf{upk})$ a fresh pair of user keys $(\mathsf{pk}^*, \mathsf{sk}^*)$ is generated using ARKG.KeyGen along with a random credential cred^*. Additionally, in line 8 the challenge oracle $Chall_{b,join}$ is changed to use sk^* instead of the algorithm $\mathsf{ARKG.DeriveSK}(\mathsf{sk}[id_b])$.

We denote by ε_k^{join} the probability $\left|\Pr[\mathsf{Join_k}] - \frac{1}{2}\right|$ where $k \in [4]$, and have

$$\varepsilon_1^{\mathsf{join}} \;:=\; \mathsf{Adv}^{\mathsf{join}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda).$$

Consider an adversary $\mathcal{A}$ that wins Join_1 but not Join_2. By the correctness of DS, every $\mathsf{info} \in IL$ will contain σ_{info} such that $\mathsf{DS.Verify}(\mathsf{gpk}, \sigma_{\mathsf{info}}) = 1$ so $SignU$ will behave identically in Join_1 and Join_2 for such queries. But $\mathcal{A}$ achieves a different outcome in Join_2 than in Join_1 and hence it must make at least one query to $SignU$ with some input value $\mathsf{info} \notin IL$. Furthermore, in the case that DS.Verify rejects some σ_{info}, the output of $SignU$ is simply $(\bot, \bot, \bot)$ and does not depend on the input id. In particular, when $SignU$ is called via $Chall_{b,join}$ its output does not depend on the challenge bit b and hence $\mathcal{A}$ learns nothing. Therefore $\mathcal{A}$ must make at least one query to $SignU$ with input $\mathsf{info} \notin IL$ such that $\mathsf{DS.Verify}(\mathsf{gpk}, (pkl, \mathsf{RL}), \sigma_{\mathsf{info}}) = 1$.

We construct an adversary $\mathcal{B}$ against the euf-cma game for DS that simulates Join_1 for $\mathcal{A}$ while also storing IL in its state. To simulate $IssueU$ and $RevokeU$ $\mathcal{B}$ uses the euf-cma signing oracle. For each query to $SignU$ $\mathcal{B}$ checks whether $\mathsf{info} \notin IL$ and $\mathsf{DS.Verify}(\mathsf{gpk}, (pkl, \mathsf{RL}), \sigma_{\mathsf{info}}) = 1$. If such a query is made $\mathcal{B}$ terminates the simulation of Join_1 and adds info to IL. Once Join_1 terminates as above, or otherwise, $\mathcal{B}$ submits IL to the euf-cma game, winning in the former case. This is precisely the case that $\mathcal{A}$ wins the simulated instance of Join_1 without winning the corresponding instance of Join_2.

$$\left|\varepsilon_1^{\mathsf{join}} - \varepsilon_2^{\mathsf{join}}\right| \;\leqslant\; \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{DS}}(\lambda)$$

The probability that the adversary chooses the required value for i to avoid Join_3 being aborted is $1/n$, so its advantage is adjusted by the same factor.

$$\varepsilon_3^{\mathsf{join}} \;=\; \frac{1}{n}\varepsilon_2^{\mathsf{join}}$$

An adversary $\mathcal{A}$ that can distinguish Join_3 from Join_4 can be used to construct an adversary $\mathcal{B}$ against the pk-unlinkability game as follows.

Upon receiving the challenge $(\mathsf{pp}, \mathsf{pk})$ from the pk-unlinkability game, $\mathcal{B}$ generates the group keys for the anonymity game with $\mathsf{GKeyGen}(\mathsf{pp})$. In the choose phase $\mathcal{B}$ simulates the oracles for $\mathcal{A}_0$ according to the normal protocol of DGS-VLR except that on the i-th call to $HonestU$, $\mathcal{B}$ returns (i, pk). After the choose phase $\mathcal{B}$ generates keys $(\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{cred})$ from the pk-unlink oracle O_{pku}, generates a trapdoor $(\mathsf{pk}^*, \tau_{\mathsf{pk}^*}) \leftarrow \mathsf{SFPK.TrapGen}(\mathsf{pk}_b)$ and sets $pkl \leftarrow pkl \cup \{(\mathsf{pk}^*, \mathsf{cred})\}, okl \leftarrow okl \cup \{(i, \tau_{\mathsf{pk}^*}, \bot)\}$ unless $\mathcal{A}_0$ has invoked $RevealU(i)$. Then $\mathcal{B}$ generates a signature $\sigma_{\mathsf{info}} \leftarrow \mathsf{DS.Sign}(ik, (pkl, \mathsf{RL}))$ and returns $(pkl, \mathsf{RL}, \sigma_{\mathsf{info}})$ to $\mathcal{A}$. Additionally $Chall_{b,join}^{\sim}$ is changed to use sk_b instead of $\mathsf{ARKG.DeriveSK}(\mathsf{sk}[id_b])$.

