

Practical Dynamic Group Signatures Without Knowledge Extractors

Hyoseung Kim* Olivier Sanders† Michel Abdalla‡ Jong Hwan Park§

Abstract

Dynamic group signature (DGS) allows a user to generate a signature on behalf of a group, while preserving anonymity. Although many existing DGS schemes have been proposed in the random oracle model for achieving efficiency, their security proofs require knowledge extractors that cause loose security reductions. In this paper, we first propose a new practical DGS scheme whose security can be proven without knowledge extractors in the random oracle model. Moreover, our scheme can also be proven in the strong security model where an adversary is allowed to generate group managers' keys maliciously. The efficiency of our scheme is comparable to existing secure DGS schemes in the random oracle model using knowledge extractors. The security of our scheme is based on a new complexity assumption that is obtained by generalizing the Pointcheval-Sanders (PS) assumption. Although our generalized PS (GPS) assumption is interactive, we prove that, under the symmetric discrete logarithm (SDL) assumption, the new GPS assumption holds in the algebraic group model.

Keywords: Algebraic Group Model, Group Signatures, Knowledge Extractors, PS assumption, Random Oracle Model, Subverted CRS.

1 Introduction

1.1 Related Works

Introduced by Chaum and Van Heyst [22] in 1991, group signature is a primitive which enables users to join a group managed by a so-called *issuer*. From this point on, these users become group members that are able to anonymously sign on behalf of the group. More specifically, a common verifier can check that a signature σ was issued by a group member thanks to the group public key but it cannot link σ back to a specific member. Without additional requirements, group signature would be trivial to achieve: the group manager could simply generate a signature key pair (sk, pk) for some standard digital signature scheme and distribute sk to each member. Such a system would obviously achieve perfect and irrevocable anonymity. However, such a strong level of anonymity may be undesirable in many scenarios as it is likely to encourage abuses that could tarnish the reputation of the group as a whole or even engage liability for each of its members.

It is therefore more reasonable to implement safeguards which, in the case of group signature schemes, take the form of a specific procedure called opening. This procedure, run by an appointed entity, the opener,

*Korea University, Seoul, Korea. Email: hyoseung.kim@korea.ac.kr.

†Orange Labs, Applied Crypto Group, Cesson-Sévigné, France. Email: olivier.sanders@orange.com.

‡DIENS, École normale supérieure, CNRS, PSL University, and INRIA, Paris, France. Email: michel.abdalla@ens.fr.

§Sangmyung University, Seoul, Korea. Email: jhpark@smu.ac.kr.

allows to identify the issuer of any group signature and thus constitutes a threat of anonymity revocation that should deter bad behaviour. One of its consequences is to invalidate the trivial construction we sketched above: each group member needs its own signing keys gsk . Over the years, the security model of group signature evolved to reach the notion of dynamic group signature DGS [8, 39] where members can join the group by generating their own signing keys. This has led [8] to define three security properties called *anonymity*, *traceability* and *non-frameability*. The former ensures that group signatures are anonymous, except for the opener. The latter two ensure balanced accountability for group members. Traceability states that any valid group signature σ can be linked back to a member i whereas non-frameability assures that i was indeed involved in the generation of σ .

Today, it is hard to overstate the importance of group signature. It can indeed be transformed very easily into several variants, such as DAA [15] and EPID [16], which are embedded in billions of chips [38, 48]. More generally, it has connections with most privacy preserving primitives, which naturally enhances any advances in the design of group signature. Any practical or theoretical improvement in the area of group signature is indeed likely to have similar consequences for many related primitives.

In this regard, it is no surprise that group signature is still a very active cryptographic topic 30 years after its introduction, with countless contributions (see, *e.g.* [11, 14, 23, 25, 26, 36, 40–42, 45]). Although all these constructions have their own features, they share a common idea. During enrolment, each new member receives some information from the group manager (usually a certificate, or a witness for some public accumulator for non-dynamic constructions) that will allow him to prove membership to the group. To retain anonymity, this information cannot be simply revealed in each signature, which naturally calls for non-interactive zero-knowledge proofs (NIZK). This leads us to distinguish two families of groups signatures, those proved in the standard model and those proved in the random oracle model (ROM). The former (*e.g.* [14, 36, 42]) are clearly better from the theoretical standpoint but they are rather inefficient in practice. This explains the popularity of the second family where the NIZK proofs is usually generated by applying the celebrated Fiat-Shamir transform [28] to a three-move zero-knowledge protocol, yielding very efficient signatures containing as few as 4 elements in [45]. However, despite their better efficiency, such NIZK proofs cause loose security reductions [46] because of the so-called *knowledge extractor* that rewinds an adversary and reprograms the random oracle to extract the witness of the proof, which is often omitted in the proof. Moreover, there is usually no rigorous security analysis when extraction is needed for many NIZK proofs although this case is known to be quite complex [10].

To the best of our knowledge, all existing DGS schemes that are secure in the ROM (including [11, 18, 25, 41, 45]) suffer from the loose security reduction inherent to such rewinding strategies. More precisely, knowledge extractors are required in traceability and non-frameability security proofs. Regarding traceability, this is due to the framework followed by most dynamic constructions where, in a joining protocol execution, a member's signing key gsk is usually generated as a signature (under the issuer's public key) on some committed values usk using appropriate schemes, such as [12, 20, 45]. As traceability is expected to rely on the unforgeability of these schemes, the security proof must usually extract each usk to query the corresponding signing oracles, which has the additional effect of disallowing concurrent join protocols. Moreover, one can show that an untraceable signature necessarily contains a forgery σ^* , which is valid only if one can extract the corresponding message usk^* . Extractors are thus also necessary for the NIZK contained in the group signature. To overcome this, Derler and Slamanig [26] recently proposed a DGS scheme based on a structure-preserving signature (SPS) on equivalence classes [37], entirely eliminating all knowledge extractors in the traceability proof. Very recently, Kim et al. [40] used a variant of the PS signature to construct a practical DGS scheme without knowledge extractors only in the joining protocol executions. However, these two constructions [26, 40] and all the others that are secure in the ROM still

require knowledge extractors when proving the non-frameability. Indeed, non-frameability usually relies on the knowledge of the secret value usk which embeds a discrete logarithm challenge during the security proof. Roughly speaking, a successful adversary against non-frameability must have produced a valid proof of usk which is then extracted through rewinding techniques.

Removing Knowledge Extractors. Several approaches, including straight-line extraction techniques such as extractable commitments [33], online extractors [29] and adaptive proofs of knowledge [10], have led to fairly inefficient DGS constructions in attempts to remove (rewinding) knowledge extractors from all security proofs in the ROM. Delerablée and Pointcheval [25] used the Paillier encryption [44] as an extractable commitment based on a common reference string (CRS). However, such a CRS model entails a critical issue in practice, aside from its complexity. During the security analysis, the CRS is generated along with a *trapdoor* (e.g., the decryption key of the Paillier encryption), which is used to extract a witness for a NIZK proof. Although the trapdoor enables to effectively complete the relevant security proof, a recent work by Bellare, Fuchsbauer, and Scafuro [5] revisited the security of NIZKs in the presence of a *subverted* CRS. The authors stressed that the existing techniques to obtain the trusted CRS (such as [9]) are not clear, and thus a CRS subversion *using the trapdoor* is a realistic attack as shown in the standardized random number generator embedding backdoors (disclosed by Snowden).

Until now, such a CRS-based technique has been widely used in many DGS constructions (e.g., [23, 25, 26]), based on the DGS model suggested by Bellare, Shi, and Zhang [8]. In the BSZ notion, it is assumed that there exists a trusted third party which generates an entire group public key gpk , containing all group managers' keys, public parameters, and (if necessary) a CRS (a more general case where these elements are generated independently is considered in [13]). This is the most convenient setting from the theoretical standpoint but its plausibility is questionable. In all cases, removing the trusted setup assumption is a worthwhile goal as it clearly facilitates practical deployments.

Recently, Fischlin et al. [30] introduced *sequential OR-proofs* that were used as a versatile tool to build a DGS scheme without a CRS and knowledge extractors. The resulting DGS scheme is provably secure in the *non-programmable* ROM, but in a weaker security model where the group managers (i.e., the issuer and the opener) are not separated. In addition, the size of a group public key and a group signature increases linearly in the number of group members. If we consider the real world use of group signatures (and variants) such as [38, 48], this means group signatures containing billions of elements.

To sum up, several works such as those mentioned above have attempted to overcome the drawbacks of knowledge extractors, but usually at the cost of efficiency or by assuming a trusted setup. A practical DGS scheme, without a CRS, proven without knowledge extractors is thus still an open problem.

1.2 Our Contributions

Our goal is to construct a practical DGS scheme that is proven to be secure in the ROM without a CRS and knowledge extractors. To achieve this, we focus on the following three steps:

Revisiting traceability proofs. As we explain above, there is often a small gap in the traceability proof of previous DGS schemes. The latter indeed try to rely on the unforgeability of the signature scheme Σ used by the issuer to sign usk but need at the same time to hide this value from the issuer to ensure non-frameability and also anonymity. This leads to enrolment protocols where the new member first sends to the issuer a commitment of usk (in practice some elements of a group \mathbb{G}) to get a signature on usk and thereby to essentially two proofs strategies.

If Σ signs elements of \mathbb{G} [2, 37], then the proof is rather simple as the reduction can simply forward the commitment to the signing oracle of Σ . The price to pay is the use of such specific signature schemes

(called structure-preserving) that are less efficient than their counterparts signing scalars in \mathbb{Z}_p (in particular because they yield signatures containing elements of both \mathbb{G}_1 and \mathbb{G}_2 , where $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ are so-called bilinear groups). Moreover, as said above, the use of such signatures will not solve our problems regarding non-frameability.

Else, Σ signs scalars and the reduction must then find a way to recover usk to query the corresponding signing oracle. This solution has the merit of being compatible with extremely efficient signature schemes [12, 20, 45] but requires knowledge extractors or one of the alternatives discussed in Section 1.1.

At first sight, it seems that we are stuck with these two alternatives. Fortunately, this is true only if one is willing to rely on the unforgeability of Σ , which is more than what is needed to prove traceability. Indeed, in the case where Σ signs groups elements, we do not need full-fledged structure-preserving signatures and their associated negative results [2] as a forgery (σ^*, M^*) is intuitively useful to an adversary against traceability only if it can prove some statement about M^* .

We therefore follow a different strategy and prove traceability directly under computational assumptions. As we want to leverage the efficiency of PS signatures [45], our computational assumptions will be related to the one (called PS assumption) underlying security of such signatures. More specifically, we define generalized variants of the PS assumption, which we call GPS_i ($i \in \{0, 1, 2\}$) and prove that traceability of our scheme essentially relies on GPS_2 . Clearly, introducing a new assumption to prove the security of our scheme is not ideal but we would like to emphasize two points.

Firstly, the most efficient groups signature schemes (e.g. [11, 45]) are built upon signature schemes that already rely on interactive ad-hoc assumptions. In this regard, it seems more natural to prove traceability directly under a tailored assumption: this is the same methodology but in our case it has the advantage of avoiding knowledge extractors. Put differently, as it is the case with the most efficient group signature schemes, our construction relies on a tailored assumption but our assumption at least allows us to get a tighter proof.

Secondly, our GPS_2 assumption is not a straightforward adaptation of the PS assumption where one could simply query the signing oracle on g^α instead of α . To provide confidence in our new computation problems, we indeed want to prove that they are as hard as the well-known symmetric discrete logarithm (SDL) problem in the algebraic group model (AGM) [31]. This leads us to define more elaborate inputs and outputs compared to original PS problem. In particular, the adversary now has to query its oracle with pairs (g^α, u^α) for different basis g and u which brings us back to something similar to the knowledge of exponents assumption (KEA) [7, 24].

Revisiting non-frameability proofs. Once we have dealt with the problem of knowledge extractors in the traceability proofs, we can focus on removing those from the non-frameability proofs. As we explain above, the need for knowledge extractors in the latter comes from the classical combination of proofs of knowledge (of usk) with the Fiat-Shamir heuristic. In the non-frameability proof, the challenger extracts usk from the NIZK proof contained in the forged signature and thereby solves some computational problem. In our construction, we depart from this standard strategy by relying instead on the lossy key technique from [1, 34]. Thanks to this technique, we only require simulation soundness (not extractability) from our NIZK proof, which can be proved without rewinding techniques in our case using the results from [27]. Concretely, in our non-frameability proof, we will gradually change the targeted user's public key and thus eventually force the adversary to produce a NIZK proof of a false statement, which contradicts simulation soundness. Of course, it remains to show how one can switch to lossy keys but this can be done quite easily in our concrete construction.

Revisiting anonymity proofs. Our last step in removing trusted setup and knowledge extractors is to revisit the anonymity proof. Indeed, to be as efficient as possible, we have chosen to use a rerandomizable signature

scheme (essentially the PS signature scheme [45]) to issue members' certificates. Thanks to such schemes, the groups members can include a rerandomized version of their certificate without encrypting it, which improves efficiency but also makes zero-knowledge proofs simpler (which further decreases complexity). This approach, originating (to our knowledge) from [11], has thus the merit of being more efficient than the standard one where the certificate is encrypted, but it makes the anonymity proof slightly harder. Indeed, contrarily to the standard approach, anonymity of the group signature cannot simply rely on the IND-CCA security of the encryption scheme. The usual strategy for such schemes (*e.g.* [11, 23, 45]) is then to replace, in the challenge signature, the certificate on the user's secret key usk_b^* by one generated on a random key usk , which removes any information on the signer from the group signature and thereby makes any adversary's advantage negligible. However, this strategy requires to issue a valid certificate on the random usk without interacting with the issuer. Existing solutions solve this problem by assuming a trusted setup [11, 23, 45], which allows the challenger to generate the issuer's secret key ik and so to produce new certificates. An alternative solution where the issuer's public key would contain a proof of knowledge of ik can be found in [37]. The challenger could then extract ik , which brings us back to our first case. Unfortunately, these solutions force us to either rely on a trusted setup or to use knowledge extractors, which we want to avoid. We therefore use a different proof strategy where we end up with a certificate (generated by the adversary) valid for both targeted users, making the adversary's advantage negligible.

In the end, we thus get a DGS construction that can be proved in a model stronger than the BSZ one [8] (that is, we do not require honest key generation for the issuer), without knowledge extractors. Contrarily to previous attempts, this is done with almost no impact on the efficiency, as shown in Section 6 where we compare our scheme with the most relevant ones from the state-of-the-art.

2 Preliminaries

2.1 Basic Notations

We use \mathbb{Z}_p to denote the set $\{0, \dots, p-1\}$, where p is a prime. Function $f : \mathcal{R} \rightarrow \mathcal{R}$ is called *negligible* if, for any $d > 0$, the inequality $|f(k)| < 1/k^d$ holds for sufficiently large k . We denote by $\mathbf{P}(\mathbf{A}(str_A) \leftrightarrow \mathbf{B}(str_B)) \rightarrow (out_A; out_B)$ the protocol \mathbf{P} containing the interactive algorithms \mathbf{A} and \mathbf{B} , where the algorithms respectively take str_A and str_B as input and then output out_A and out_B as results. $x \leftarrow_{\mathbb{S}} \mathbb{S}$ indicates the uniformly random sampling of the element x from the finite set \mathbb{S} . λ is used to denote a security parameter. All group operations are done modulo p unless otherwise stated throughout this paper.