Then $\mathcal{B}$ sends this game instance to $\mathcal{A}$, which outputs a bit $b' \in \{0, 1\}$ with $b' = 0$ indicating that the given instance represents Join_3 and $b' = 1$ indicating Join_4. Now if $b = 0$ then the pk-unlink oracle $\mathcal{O}_{pku}$ provides keys derived from pk matching the protocol of Join_3, whereas if $b = 1$ then $\mathcal{O}_{pku}$ instead provides freshly generated keys matching the protocol of Join_4. Thus by forwarding the bit b' to its own game, $\mathcal{B}$ wins in the case that $\mathcal{A}$ was able to identify the game instance correctly.

$$\left|\varepsilon_4^{\mathsf{join}} - \varepsilon_3^{\mathsf{join}}\right| \;\leqslant\; \mathsf{Adv}^{\mathsf{pku}}_{\mathcal{A},\mathsf{ARKG}}(\lambda)$$

In the guess phase of Join_4, the value of info does not depend on id_b since neither pk^* nor cred^* depends on id_b. Additionally, the functionality of the challenge oracle $Chall_{b,join}$ does not depend on id_b since it uses sk^* to sign. Therefore $\mathcal{A}$ has a probability of exactly $\frac{1}{2}$ of winning Join_4.

$$\varepsilon_4^{\mathsf{join}} \;=\; 0$$

Now tracing back the changes in advantage recovers the theorem statement. □

Theorem 5 (Leave Privacy)**.** *The dynamic group signature scheme* DGS-VLR *defined in Figure 2 achieves the leave-privacy security property if* ARKG *has pk-unlinkability and* DS *is* euf-cma *secure. That is, an adversary $\mathcal{A}$ wins the leave-privacy game defined in Definition 10 with advantage*

$$\begin{aligned}\mathsf{Adv}^{\mathsf{leave}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda) \;&\leqslant\; 2n(n-1)\cdot \mathsf{Adv}^{\mathsf{pku}}_{\mathcal{A},\mathsf{ARKG}}(\lambda) \\ &\quad + \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{DS}}(\lambda).\end{aligned}$$

Proof. Let the series of games be defined as follows:

Leave_1 Original leave-privacy game.

Leave_2 Modify the $SignU$ oracle such that instead of checking $\mathsf{DS.Verify}(\mathsf{gpk}, \sigma_{\mathsf{info}})$, check that the value of info given matches some previous output of either $IssueU$ or $RevokeU$ tracked by a list IL.

Leave_3 Extend line 5 as follows. Choose random $i, j \leftarrow\!\!\$ \{1, \ldots, n\}$, if $(id_0, id_1) \neq (i, j)$ then abort.

Leave_4 In the choose phase for $\mathcal{A}_0$, modify $ActivateU$ so that on input i, the $\mathsf{ARKG.DerivePK}$ call during the AddUser step is replaced with a fresh key generation $(\mathsf{pk}^*, \mathsf{sk}^*) \leftarrow$ ARKG.KeyGen and a random credential cred^*. Additionally, for queries with user identity i the $SignU$ oracle uses sk^* directly in place of the derived key $\mathsf{sk}' \leftarrow \mathsf{ARKG.DeriveSK}(\mathsf{sk}_i)$.

Leave_4 Perform the same change made in the previous game but now for both identity i and identity j.

We denote by $\varepsilon_k^{\mathsf{leave}}$ the probability $\left|\Pr[\mathsf{Leave_k}] - \frac{1}{2}\right|$ where $k \in [5]$ and

$$\varepsilon_1^{\mathsf{leave}} := \mathsf{Adv}^{\mathsf{leave}}_{\mathcal{A},\mathsf{DGS\text{-}VLR}}(\lambda).$$

Consider an adversary $\mathcal{A}$ that wins Leave_1 but not Leave_2. By the correctness of DS, every info $\in IL$ will contain σ_{info} such that $\mathsf{DS.Verify}(\mathsf{gpk}, \sigma_{\mathsf{info}}) = 1$ so $SignU$ will behave identically in Leave_1 and Leave_2 for such queries. But $\mathcal{A}$ achieves a different outcome in Leave_2 than in Leave_1 and hence it must make at least one query to $SignU$ with some input value info $\notin IL$. Furthermore, in the case that DS.Verify rejects some σ_{info}, the output of $SignU$ is simply $(\bot, \bot, \bot)t$ and does not depend on the input id. In particular, when $SignU$ is called via $Chall_{b,join}$ its output does not depend on the challenge bit b and hence $\mathcal{A}$ learns nothing. Therefore $\mathcal{A}$ must make at least one query to $SignU$ with input info $\notin IL$ such that $\mathsf{DS.Verify}(\mathsf{gpk}, (pkl, \mathsf{RL}), \sigma_{\mathsf{info}}) = 1$.

We construct an adversary $\mathcal{B}$ against the euf-cma game for DS that simulates Leave_1 for $\mathcal{A}$ while also storing IL in its state. In order to simulate $IssueU$ and $RevokeU$ $\mathcal{B}$ uses the euf-cma signing oracle. For each query to $SignU$ $\mathcal{B}$ checks whether info $\notin IL$ and $\mathsf{DS.Verify}(\mathsf{gpk}, (pkl, \mathsf{RL}), \sigma_{\mathsf{info}}) = 1$. If such a query is made $\mathcal{B}$ terminates the simulation of Leave_1 and adds info to IL. Once Leave_1 terminates as above, or otherwise, $\mathcal{B}$ submits IL to the euf-cma game, winning in the former case. Recall that this is precisely the case that $\mathcal{A}$ wins the simulated instance of Leave_1 without winning the corresponding instance of Leave_2.

$$\left|\varepsilon_1^{\mathsf{leave}} - \varepsilon_2^{\mathsf{leave}}\right| \leqslant \mathsf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathcal{A},\mathsf{DS}}(\lambda)$$

The probability that the adversary chooses the required values for i, j to avoid Leave_3 being aborted is $1/n(n-1)$, so its advantage is adjusted by the same factor.

$$\varepsilon_3^{\mathsf{leave}} = \frac{1}{n(n-1)}\varepsilon_2^{\mathsf{leave}}$$

An adversary $\mathcal{A}$ that can distinguish Leave_3 from Leave_4 can be used to construct an adversary $\mathcal{B}$ against the pk-unlinkability game as follows.

Upon receiving the challenge $(\mathsf{pp}, \mathsf{pk})$ from the pk-unlinkability game, $\mathcal{B}$ generates the group keys for $\mathsf{GKeyGen}(\mathsf{pp})$. In the choose phase $\mathcal{B}$ simulates the oracles for $\mathcal{A}_0$ according to the normal protocol of DGS-VLR except that on the i-th call to $HonestU$, $\mathcal{B}$ returns (i, pk). Furthermore, when $ActivateU$ is called on user i $\mathcal{B}$ replaces the ARKG.DerivePK step in DGS-VLR.AddUser and instead generates keys $(\mathsf{sk}_b, \mathsf{pk}_b, \mathsf{cred})$ from the pk-unlink oracle O_{pku}. Additionally the challenge oracle $Chall_{1-\widetilde{b},leave}$ is changed to use sk_b instead of $\mathsf{ARKG.DeriveSK}(\mathsf{sk}[i])$ in the case that $\widetilde{b} = 1 - b$ is chosen in line 5 of the leave game.

Then $\mathcal{B}$ sends this game instance to $\mathcal{A}$, which outputs a bit $b' \in \{0, 1\}$ with $b' = 0$ indicating that the given instance represents Leave_3 and $b' = 1$ indicating Leave_4. Now if $b = 0$ then the pk-unlink oracle $\mathcal{O}_{pku}$ provides keys derived from pk matching the protocol of Leave_3, whereas if $b = 1$ then O instead provides freshly generated keys matching the protocol of Leave_4. Thus by forwarding the bit b' to its own game, $\mathcal{B}$ wins in the case that $\mathcal{A}$ was able to identify the game instance correctly.

$$\left|\varepsilon_4^{\mathsf{leave}} - \varepsilon_3^{\mathsf{leave}}\right| \leqslant \mathsf{Adv}^{\mathsf{pku}}_{\mathcal{A},\mathsf{ARKG}}(\lambda)$$

By an analogous argument as for distinguishing between Leave_3 and Leave_4, the advantage for Leave_5 grows by a factor at most that of an adversary against pk-unlinkability of ARKG.

$$\left|\varepsilon_5^{\mathsf{leave}} - \varepsilon_4^{\mathsf{leave}}\right| \leqslant \mathsf{Adv}^{\mathsf{pku}}_{\mathcal{A},\mathsf{ARKG}}(\lambda)$$

In Leave_5 since both possible revocation tokens τ_0, τ_1 that the adversary can receive in the choose phase are identically distributed, it must have probability exactly $\frac{1}{2}$ of winning.

$$\varepsilon_5^{\mathsf{leave}} = 0$$

Now tracing back the changes in advantage recovers the theorem statement. □

6. Instantiation and Efficiency Analysis

We introduced a generic construction for dynamic group signature construction that mainly relies on ARKG and SFPK. In this section we ensure that our instantiations of those primitives are compatible, and exhibit the necessary trapdoors. We present an adapted version of ARKG from Frymann et al. [26] with trapdoors, we prove in Lemma 1 that it satisfies our new linking soundness property. We also utilise the SFPK scheme from Backes et al. [11]. Both constructions are based on bilinear maps, and we formally present them in Figures 3 and 4.

For the digital signature needed to sign on two dynamic lists: pkl and RL, we rely on the standard signature ECDSA [30] and hash function SHA-3 [31]. To implement this scheme, we hash together the lists as $h_2 = \mathsf{H}_2(pkl, \mathsf{RL})$ and sign on h_2. This implicitly assumes collision resistance of our chosen hash function. Finally, we use HMAC-SHA256 [32] for the internal MAC used in ARKG.

Lemma 1 (Linking Soundness of ARKG)**.** *The* ARKG *protocol shown in Figure 3 with* $\mathsf{pp} = ((\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, q), \mathsf{H}, \mathsf{MAC}, \mathsf{KDF}_1, \mathsf{KDF}_2)$ *has the linking soundness property.*

Proof. Let $(s, S) \leftarrow\!\!\$\ \mathsf{ARKG.KeyGen}(\mathsf{pp})$ and consider some $\mathsf{cred} := (E, aux, \mu)$. Suppose $\mathsf{ChkLink}(S, X, \mathsf{cred}, e) = 1$, then we have the following.

$$\begin{aligned}\mu &= \mathsf{MAC}(\mathsf{KDF}_2(S^e), (E, \mathsf{aux}))\\ X &= (g_1^{\mathsf{KDF}_1(S^e)}, g_2^{\mathsf{KDF}_1(S^e)}) \cdot S\\ E &= (g_1^e, g_2^e)\end{aligned}$$

Relying on $S^e = E^s$, we have: $\mu = \mathsf{MAC}(\mathsf{KDF}_2(E^s), (E, \mathsf{aux}))$. Then for $x \leftarrow$