Bilinear Maps. A bilinear group is a tuple $\mathcal{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, e)$, where $\mathbb{G}_1, \mathbb{G}_2$ and \mathbb{G}_T are groups of prime order p , $g \in \mathbb{G}_1, \hat{g} \in \mathbb{G}_2$ are generators of each group, and e is a bilinear map such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. As usual, a bilinear map (*i.e.*, pairing) e satisfies the following properties for all g and \hat{g} : bilinearity, non-degeneracy, and efficiency. We assume the existence of a (deterministic) group-generation algorithm $\text{GrGen}(1^\lambda) \rightarrow \mathcal{BG}$. In this paper, we only consider type-3 bilinear groups according to the classification in [32], that is, we assume that no efficiently computable homomorphism exists between \mathbb{G}_1 and \mathbb{G}_2 , in either direction.

Digital Signature. A digital signature scheme consists of three algorithms: (1) $\text{DSKg}(1^\lambda) \rightarrow (sk, vk)$: a key generation algorithm that outputs a signing key sk and a public key vk under λ ; (2) $\text{DSSig}(sk, m) \rightarrow \sigma_{DS}$: a signing algorithm that takes a signing key sk and a message m as input and then outputs a signature σ_{DS} ; and (3) $\text{DSVf}(vk, m', \sigma) \rightarrow \{0, 1\}$: a verifying algorithm that takes a verifying key vk , a message m' , and a signature σ_{DS} and then returns 1 if (m, σ) is valid under vk ; otherwise, 0. We consider that a digital signature scheme is secure if it achieves existential unforgeability under chosen message attacks (EUF-CMA) [35].

2.2 Algebraic Group Model

Following [31], we use an *algebraic* security game parametrized by a cyclic group $\mathcal{G} = (p, \mathbb{G})$ where a challenger \mathcal{C} interacts with an *algebraic* adversary \mathcal{A}_{alg} . Intuitively, an algebraic adversary can produce a new element in \mathbb{G} only through linear combinations of known elements in \mathbb{G} . The formal definition is as follows:

Definition 2.1. *An algorithm \mathcal{A}_{alg} executed in an algebraic security game $\mathbf{G}_{\mathcal{G}}$ is called algebraic if for all elements $h \in \mathbb{G}$ that \mathcal{A}_{alg} returns, it also outputs the representation of h regarding received elements in \mathbb{G} . More precisely, if $\mathcal{Q} = \{g_1, \dots, g_{|\mathcal{Q}|}\}$ is the list of group elements in \mathbb{G} that \mathcal{A}_{alg} has received so far, then \mathcal{A}_{alg} additionally returns a vector $\vec{h} = (h_1, \dots, h_{|\mathcal{Q}|}) \in \mathbb{Z}_p^{|\mathcal{Q}|}$, so that $h = \prod_{k=1}^{|\mathcal{Q}|} g_k^{h_k}$, letting $[h]_{\vec{h}}$ denote such an output.*

As with the standard security model, when $\mathbf{G}_{\mathcal{G}}$ provides a collection of oracles, \mathcal{C} deals with those oracles to answer queries from \mathcal{A}_{alg} . At the end of the game, $\mathbf{G}_{\mathcal{G}, \mathcal{A}_{alg}}^1$ outputs a bit $b \in \{0, 1\}$ as usual, and we identify that \mathcal{A}_{alg} wins when $b = 1$. Then, we define its advantage as $\mathbf{Adv}_{\mathcal{G}, \mathcal{A}_{alg}}^{\mathbf{G}} = \Pr[\mathbf{G}_{\mathcal{G}, \mathcal{A}_{alg}} = 1]$.

According to [4], the algebraic algorithm for a type-3 bilinear group \mathcal{BG} can be defined by setting $\mathbb{G} \in \{\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T\}$. An important fact is that elements in \mathbb{G}_2 and \mathbb{G}_T are not helpful to produce an element in \mathbb{G}_1 for this type of bilinear group as no efficiently computable isomorphism is known between these groups. We use this to prove new assumptions later in Section 3.2.

2.3 Dynamic Group Signature

We begin with the syntax of \mathcal{DGS} which is a slight modification of [13]. In the well-known models such as BSZ [8] and KY [39], a single trusted entity executes a group key generation to obtain a group public key and secret keys, and gives the relevant secret keys to each of the issuer and the opener. Such a key generation in the BSZ model must be assumed to be honest, and (if necessary) a CRS can also be included in the group public key. Conversely, Bootle et al. [13] separated the group key generation into three ones; a setup algorithm for generating public parameters, a key generation algorithm for the issuer and a key generation algorithm for the opener. Moreover, in terms of security, [13] allows an adversarial issuer or opener to generate its key maliciously and thus encompasses the weaker security model of [8, 39].

In the following description, there exists slight difference between [13] and our syntax; a user is associated with an identity i by using a PKI (such as [8]), and the issuer/opener key generation algorithm is executed individually without interaction.

$\text{Setup}(1^\lambda) \rightarrow pp$. Under a security parameter λ , the setup algorithm outputs the public parameter pp .

$\text{IKg}(pp) \rightarrow (ik, ipk)$. Given a public parameter pp , the issuer key generation algorithm outputs the issuer's public/secret key pair (ipk, ik) .

$\text{OKg}(pp) \rightarrow (ok, opk)$. Given a public parameter pp , the opener key generation algorithm outputs the opener's public/secret key pair (opk, ok) .

$\text{UKg}(1^\lambda, i) \rightarrow (upk_i, usk_i)$. User i invokes the user key generation algorithm to produce its public key upk_i and secret key usk_i . We assume that upk_i is authenticated by a certification authority (CA).

¹For simplicity, we omit λ in related notation throughout this paper.

$\mathbf{GJoin}(\mathbf{Join}(i, usk_i, gpk) \leftrightarrow \mathbf{Iss}(i, upk_i, ik)) \rightarrow (gsk_i; reg_i)$. User i executes the \mathbf{GJoin} protocol with the issuer. The algorithm \mathbf{Join} , run by the user i , takes the user secret key usk_i and the group manager's public key $gpk := (pp, ipk, opk)$. The algorithm \mathbf{Iss} , run by the issuer, takes the user public key upk_i and its secret key ik . At the end of the protocol, the user obtains a group signing key gsk_i , and the issuer adds registration information reg_i for the user i onto registration list \mathbf{reg} . This list is shared with the opener, who then has access to read \mathbf{reg} .

$\mathbf{GSig}(gsk_i, m) \rightarrow \sigma$. User i , with group signing key gsk_i , invokes the signing algorithm to produce a signature σ on a message $m \in \{0, 1\}^*$.

$\mathbf{GVf}(gpk, m, \sigma) \rightarrow \{0, 1\}$. Anyone can invoke the verification algorithm to verify a group signature. The algorithm takes the group manager's public key $gpk = (pp, ipk, opk)$, a message m , and a signature σ , and outputs 1 if σ is valid on m ; otherwise, it outputs 0.

$\mathbf{GOpen}(ok, m, \sigma, \mathbf{reg}) \rightarrow (i, \Pi)$ or \perp . The opener invokes the opening algorithm to recover the identity i of a user who has produced a given signature. The algorithm takes the opening key ok , a message m , a valid signature σ and the registration list \mathbf{reg} . It outputs an identity i and a proof Π claiming that the user i produced σ on m or it outputs \perp .

$\mathbf{GJudge}(m, \sigma, gpk, i, upk_i, \Pi) \rightarrow \{0, 1\}$. Anyone can invoke the judging algorithm to verify the opener's output. The algorithm takes a message m , a valid signature σ on m , the group public key $gpk = (pp, ipk, opk)$, an identity i along with its public key upk_i , and a proof Π that was generated while opening. It outputs 1 if Π is valid; otherwise, it outputs 0.