$\mathsf{Setup}(\lambda)$

$\mathsf{KDF}_1, \mathsf{KDF}_2 \leftarrow\$ \mathsf{KDF}.Gen(\lambda)$
$fp \leftarrow (\mathsf{MAC}, \mathsf{KDF}_1, \mathsf{KDF}_2)$
$pp \leftarrow (gp, fp)$
return pp

$\mathsf{KeyGen}(\mathsf{pp})$

$s \leftarrow\$ \mathbb{Z}_q^*;\ S \leftarrow (g_1^s, g_2^s)$
return (s, S)

$\mathsf{ChkKeys}(x, X)$

$X' = (g_1^x, g_2^x)$
return $X \stackrel{?}{=} X'$

$\mathsf{DerivePK}(S, aux)$

parse $S = (S_1, S_2)$
$(e, E) \leftarrow KeyGen(\mathsf{pp})$
$ck \leftarrow \mathsf{KDF}_1((S_1^e, S_2^e))$
$mk \leftarrow \mathsf{KDF}_2((S_1^e, S_2^e))$
$X \leftarrow (g_1^{ck} \cdot S_1, g_2^{ck} \cdot S_2)$
$\mu \leftarrow \mathsf{MAC}(mk, (E, aux))$
$cred \leftarrow (E, aux, \mu)$
return $(X, cred, e)$

$\mathsf{DeriveSK}(s, cred)$

parse $cred = (E, aux, \mu)$
parse $E = (E_1, E_2)$
$ck \leftarrow \mathsf{KDF}_1((E_1^s, E_2^s))$
$mk \leftarrow \mathsf{KDF}_2((E_1^s, E_2^s))$
$x \leftarrow ck + s$
$\mu' \leftarrow \mathsf{MAC}(mk, (E, aux))$
if $\mu \neq \mu'$ **then return** $\perp$
return x

$\mathsf{ChkLink}(S, X, cred, e)$

parse $S = (S_1, S_2)$
parse $cred = (E, aux, \mu)$
$ck \leftarrow \mathsf{KDF}_1((S_1^e, S_2^e))$
$mk \leftarrow \mathsf{KDF}_2((S_1^e, S_2^e))$
$X' \leftarrow (g_1^{ck} \cdot S_1, g_2^{ck} \cdot S_2)$
$\mu' \leftarrow \mathsf{MAC}(mk, (E, aux))$
return $\mu \stackrel{?}{=} \mu' \ \wedge\ X \stackrel{?}{=} X' \ \wedge\ \mathsf{ChkKeys}(E, e)$

Figure 3. ARKG algorithms with proof of derivation adapted from Frymann et al. [20] our instantiation.

$\mathsf{DeriveSK}(s, \mathsf{cred})$ we have $x = \mathsf{KDF}_1(E^s) + s$ and hence

$$\begin{aligned} X &= \left(g_1^{\mathsf{KDF}_1(S^e)},\ g_2^{\mathsf{KDF}_1(S^e)}\right) \cdot S \\ &= \left(g_1^{\mathsf{KDF}_1(S^e)},\ g_2^{\mathsf{KDF}_1(S^e)}\right) \cdot (g_1^s,\ g_2^s) \\ &= \left(g_1^{\mathsf{KDF}_1(S^e)+s},\ g_2^{\mathsf{KDF}_1(S^e)+s}\right) \\ &= \left(g_1^{\mathsf{KDF}_1(E^s)+s},\ g_2^{\mathsf{KDF}_1(E^s)+s}\right) \\ &= (g_1^x,\ g_2^x)\,. \end{aligned}$$

Therefore $\mathsf{ChkKeys}(x, X) = 1$ as required. □

We now discuss the adjustments required for the SFPK scheme [11] to be compatible with our DGS-VLR construction. In particular, we modify the SFPK key generation algorithm TrapGen to take as input an existing public key primitive X. It outputs a public key pk alongside a trapdoor τ as before, but it no longer outputs a secret key sk. Internally, it still samples the first two secrets a, b as in [11] but no longer samples x, parsing $\hat{\mathsf{pk}} = X$ to obtain pre-computed values for g_1^x, g_2^x instead. This change shifts the responsibility of trapdoor generation away from the signer, deviating slightly from the model of Backes et al. The CRS generation for the vector Y is separated from the Setup algorithm to highlight the requirement for this to be implemented by a trusted party.

6.1. Instantiating the Zero-Knowledge Proof

For generality, we have presented our construction that makes use of a zero-knowledge proof to link together a derived key, admitting a user to the group, and the signing key, which has been randomised within an equivalence class. To instantiate this, we consider a zero-knowledge proof that proves the relation $\mathsf{pk}' = \mathsf{ChgPK}(\mathsf{pk}, r) \wedge \mathsf{pk} \in pkl$. The first equation can be proven using Groth-Sahai proofs [33], which would consist of proving 3 multi-scalar multiplication equations in $\mathbb{G}_1$ and the latter will be a set membership proof that the commitment contains a pk in group member list pkl, for this we utilise the OR proofs detailed in Ràfols [24].