Definition 2.2. A DGS scheme is correct if the following conditions hold for an honest user i and any message m :

$$\Pr \left[\begin{array}{l} //gpk := (pp, ipk, opk) \\ \mathbf{GVf}(gpk, m, \sigma) = 1 \wedge \\ \mathbf{GJudge}(m, \sigma, gpk, i, upk_i, \Pi) = 1 \\ \wedge i = j \end{array} : \begin{array}{l} pp \leftarrow \mathbf{Setup}(1^\lambda), \\ (ik, ipk) \leftarrow \mathbf{IKg}(pp), (ok, opk) \leftarrow \mathbf{OKg}(pp), \\ (upk_i, usk_i) \leftarrow \mathbf{UKg}(1^\lambda, i), \\ (gsk_i, reg_i) \leftarrow \mathbf{GJoin}(\mathbf{Join}(i, usk_i, gpk) \\ \quad \leftrightarrow \mathbf{Iss}(i, upk_i, ik)), \\ \sigma \leftarrow \mathbf{GSig}(gsk_i, m), \\ (j, \Pi) \leftarrow \mathbf{GOpen}(ok, m, \sigma, \mathbf{reg}) \end{array} \right] = 1.$$

2.3.1 Oracles Description.

To define security models where the challenger \mathcal{C} interacts with an adversary \mathcal{A} , we give the description of oracles used by \mathcal{A} in Figure 1. \mathcal{C} therein maintains the following lists (\mathcal{L}_h , \mathcal{L}_c , \mathcal{L}_{sk} , \mathcal{L}_σ , and \mathcal{L}_{ch}) to control those oracles. Here, the state \mathbf{cont} indicates that the user is corrupted but has not yet joined, while the state \mathbf{accept} indicates that the user is corrupted and has already been accepted to join the system.

2.3.2 Security Models.

Following [13] with the minor syntactic changes, our security definitions of anonymity and non-frameability capture strong notions by allowing maliciously generated keys of issuer and/or opener.

Anonymity. We here consider the *selfless anonymity* notion [19] which states that a signature generated using a key gsk can be opened only by the opener and the owner of gsk (that is, a group member is always

<p><u>AddU</u>(i):</p> <ul style="list-style-type: none"> - $(upk_i, usk_i) \leftarrow \text{UKg}(1^\lambda, i)$. - $(gsk_i; reg_i) \leftarrow \text{GJoin}(\text{Join}(i, usk_i, gpk) \leftrightarrow \text{lss}(i, upk_i, ik))$. - $\mathbf{reg} \leftarrow \mathbf{reg} \cup \{reg_i\}$. - $\mathcal{L}_h \leftarrow \mathcal{L}_h \cup \{i\}$. - Return upk_i. <p><u>CrptU</u>(i, upk):</p> <ul style="list-style-type: none"> - $upk_i \leftarrow upk$. - $\mathcal{L}_c \leftarrow \mathcal{L}_c \cup \{(i, \text{cont})\}$. <p><u>SndToI</u>(i):</p> <ul style="list-style-type: none"> - If $(i, \text{cont}) \notin \mathcal{L}_c$ then return \perp. - $reg_i \leftarrow \text{GJoin}(\mathcal{A} \leftrightarrow \text{lss}(i, upk_i, ik))$. - $\mathbf{reg} \leftarrow \mathbf{reg} \cup \{reg_i\}$. - $\mathcal{L}_c \leftarrow \mathcal{L}_c \cup \{(i, \text{accept})\}$. <p><u>SndToU</u>(i):</p> <ul style="list-style-type: none"> - $(upk_i, usk_i) \leftarrow \text{UKg}(1^\lambda, i)$. - $gsk_i \leftarrow \text{GJoin}(\text{Join}(i, usk_i, gpk) \leftrightarrow \mathcal{A})$. - $\mathcal{L}_h \leftarrow \mathcal{L}_h \cup \{i\}$. <p><u>USK</u>(i):</p> <ul style="list-style-type: none"> - If $(i, *, *) \in \mathcal{L}_{ch}$ then return \perp. - $\mathcal{L}_{sk} \leftarrow \mathcal{L}_{sk} \cup \{i\}$. - Return (gsk_i, usk_i). 	<p><u>RReg</u>(i):</p> <ul style="list-style-type: none"> - Return reg_i. <p><u>WReg</u>(i, ρ):</p> <ul style="list-style-type: none"> - $reg_i \leftarrow \rho$. <p><u>Sig</u>(i, m):</p> <ul style="list-style-type: none"> - If $i \notin \mathcal{L}_h$, then return \perp. - $\sigma \leftarrow \text{GSig}(gsk_i, m)$. - $\mathcal{L}_\sigma \leftarrow \mathcal{L}_\sigma \cup \{(i, m, \sigma)\}$ - Return σ. <p><u>Open</u>(m, σ):</p> <ul style="list-style-type: none"> - If $(*, m^*, \sigma^*) \in \mathcal{L}_{ch}$, then return \perp. - Return $\text{GOpen}(ok, m, \sigma, \mathbf{reg})$. <p><u>CH_b</u>(i_0^*, i_1^*, m^*):</p> <ul style="list-style-type: none"> - If $i_0^* \notin \mathcal{L}_h$ or $i_1^* \notin \mathcal{L}_h$ then return \perp. - If $i_0^* \in \mathcal{L}_{sk}$ or $i_1^* \in \mathcal{L}_{sk}$ then return \perp. - $\sigma^* \leftarrow \text{GSig}(gsk_{i_b^*}, m^*)$. - $\mathcal{L}_{ch} \leftarrow \{(i_0^*, m^*, \sigma^*), (i_1^*, m^*, \sigma^*)\}$. - Return σ^*.
---	---

Figure 1: Oracles used by an adversary

able to recognize his signatures). It is formally defined in Figure 2. The goal of \mathcal{A} , which may control the issuer, is to successfully guess if either i_0^* or i_1^* is the signer's identity for the challenge signature σ^* , provided that $gsk_{i_0^*}$ and $gsk_{i_1^*}$ did not leak. The advantage of \mathcal{A} in the anonymity game in Figure 2 is therefore defined as

$$\mathbf{Adv}_{\mathcal{DGS}, \mathcal{A}}^{\text{Anon}} = \left| \Pr[\mathbf{G}_{\mathcal{DGS}, \mathcal{A}}^{\text{Anon}-0} = 1] - \Pr[\mathbf{G}_{\mathcal{DGS}, \mathcal{A}}^{\text{Anon}-1} = 1] \right|.$$

Definition 2.3. A group signature scheme \mathcal{DGS} is anonymous if $\mathbf{Adv}_{\mathcal{DGS}, \mathcal{A}}^{\text{Anon}}$ is negligible for any λ and any polynomial-time adversary \mathcal{A} .

Non-frameability. This notion ensures that an adversary cannot generate a valid signature that opens to a certain honest user who has never actually signed the message of the signature. Non-frameability is required to hold even when the issuer and the opener are both compromised. The non-frameability game is defined in

$\mathbf{G}_{DGS, \mathcal{A}}^{Anon-b}$:

- $pp \leftarrow \text{Setup}(1^\lambda)$.
- $(ipk, st_{\mathcal{A}}) \leftarrow \mathcal{A}(pp)$.
- $(ok, opk) \leftarrow \text{OKg}(pp)$.
- $b' \leftarrow \mathcal{A}^{\text{SndToU, USK, WReg, Sig, Open, CH}_b}(opk; st_{\mathcal{A}})$.
- Return b' .

Figure 2: Anonymity game for DGS

Figure 3. Following [13], we assume without loss of generality that a single user i^* is honest. The goal of \mathcal{A} is then to create a valid forgery σ^* on a message m^* that can be opened to i^* , along with a judge-accepted proof Π^* . The advantage of \mathcal{A} in the non-frameability game defined in Figure 3 is $\text{Adv}_{DGS, \mathcal{A}}^{Nf} = \Pr[\mathbf{G}_{DGS, \mathcal{A}}^{Nf} = 1]$.

Definition 2.4. A group signature scheme DGS is non-frameable if $\text{Adv}_{DGS, \mathcal{A}}^{Nf}$ is negligible for any λ and any polynomial-time adversary \mathcal{A} .