However, this incurs significant cost from an efficiency point of view, namely the signature size is dependent on the number of group members. This is unsatisfactory so instead observe that Structure-Preserving Signatures on Equivalence Classes (SPS-EQ) [22] can be adopted with keys derived from ARKG. The signing algorithm of SPS-EQ defines an equivalence relation over the message space. It allows a signer to sign on one representative of an equivalence class to create a signature for the whole class. The signature can then be changed without the knowledge of the signing key to a different representative. SPS-EQ signature have been used in existing dynamic group signatures [11], [7], however we note that its function in our scheme does differ to these works. Namely, the group manager creates SPS-EQ keys and outputs the public key as part of $gmpk$. Then, during AddUser, the group manager creates an SPS-EQ signature over pk and adds this to the group information info. During signing, the signer forgoes the proof Π, and instead outputs a changed signature that verifies the randomised key pk'. Intuitively, a verifier is assured that the member must be a valid group member because the unforgeability property of SPS-EQ ensures that, without knowledge of the secret key, the changed signature must have been changed from an existing valid SPS-EQ signature.

6.2. Efficiency Analysis

We now present the size of the keys, signature and revocation list in Table 2. We focus on the SPS-EQ certificate approach due to its favourable signature size. Overall, our signature consists of a DS signature (512bits), a public key pk' $(1\mathbb{G}_1 + 1\mathbb{G}_2)$ and a SPS-EQ signature $(10\mathbb{G}_1 + 4\mathbb{G}_2)$, thus our total signature size is $11\mathbb{G}_1 + 5\mathbb{G}_2 + 512\texttt{bits}$. The public key consists of 3 elements in $\mathbb{G}_1$, whereas our secret only consists of 1 element in $\mathbb{Z}_q^*$. Our revocation list stores 1 element from $\mathbb{G}_2$ and 2 from $\mathbb{Z}_q^*$ per revoked user.

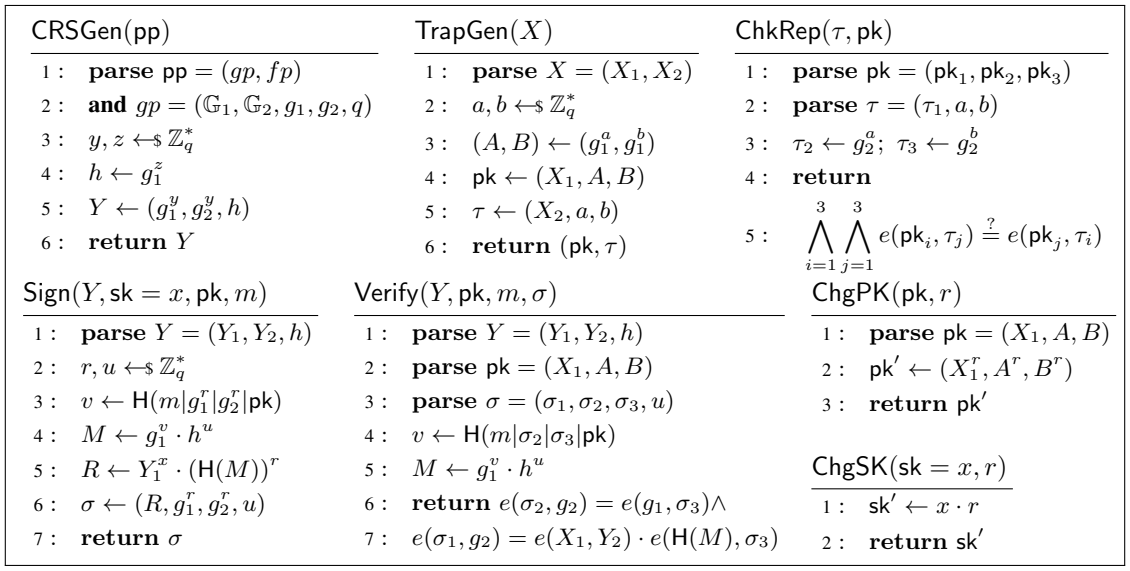

Figure 4. SFPK algorithms adapted for pre-generated key pairs $(\mathsf{sk}, \mathsf{pk}) = (x, X)$, where H is a collision-resistant hash function with image $\mathbb{G}_1$.

Table 2. COMPARISON AGAINST EXISTING DYNAMIC GROUP SIGNATURES. WE USE pk TO DENOTE THE USE OF THE USER PUBLIC KEY, usk FOR THEIR SECRET KEY, Σ FOR THE SIZE OF THE SIGNATURE AND RL FOR THE REVOCATION LIST, WHERE r IS THE NUMBER OF ENTRIES IN THE LIST. REVOKE SHOWS ASYMPTOTIC RUNTIME FOR REVOKING r OUT OF n GROUP MEMBERS. ZK PROOF SIZE[1] COMPUTED USING [24].