$\mathbf{G}_{DGS, \mathcal{A}}^{Nf}$:

- $pp \leftarrow \text{Setup}(1^\lambda)$.
- $(ipk, opk, st_{\mathcal{A}}) \leftarrow \mathcal{A}(pp)$.
- $gpk := (pp, ipk, opk)$.
- $(m^*, \sigma^*, i^*, \Pi^*) \leftarrow \mathcal{A}^{\text{SndToU, USK, Sig}}(st_{\mathcal{A}})$.
- If the following conditions hold, then return 1.
 - (1) $\text{GVf}(gpk, m^*, \sigma^*) = 1$.
 - (2) $\{i^*\} = \mathcal{L}_h \setminus \mathcal{L}_{sk}$ and $i^* \notin \mathcal{L}_{sk}$ and $(i^*, m^*, *) \notin \mathcal{L}_\sigma$.
 - (3) $\text{GJudge}(m^*, \sigma^*, gpk, i^*, upk^*, \Pi^*) = 1$.
- Return 0.

Figure 3: Non-frameability game for DGS

Traceability. This notion implies that an adversary can only produce traceable signatures when the issuer is honest and the opener is partially corrupt. The meaning of being partially corrupt is introduced in [8], where the opener is controlled by \mathcal{A} but invokes the GOpen algorithm honestly. This is a necessary restriction as one cannot retain any meaningful notion otherwise (a fully malicious opener could return \perp as the result of each query). Concretely, this means that the knowledge of the opener secret key ok is of no help to escape identification. We note that the notion of the partial corruption is almost the same as in [13]. The traceability game is defined in Figure 4. \mathcal{A} can succeed in either of two ways: (a) by generating a valid signature σ^* that cannot be opened (i.e., GOpen outputs \perp on σ^*) (b) by preventing the challenger from outputting a valid proof Π^* , i.e. one accepted by GJudge . The advantage of \mathcal{A} in the traceability game shown in Figure 4 is $\text{Adv}_{DGS, \mathcal{A}}^{\text{Trace}} = \Pr[\mathbf{G}_{DGS, \mathcal{A}}^{\text{Trace}} = 1]$.

Definition 2.5. A group signature scheme \mathcal{DGS} is traceable if $\mathbf{Adv}_{\mathcal{DGS}, \mathcal{A}}^{\text{Trace}}$ is negligible for any λ and any polynomial-time adversary \mathcal{A} .

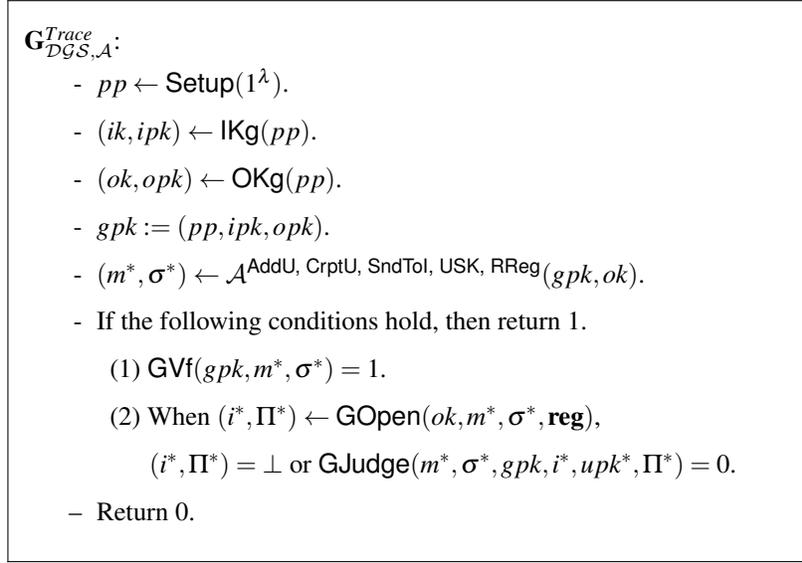


Figure 4: Traceability game for \mathcal{DGS}

2.4 Complexity Assumptions

Let $\mathcal{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, e)$ be a type-3 pairing group.

Symmetric Discrete Logarithm (SDL) Assumption [11]. Given a tuple (g, g^z) , where $z \leftarrow_{\$} \mathbb{Z}_p$, the DL problem is to find z . The symmetric version (the so-called SDL) is defined when a pair (\hat{g}, \hat{g}^z) is additionally given, where $\hat{g} \in \mathbb{G}_2$ is a generator.

External Diffie–Hellman (XDH) Assumption [12]. Given a tuple (g, g^a, g^b, T) , where $a, b \leftarrow_{\$} \mathbb{Z}_p$, the XDH problem in \mathbb{G}_1 (denoted by $\text{XDH}_{\mathbb{G}_1}$) is to decide whether $T = g^{ab}$ or $R \leftarrow_{\$} \mathbb{G}_1$.

3 GPS Assumptions and Their Relations

3.1 GPS Assumptions

Pointcheval and Sanders [45] introduced a new interactive assumption called the PS assumption, thereby realizing a very efficient randomizable signature scheme. Recently, the generalized version of PS (denoted as GPS) assumption was introduced in [40] by splitting the equipped oracle into two ones, analogously to the generalized LRSW assumption [17]. On top of those results, we furthermore expand the GPS assumption in the direction of replacing all exponential values with group elements, which can be viewed as a first step to *construct a structure-preserving signature from the PS assumption*². Those assumptions are defined in a type-3 pairing group $\mathcal{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, e)$ as follows:

²As explained in the Section 1.2, we do not want a full-fledged structure-preserving signature as it would expose us to the associated negative results [2] such as the impossibility of unilateral signatures (that is, signatures only containing elements of \mathbb{G}_1).

PS Assumption [45]. Given a tuple (\hat{g}^x, \hat{g}^y) and an oracle $\mathcal{O}^{PS}(m) \rightarrow (u, u^{x+ym}) \in \mathbb{G}_1^2$, where $x, y \leftarrow_{\$} \mathbb{Z}_p$, $m \in \mathbb{Z}_p$ and $u \leftarrow_{\$} \mathbb{G}_1$, the PS problem is to find a tuple $(u^*, v^*, m^*) \in \mathbb{G}_1^2 \times \mathbb{Z}_p$ such that (1) $u^* \neq 1$ and $m^* \neq 0$, (2) $v^* = u^{*x+ym^*}$, and (3) for all m queried to \mathcal{O}^{PS} , $m^* \neq m$. The PS assumption holds if the advantage of solving the problem is negligible in the security parameter λ .

GPS_i Assumption. For $i \in \{0, 1, 2\}$, given a tuple (\hat{g}^x, \hat{g}^y) , where $x, y \leftarrow_{\$} \mathbb{Z}_p$ and two oracles \mathcal{O}_1 and \mathcal{O}_2 as shown in Figure 6, the GPS_i problem is to find a tuple (u^*, v^*, m^*) in the GPS₀ and GPS₁ along with a tuple (f^*, u^*, v^*, w^*) in the GPS₂. The GPS_i assumption holds if the advantage of solving the problem is negligible in λ .

We note that the definition of the GPS₂ problem does not allow to verify the relation $\log_g f^* = \log_{u^*} w^*$ given only the group elements (f^*, u^*, v^*, w^*) in \mathbb{G}_1 . However, this will not be a problem for our *DGS* construction because it uses an additionally NIZK proof to prove the equality of these discrete logarithms. The Figure 5 shows the relations between these new assumptions and the SDL one.

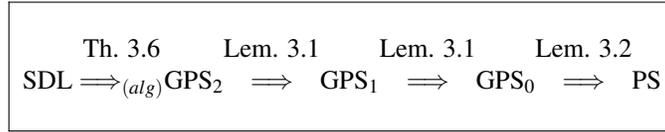


Figure 5: Relations between GPS assumptions

Lemma 3.1. *The GPS₁ assumption holds under the GPS₂ assumption. Moreover, the GPS₀ assumption holds under the GPS₁ assumption.*

Proof. Consider two security reductions: one from GPS₂ to GPS₁ where an adversary \mathcal{A}_1 against GPS₁ and its challenger \mathcal{C}_1 (i.e., an adversary \mathcal{A}_2 against GPS₂ together) exist, and one from GPS₁ to GPS₀, where an adversary \mathcal{A}_0 against GPS₀ and its challenger \mathcal{C}_0 exist. Therein, \mathcal{C}_0 bridges these reductions by playing a role in \mathcal{A}_1 together.

Now, it is desirable to show that while \mathcal{C}_1 , given an GPS₂ instance, simulates GPS₁ for $\mathcal{A}_1 (= \mathcal{C}_0)$, \mathcal{C}_0 simulates GPS₀ for \mathcal{A}_0 . Since the problem instances are the same as (\hat{g}^x, \hat{g}^y) for some hidden $x, y \in \mathbb{Z}_p$, \mathcal{C}_1 relays the instance to \mathcal{C}_0 , and sequentially, \mathcal{C}_0 relays it to \mathcal{A}_0 . Whenever \mathcal{C}_0 is asked to invoke $\mathcal{O}_1^{GPS_0}$ from \mathcal{A}_0 , it requests $\mathcal{O}_1^{GPS_1}$, which is managed by \mathcal{C}_1 . Then, \mathcal{C}_1 calls $\mathcal{O}_1^{GPS_2}$ to obtain u as output. This value u is forwarded to \mathcal{C}_0 and \mathcal{A}_0 in sequence. Whenever \mathcal{A}_0 sends a query $(u, m) \in \mathbb{G}_1 \times \mathbb{Z}_p$ to $\mathcal{O}_2^{GPS_0}$, \mathcal{C}_0 computes $f = g^m$ and $w = u^m$. Using these, \mathcal{C}_0 calls $\mathcal{O}_2^{GPS_1}$ with (f, u, w) , and then \mathcal{C}_1 relays the tuple to $\mathcal{O}_2^{GPS_2}$. Clearly, such a (f, u, w) should be valid for both $\mathcal{O}_2^{GPS_1}$ and $\mathcal{O}_2^{GPS_2}$ as long as the query from \mathcal{A}_0 valid. If $\mathcal{O}_2^{GPS_2}$ returns v , then \mathcal{A}_0 and \mathcal{C}_0 can receive v from their respective challengers.

In the end, \mathcal{A}_0 outputs $(u^*, v^*, m^*) \in \mathbb{G}_1^2 \times \mathbb{Z}_p$. If \mathcal{A}_0 wins, then \mathcal{C}_0 can send a valid output to \mathcal{C}_1 by forwarding the tuple. From this, \mathcal{C}_1 obtains $f^* = g^{m^*}$ and $w^* = u^{*m^*}$. It is obvious that \mathcal{C}_1 can win with (f^*, u^*, v^*, w^*) . \square

Finally, it was shown in [40] that the adversary against the GPS₀ problem can be changed into one against the original PS problem, leading to the following lemma.

Lemma 3.2. *The original PS assumption holds under the GPS₀ assumption.*

	Oracles	Output
GPS ₀ [40]	$\mathcal{O}_1(\cdot) \rightarrow u$: - $u \leftarrow_{\mathcal{S}} \mathbb{G}_1$. - $\mathcal{Q}_1 = \mathcal{Q}_1 \cup \{u\}$. $\mathcal{O}_2(u, m) \rightarrow v$: // $m \in \mathbb{Z}_p$. - If $u \notin \mathcal{Q}_1$ or $(u, *) \in \mathcal{Q}_2$ then return \perp . - $v = u^{x+my}$. - $\mathcal{Q}_2 = \mathcal{Q}_2 \cup \{(u, m)\}$.	$\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2}(\hat{g}^x, \hat{g}^y) \rightarrow (u^*, v^*, m^*)$. // $u, v \in \mathbb{G}_1$. // $m \in \mathbb{Z}_p$. Output is valid if (1) $u^* \neq 1$ and $m^* \neq 0$. (2) $v^* = u^{*x+m^*y}$. (3) $\forall (*, m) \in \mathcal{Q}_2, m^* \neq m$.
GPS ₁ [40]	$\mathcal{O}_1(\cdot) \rightarrow u$: - $r \leftarrow_{\mathcal{S}} \mathbb{Z}_p$. - $u = g^r$. - $\mathcal{Q}_1 = \mathcal{Q}_1 \cup \{u\}$. $\mathcal{O}_2(f, u, w) \rightarrow v$: // $f, w \in \mathbb{G}_1$. - If $u \notin \mathcal{Q}_1$ or $(u, *) \in \mathcal{Q}_2$ then return \perp . - If $w \neq f^r$ s.t. $r = \log_g u$, then return \perp . - $v = u^x w^y$. - $\mathcal{Q}_2 = \mathcal{Q}_2 \cup \{(u, f)\}$.	$\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2}(\hat{g}^x, \hat{g}^y) \rightarrow (u^*, v^*, m^*)$. // $u, v \in \mathbb{G}_1$. // $m \in \mathbb{Z}_p$. Output is valid if (1) $u^* \neq 1$ and $m^* \neq 0$. (2) $v^* = u^{*x+m^*y}$. (3) $\forall (*, f) \in \mathcal{Q}_2, g^{m^*} \neq f$.
GPS ₂	$\mathcal{O}_1(\cdot) \rightarrow u$: - $r \leftarrow_{\mathcal{S}} \mathbb{Z}_p$. - $u = g^r$. - $\mathcal{Q}_1 = \mathcal{Q}_1 \cup \{u\}$. $\mathcal{O}_2(f, u, w) \rightarrow v$: // $f, w \in \mathbb{G}_1$. - If $u \notin \mathcal{Q}_1$ or $(u, *) \in \mathcal{Q}_2$ then return \perp . - If $w \neq f^r$ s.t. $r = \log_g u$, then return \perp . - $v = u^x w^y$. - $\mathcal{Q}_2 = \mathcal{Q}_2 \cup \{(u, f)\}$.	$\mathcal{A}^{\mathcal{O}_1, \mathcal{O}_2}(\hat{g}^x, \hat{g}^y) \rightarrow (f^*, u^*, v^*, w^*)$. // $f^*, u^*, v^*, w^* \in \mathbb{G}_1$. Output is valid if (1) $u^* \neq 1$ and $f^* \neq 1$. (2) $v^* = u^{*x} w^{*y}$. (3) $\log_g f^* = \log_{u^*} w^*$. // non-falsifiable (4) $\forall (*, f) \in \mathcal{Q}_2, f^* \neq f$.

Figure 6: Description of GPS_i problems

3.2 Reduction from SDL to GPS₂ in the AGM

Our reduction from SDL to GPS₂ in the AGM shares many similarities with the one from DL to Knowledge of Exponent Assumption (KEA) [7, 24]. We therefore start by describing the latter³, as it may help to understand the strategy behind the former.

KEA in the AGM. The KEA states that, for any adversary \mathcal{A} that has a tuple $(g, u = g^r) \in \mathbb{G}^2$ for input will output $(f, w = f^r) \in \mathbb{G}^2$, there exists an extractor $\mathcal{E}_{\mathcal{A}}^{KEA}$ that, with the same input, returns m such that $f = g^m$ and $w = u^m$. In the AGM, when an algebraic adversary \mathcal{A}_{alg} outputs (f, w) along with their representations (described below), there exists an extractor $\mathcal{E}_{\mathcal{A}_{alg}}^{KEA}$ which outputs the discrete logarithm $m = \log_g f = \log_u w$ that \mathcal{A}_{alg} knows if the DL assumption holds in the group \mathbb{G} . In terms of proving the equality of discrete logarithms (EDL) especially using fixed bases given to \mathcal{A}_{alg} , we now have an efficient knowledge extractor in the AGM even without any extraction techniques, such as (rewinding) proofs of the EDL or extractable commitments. More formally, we prove that there is an extractor for KEA in the AGM under the DL assumption.