Scheme	pk	usk	Σ	RL	Revoke
Backes et al. [7]	$2\mathbb{G}_1$	$3\mathbb{G}_1$	$28\mathbb{G}_1 + 15\mathbb{G}_2 + 512\texttt{bits}$	N/A	$\mathcal{O}(n)$
DGS-VLR w/ NIZK	$3\mathbb{G}_1$	$1\mathbb{Z}_q^*$	$(4n+21)\mathbb{G}_1 + 19\mathbb{G}_2 + 512\texttt{bits}$	$r(1\mathbb{G}_2 + 2\mathbb{Z}_q^*)$	$\mathcal{O}(1)$
DGS-VLR w/ SPS-EQ	$3\mathbb{G}_1$	$1\mathbb{Z}_q^*$	$11\mathbb{G}_1 + 5\mathbb{G}_2 + 512\texttt{bits}$	$r(1\mathbb{G}_2 + 2\mathbb{Z}_q^*)$	$\mathcal{O}(1)$

Compared to Backes et al. [7], we achieve the smaller signature size despite use of the same underlying use of the SFPK scheme. This is due to the fact that our VLR approach means we can reduce the complexity of the relation for the proof that comprises part of the signature. However, our main advantage comes from the fact we have dropped the notion of epochs to manage revocation and joining within the scheme. In particular, to process a group update, our scheme publishes the group information info which has size $512(n+1)\texttt{bits} + 3n\mathbb{G}_1 + (3n+r)\mathbb{G}_2 + 2r\mathbb{Z}_q^*$, where n is the number of group members and r is the number of revoked users. As mentioned previously, this update is the same for all users and can be published globally. This compares to Backes et al. where each group member needs to regenerate its signing key, where user-specific computation consists of generating ciphertexts which dominate this phase. The size of the update is $10\mathbb{G}_1 + 14\mathbb{G}_2 + 512\texttt{bits}$[2] $+ 2\mathbb{G}$[3] per group member. Additionally, to perform revocation (i.e., calling Revoke) is more efficient for VLR schemes $\mathcal{O}(1)$ compared to schemes on secret credentials $\mathcal{O}(n)$ for n group members.

2. The original paper does not specify which digital signature the authors instantiate their scheme with, so we use ECDSA for fair comparison.

3. This term is derived from the author's use of the ElGamel encryption scheme, for which the underlying group can be independently chosen of $\mathbb{G}_1$ and $\mathbb{G}_2$, so we do not specify further.

7. Conclusion

In this paper we have proposed a Dynamic Group Signature scheme with a Verifier-Local Revocation – the first dynamic group signature scheme with membership privacy to benefit from the efficiency improvements for using a VLR approach. We were able to achieve this by adapting Asynchronous Remote Key Generation to also support a linking trapdoor, for which we defined a new security property *Linking Soundness*. When combined with a trapdoor from Signatures with Flexible Public Keys to form an extended two-stage tracing process, this allowed the latter trapdoor to serve as the revocation token in our generic construction without compromising membership privacy. We have instantiated our generic scheme with pairing-based building blocks to give a DGS-VLR that is competitive with the state-of-the-art from an efficiency point of view, but with improved functionality. By adopting a VLR approach, our scheme avoids reissuing keys during group updates and thus enables more efficient joining and leaving. Finally, when instantiated with SPS-EQ our signatures each consist of $11\mathbb{G}_1 + 5\mathbb{G}_2 + 512\texttt{bits}$, which is smaller than all other dynamic schemes to date (cf. Table 2).

Future Work. Whilst we have demonstrated favourable asymptotic performance, a concrete implementation would provide better metrics for real world use. Furthermore, post-quantum security guarantees would be an additional highly desirable property. This could be achieved through a lattice-

based implementation; in fact, lattice-based constructions have already been proposed for ARKG [27], [28], [29]. However, these implementations do not support the necessary trapdoor for *Linking Soundness*. Additionally, SPS-EQ from lattices is also an open problem, both of which would be required to extend our results to achieve PQ security.

Acknowledgments

C. C. Drăgan is partially supported by TrustVote – EPSRC grant EP/Y020529/1, AP4L - EPSRC grant EP/W032473/1, CONNECT - Horizon Europe Guarantee 10043730 and EU Horizon grants 101069688, REWIRE - Horizon Europe Guarantee 10043743 and EU Horizon grants 101070627.

References

[1] D. Chaum and E. van Heyst, "Group Signatures," in *Proceedings of EUROCRYPT 1991*, vol. 547 of *LNCS*, pp. 257–265, Springer, 1991.

[2] A. Lysyanskaya and Z. Ramzan, "Group blind digital signatures: A scalable solution to electronic cash," in *Financial Cryptography, Second International Conference, FC'98, Anguilla, British West Indies, February 23-25, 1998, Proceedings*, vol. 1465 of *LNCS*, pp. 184–197, Springer, 1998.

[3] S. Canard and J. Traoré, "On fair e-cash systems based on group signature schemes," in *Information Security and Privacy, 8th Australasian Conference, ACISP 2003, Wollongong, Australia, July 9-11, 2003, Proceedings*, vol. 2727 of *LNCS*, pp. 237–248, Springer, 2003.

[4] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of Group Signatures: Formal Definitions, Simplified Requirements, and a Construction Based on General Assumptions," in *Proceedings of EUROCRYPT 2003*, vol. 2656 of *LNCS*, pp. 614–629, Springer, 2003.

[5] M. Bellare, H. Shi, and C. Zhang, "Foundations of Group Signatures: The Case of Dynamic Groups," in *Proceedings of CT-RSA 2005*, vol. 3376 of *LNCS*, pp. 136–153, Springer, 2005.

[6] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth, "Foundations of Fully Dynamic Group Signatures," in *Proceedings of ACNS 2016*, vol. 9696 of *LNCS*, pp. 117–136, Springer, 2016.

[7] M. Backes, L. Hanzlik, and J. Schneider-Bensch, "Membership Privacy for Fully Dynamic Group Signatures," in *Proceedings of the CCS 2019*, pp. 2181–2198, ACM, 2019.