Theorem 3.3. *Under the DL assumption, the KEA holds in the AGM.*

Proof. Borrowing the proof strategy in [4], we show that a non-zero polynomial yields the DL solution as one of its roots, given the adversary's representations. Let (g, g^z) be a DL instance. If an algebraic adversary \mathcal{A}_{alg} with input $(g, u = g^r)$ outputs $(f = g^m, w = u^m)$ along with their representations $\vec{f} = (f_1, f_2) \in \mathbb{Z}_p^2$ and $\vec{w} = (w_1, w_2) \in \mathbb{Z}_p^2$, this means that $f = g^{f_1} u^{f_2}$ and $w = g^{w_1} u^{w_2}$. The equation $\log_g f = \log_u w$ then gives the following equation:

$$w_1 + (w_2 - f_1)r - f_2r^2 = 0. \quad (1)$$

There are two possible cases. If the polynomial $w_1 + (w_2 - f_1)r - f_2r^2$ is the zero polynomial (i.e., all coefficients are zero with respect to the variable r), we have the relations $w_1 = 0$, $w_2 = f_1$, and $f_2 = 0$, which give $f = g^{f_1}$ and $w = u^{w_2}$. Therefore, we can build an extractor $\mathcal{E}_{\mathcal{A}_{alg}}^{KEA}$ that outputs $m := f_1$ as the discrete logarithm. Otherwise, if the polynomial $w_1 + (w_2 - f_1)r - f_2r^2$ is a non-zero polynomial, (i.e., some coefficients are not zero with respect to the variable r), we can solve the DL problem as follows: by setting $u = (g^z)^s g^t$ for randomly chosen s, t in \mathbb{Z}_p , r can be implicitly set as $r = sz + t$. In this case, the equation (1) can be rewritten to $a_0 + a_1z + a_2z^2 = 0$, where $a_0 = w_1 + w_2t - f_1t - f_2t^2$, $a_1 = w_2s - f_1s - 2f_2st$, and $a_2 = -f_2s^2$. As $w_1 + (w_2 - f_1)r - f_2r^2$ is a non-zero polynomial with one root, we know that we cannot have $w_2 - f_1 = f_2 = 0$. Therefore, since s is randomly sampled and information-theoretically hidden from the view of \mathcal{A}_{alg} , the probability that $a_0 = 0 \wedge a_1 = 0 \wedge a_2 = 0$ is negligible. Consequently, we can solve the DL problem by finding roots of the non-zero polynomial $a_0 + a_1Z + a_2Z^2 \in \mathbb{Z}_p[Z]$ of a degree that is at least 1. \square

With this observation, we now prove that the GPS₂ problem in the AGM is at least as hard as the SDL problem. We recall the Schwarz-Zippel lemma:

Lemma 3.4 (Schwarz-Zippel Lemma). *Let $P \in \mathbb{F}[X_1, \dots, X_\ell]$ be a non-zero polynomial of total degree $d \geq 0$ over a field \mathbb{F} . If the values r_1, \dots, r_ℓ are independently chosen at random from a finite subset $S \subset \mathbb{F}$, then $\Pr[P(r_1, \dots, r_\ell) = 0] \leq \frac{d}{|S|}$.*

³In [31], the authors mentioned that the KEA holds in the AGM, but a concrete reduction was not provided.

We now provide additional notations relative to representations that an algebraic adversary \mathcal{A}_{alg} outputs in the AGM. Let $\mathcal{Q} = \{g_1, \dots, g_{|\mathcal{Q}|}\}$ be the set of group elements known to \mathcal{A}_{alg} . By definition 2.1, \mathcal{A}_{alg} should outputs $[h]_{\vec{h}}$ which contains $h \in \mathbb{G}$ and $\vec{h} = (h_1, \dots, h_{|\mathcal{Q}|}) \in \mathbb{Z}_p^{|\mathcal{Q}|}$ such that $h = \prod_{k=1}^{|\mathcal{Q}|} g_k^{h_k}$. We denote such an element h_k regarding $g_k \in \mathcal{Q}$ by $[h|g_k]$. If each of $\{\log_g g_k\}$ is represented by an ℓ -variate polynomial in $\mathbb{Z}_p[\mathbf{X}_1, \dots, \mathbf{X}_\ell]$ for some $\ell \in \mathbb{N}$, we can define a polynomial $P_h[\vec{\mathbf{X}}]$, where $\vec{\mathbf{X}} = (\mathbf{X}_1, \dots, \mathbf{X}_\ell)$, to denote $\log_g h = \sum_{k=1}^{|\mathcal{Q}|} [h|g_k] \cdot \log_g g_k$. It is easy to see that the coefficients of $P_h[\vec{\mathbf{X}}]$ are composed of linear combinations of elements in \vec{h} . Moreover, an evaluation of the polynomial $P_h[\vec{\mathbf{X}}]$ at $\mathbf{x} = (x_1, \dots, x_\ell)$, where $x_j \in \mathbf{X}_j$, specifies $\log_g h$ to some value $z \in \mathbb{Z}_p$ (i.e., $P_h(\mathbf{x}) = z$). Regarding any polynomial P , we hereafter use $P(\mathbf{x}) = 0$ to mean that the polynomial evaluates zero at the point \mathbf{x} , and use $P[\vec{\mathbf{X}}] = 0$ to mean that the polynomial in variables $\vec{\mathbf{X}}$ is the zero-polynomial (of which all coefficients are zero). Based on these notations, we refer to the following lemma from [4].

Lemma 3.5 (Lemma 2.1 in [4]). *Let $P[\vec{\mathbf{X}}] \in \mathbb{Z}_p[\mathbf{X}_1, \dots, \mathbf{X}_\ell]$ be a non-zero multivariate polynomial of the total degree $d \in \mathbb{N}$, and let $Q[Z] \in \mathbb{Z}_p[Z]$ be a univariate polynomial generated as $P[S_1 Z + T_1, \dots, S_\ell Z + T_\ell]$. Then, the coefficient of the maximal degree of $Q[Z]$ is a polynomial in $\mathbb{Z}_p[S_1, \dots, S_\ell]$ of degree d .*

The probability that $Q[Z]$ is the zero-polynomial is less than the probability that the leading coefficient a_d of $Q[Z]$ is zero (i.e., $\Pr[Q[Z] = 0] \leq \Pr[a_d = 0]$). Putting lemmas 3.4 and 3.5 together, we can obtain the upper-bounded probability that $Q[Z] = 0$ in security reductions.

GPS₂ problem in the AGM. Figure 7 presents the GPS₂ problem in the AGM. Compared to the previous GPS₂ problem in Figure 6, there are two primary differences, besides the representations about the group elements that an adversary (denoted as \mathcal{A}_{alg}) outputs. The first one is that the \mathcal{O}_2 oracle contains the extractor $\tilde{\mathcal{E}}$ (marked in the red dotted box) which is a (deterministic) polynomial-time algorithm; briefly speaking, $\tilde{\mathcal{E}}$ takes two group elements $f, w \in \mathbb{G}_1$ along with their representations, and outputs an exponent $m \in \mathbb{Z}_p$ such that $f = g^m$ and $w = u^m$ or \perp . Whether $\tilde{\mathcal{E}}$ outputs m or \perp depends on whether or not the new polynomial $R_k P_{f_k}[\vec{\mathbf{X}}] - P_{w_k}[\vec{\mathbf{X}}]$, determined by the input of \mathcal{A}_{alg} , is the zero-polynomial. If the polynomial is not the zero-polynomial, $\tilde{\mathcal{E}}$ outputs \perp , in which case, fortunately, the SDL problem is solved. That is, assuming that the SDL problem is hard, $\tilde{\mathcal{E}}$ succeeds in extracting the discrete logarithm m and thus \mathcal{O}_2 is able to respond to the \mathcal{A}_{alg} query. The other difference is that the equality of $\log_g f^* = \log_{u^*} w^*$ in (w3) can be efficiently checked by a polynomial evaluation, which was not possible in the GPS₂ problem in Figure 6; indeed, the equality $\log_g f^* = \log_{u^*} w^*$ can be written by evaluating polynomials as follows:

$$P_f^*(\mathbf{x}) = \log_g f^* = \frac{\log_g w^*}{\log_g u^*} = \frac{P_w^*(\mathbf{x})}{P_u^*(\mathbf{x})},$$

where \mathbf{x} is a vector of points chosen by the challenger to \mathcal{A}_{alg} . The GPS₂ assumption is thus falsifiable in the AGM. We can now formally state our result regarding the reduction from SDL to GPS₂.