[8] "Docusign group signing." https://support.docusign.com/s/document-item?language=en_US&rsc_301&bundleId=gav1643676262430&topicId=zgn1578456447934.html. Accessed on 17th April 2025.

[9] B. Brecht, D. Therriault, A. Weimerskirch, W. Whyte, V. Kumar, T. Hehn, and R. Goudy, "A security credential management system for v2x communications," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, 02 2018.

[10] J. Petit, F. Schaub, M. Feiri, and F. Kargl, "Pseudonym schemes in vehicular networks: A survey," *Commun. Surveys Tuts.*, vol. 17, p. 228–255, Jan. 2015.

[11] M. Backes, L. Hanzlik, K. Kluczniak, and J. Schneider, "Signatures with Flexible Public Key: Introducing Equivalence Classes for Public Keys," in *Proceedings of Part II of ASIACRYPT 2018*, vol. 11273 of *LNCS*, pp. 405–434, Springer, 2018.

[12] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proceedings of CCS 2004*, pp. 168–177, ACM, 2004.

[13] P. Bichsel, J. Camenisch, G. Neven, N. P. Smart, and B. Warinschi, "Get Shorty via Group Signatures without Encryption," in *Proceedings of SCN 2010*, vol. 6280 of *LNCS*, pp. 381–398, Springer, 2010.

[14] A. Ishida, Y. Sakai, K. Emura, G. Hanaoka, and K. Tanaka, "Fully Anonymous Group Signature with Verifier-Local Revocation," in *Proceedings of SCN 2018*, vol. 11035 of *LNCS*, pp. 23–42, Springer, 2018.

[15] B. Libert, T. Peters, and M. Yung, "Scalable group signatures with revocation," in *Proceedings of EUROCRYPT 2012*, vol. 7237 of *LNCS*, pp. 609–627, Springer, 2012.

[16] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki, "Revocable Group Signature Schemes with Constant Costs for Signing and Verifying," in *Proceedings of PKC 2009*, vol. 5443 of *LNCS*, pp. 463–480, Springer, 2009.

[17] V. Kumar, H. Li, N. Luther, P. Asokan, J.-M. J. Park, K. Bian, M. B. H. Weiss, and T. Znati, "Direct anonymous attestation with efficient verifier-local revocation for subscription system," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, ASIACCS '18, p. 567–574, Association for Computing Machinery, 2018.

[18] Y. NAKAZAWA and T. NAKANISHI, "A strongly unlinkable group signature scheme with matching-based verifier-local revocation for privacy-enhancing crowdsensing," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E106.A, no. 12, pp. 1531–1543, 2023.

[19] K. Banti, F. Katsimpoura, M. Louta, and G. T. Karetsos, "Data quality in mobile crowd sensing systems: Challenges and perspectives," in *2018 9th International Conference on Information, Intelligence, Systems and Applications (IISA)*, pp. 1–8, 2018.

[20] N. Frymann, D. Gardham, F. Kiefer, E. Lundberg, M. Manulis, and D. Nilsson, "Asynchronous Remote Key Generation: An Analysis of Yubico's Proposal for W3C WebAuthn," in *Proceedings of CCS 2020*, pp. 939–954, ACM, 2020.

[21] J. Katz, *Digital Signatures*. Springer, 2010.

[22] G. Fuchsbauer and R. Gay, "Weakly secure equivalence-class signatures from standard assumptions," in *Public-Key Cryptography – PKC 2018*, pp. 153–183, Springer International Publishing, 2018.

[23] B. Libert, T. Peters, and M. Yung, "Group Signatures with Almost-for-Free Revocation," in *Proceedings of CRYPTO 2012*, vol. 7417 of *LNCS*, pp. 571–589, Springer, 2012.

[24] C. Ràfols, "Stretching groth-sahai: Nizk proofs of partial satisfiability," in *Theory of Cryptography*, pp. 247–276, Springer Berlin Heidelberg, 2015.

[25] N. Frymann, D. Gardham, and M. Manulis, "Unlinkable Delegation of WebAuthn Credentials," in *Proceedings of PART III of ESORICS 2022*, vol. 13556 of *LNCS*, pp. 125–144, Springer, 2022.

[26] N. Frymann, D. Gardham, M. Manulis, and H. Nartz, "Generalised Asynchronous Remote Key Generation for Pairing-Based Cryptosystems," in *Proceedings of Part I of ACNS 2023*, vol. 13905 of *LNCS*, pp. 394–421, Springer, 2023.

[27] N. Frymann, D. Gardham, and M. Manulis, "Asynchronous Remote Key Generation for Post-Quantum Cryptosystems from Lattices," in *Proceedings of IEEE EuroS&P 2023*, pp. 928–941, IEEE, 2023.

[28] D. Stebila and S. Wilson, "Quantum-safe account recovery for webauthn," in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2024, Singapore, July 1-5, 2024*, ACM, 2024.

[29] J. Brendel, S. Clermont, and M. Fischlin, "Post-quantum asynchronous remote key generation for fido2," in *ASIACRYPT 2024*, Springer-Verlag, 2024.

[30] D. Johnson, A. Menezes, and S. A. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)," *Int. J. Inf. Sec.*, vol. 1, no. 1, pp. 36–63, 2001.

[31] National Institute of Standards and Technology (NIST), "SHA-3 standard: Permutation-based hash and extendable-output functions," 2015.

[32] H. Krawczyk, M. Bellare, and R. Canetti, "Rfc2104: Hmac: Keyed-hashing for message authentication," 1997.

[33] J. Groth and A. Sahai, "Efficient non-interactive proof systems for bilinear groups," in *Advances in Cryptology – EUROCRYPT 2008*, pp. 415–432, Springer Berlin Heidelberg, 2008.

Appendix A. Building Blocks

Here we recall additional building blocks and their security properties.

The security experiments for SFPK are given in Figure 5. Note that the inclusion of σ^* in line 5 of the euf-cma experiment transforms it into the suf-cma (strong euf-cma) experiment.

The security experiments for ARKG are given in Figure 6. The presented definitions have been modified to capture the trapdoor τ as an additional output from algorithm DerivePK.

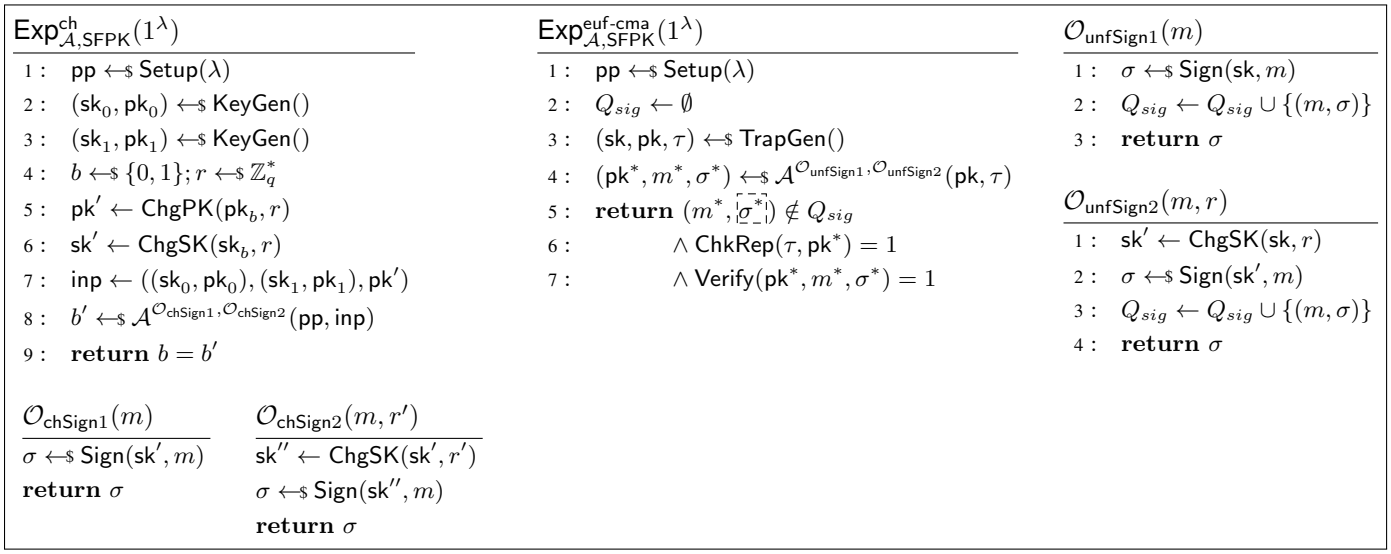

Figure 5. Experiments for SFPK class-hiding and (strong) unforgeability.

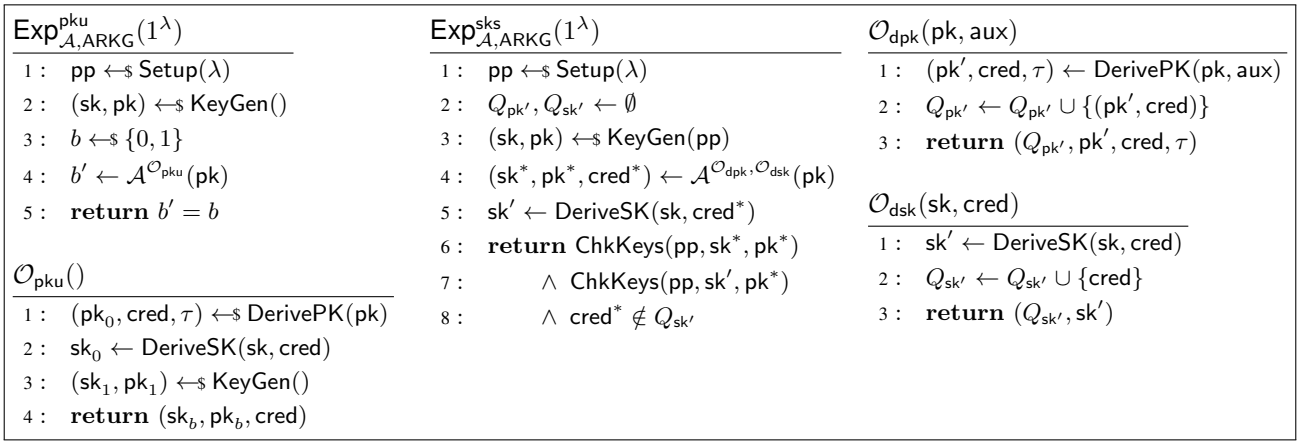

Figure 6. Experiments for ARKG pk-unlinkability and sk-secrecy.