Theorem 3.6. *Under the SDL assumption, the GPS₂ assumption in the AGM holds.*

Proof. Let \mathcal{A}_{alg} be an algebraic adversary and \mathcal{C} be a challenger. As an adversary of the SDL problem, \mathcal{C} is given an SDL instance $(g, \hat{g}, Z = g^z, \hat{Z} = \hat{g}^z)$. \mathcal{C} begins a simulation by sampling $\{s_0, t_0, s'_0, t'_0\} \leftarrow_{\$} \mathbb{Z}_p$, and implicitly sets $x = s_0 z + t_0$ and $y = s'_0 z + t'_0$. \mathcal{A}_{alg} is initially given $\hat{X} = \hat{g}^x = \hat{Z}^{s_0} \hat{g}^{t_0}$ and $\hat{Y} = \hat{g}^y = \hat{Z}^{s'_0} \hat{g}^{t'_0}$.

Let q_1 be the maximum number of oracle queries to \mathcal{O}_1 . For the i -th \mathcal{O}_1 query, \mathcal{C}_2 returns $u_i = Z^{s_i} g^{t_i}$ where $\{s_i, t_i\} \leftarrow_{\$} \mathbb{Z}_p^*$, implicitly setting $r_i = s_i z + t_i$. For the k -th \mathcal{O}_2 query, we assume that \mathcal{C} successfully responds to the previous $(k-1)$ -th \mathcal{O}_2 queries. In that case, $\{v_\ell = g^{r_\ell(x+m_\ell y)}\}_{\ell=1}^{k-1}$ are given to \mathcal{A}_{alg} ,

$\mathbf{G}_{\mathcal{BG}, \mathcal{A}_{alg}}^{GPS_2}$: <ul style="list-style-type: none"> - $\mathcal{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, e) \leftarrow \text{GrGen}(1^\lambda)$. - $x \leftarrow_{\\$} \mathbb{Z}_p := \mathbf{X}, y \leftarrow_{\\$} \mathbb{Z}_p := \mathbf{Y}$. - $[f^*]_{\mathbb{F}^*}, [u^*]_{\mathbb{U}^*}, [v^*]_{\mathbb{V}^*}, [w^*]_{\mathbb{W}^*} \leftarrow \mathcal{A}_{alg}^{\mathcal{O}_1, \mathcal{O}_2}(\hat{g}^x, \hat{g}^y)$. - If the following conditions hold, then return 1. <ul style="list-style-type: none"> (w1) $f^* \neq 1$ and $u^* \neq 1$. (w2) $v^* = u^{*x} w^{*y}$. (w3) $\log_g f^* = \log_{u^*} w^*$ <div style="border: 1px solid blue; padding: 2px; margin: 5px 0;"> $\leftrightarrow \mathbf{P}_{f^*}(\mathbf{x})\mathbf{P}_{u^*}(\mathbf{x}) - \mathbf{P}_{w^*}(\mathbf{x}) = 0$ </div> - Return 0. 	$\mathcal{O}_1(\cdot)$: //k-th query <ul style="list-style-type: none"> - $r_k \leftarrow_{\\$} \mathbb{Z}_p := \mathbf{R}_k$. - $u_k = g^{r_k}$. - $\mathcal{Q}_1 = \mathcal{Q}_1 \cup \{u_k\}$. - Return u_k.
<div style="border: 1px dashed red; padding: 5px;"> $\tilde{\mathcal{E}}([f_k]_{\mathbb{F}_k}, [w_k]_{\mathbb{W}_k})$: <ul style="list-style-type: none"> // $\vec{\mathbf{X}} := (\mathbf{X}, \mathbf{Y}, \mathbf{R}_1, \dots, \mathbf{R}_{ \mathcal{Q}_1 })$. - If $\mathbf{R}_k \mathbf{P}_{f_k}[\vec{\mathbf{X}}] - \mathbf{P}_{w_k}[\vec{\mathbf{X}}] \neq 0$, then return \perp. - Return $[f_k g]$. </div>	$\mathcal{O}_2([f_k]_{\mathbb{F}_k}, [u_k]_{\mathbb{U}_k}, [w_k]_{\mathbb{W}_k})$: <ul style="list-style-type: none"> // $\mathbf{x} := (x, y, r_1, \dots, r_{ \mathcal{Q}_1 })$ - If $u_k \neq \mathcal{Q}_1$ or $(u_k, *) \in \mathcal{Q}_2$ then return \perp. - For $r_k = \log_g u_k$, if $f_k^{r_k} \neq w_k$ (i.e., $r_k \mathbf{P}_{f_k}[\mathbf{x}] - \mathbf{P}_{w_k}[\mathbf{x}] \neq 0$), then return \perp. - $m_k \leftarrow \tilde{\mathcal{E}}([f_k]_{\mathbb{F}_k}, [w_k]_{\mathbb{W}_k})$. - $v_k = u_k^x w_k^y = u_k^{x+m_k y}$. - $\mathcal{Q}_2 = \mathcal{Q}_2 \cup \{(u_k, f_k)\}$. - Return v.

Figure 7: GPS₂ problem in the AGM

where $r_\ell = \log_g u_\ell$, and m_ℓ is extracted using $\tilde{\mathcal{E}}$. At this point, \mathcal{A}_{alg} sends as the k -th \mathcal{O}_2 query input $([f_k]_{\mathbb{F}_k}, [u_k]_{\mathbb{U}_k}, [w_k]_{\mathbb{W}_k})$, which after reordering $\{u_i\}$ values depending on the \mathcal{O}_2 query's input, determines the following polynomials:

$$\begin{aligned} \mathbf{P}_{f_k}[\vec{\mathbf{X}}] &= [f_k|g] + \sum_{\ell=1}^{q_1} \mathbf{R}_\ell [f_k|u_\ell] + \sum_{\ell=1}^{k-1} \mathbf{R}_\ell [f_k|v_\ell] (\mathbf{X} + m_\ell \mathbf{Y}), \\ \mathbf{P}_{w_k}[\vec{\mathbf{X}}] &= [w_k|g] + \sum_{\ell=1}^{q_1} \mathbf{R}_\ell [w_k|u_\ell] + \sum_{\ell=1}^{k-1} \mathbf{R}_\ell [w_k|v_\ell] (\mathbf{X} + m_\ell \mathbf{Y}), \end{aligned}$$

where $\vec{\mathbf{X}} = (\mathbf{X}, \mathbf{Y}, \mathbf{R}_1, \dots, \mathbf{R}_{q_1})$. We notice that the two polynomials are represented by the fact that $\{u_\ell = g^{r_\ell}\}_{\ell=1}^{q_1}$ (set by \mathcal{O}_1) and $\{v_\ell = g^{r_\ell(x+m_\ell y)}\}_{\ell=1}^{k-1}$ (set by \mathcal{O}_2). If $w_k = f_k^{r_k}$, which is ensured by the second condition of \mathcal{O}_2 , then this equality implies that the following polynomial $\mathbf{P}_k^*[\vec{\mathbf{X}}] = \mathbf{R}_k \mathbf{P}_{f_k}[\vec{\mathbf{X}}] - \mathbf{P}_{w_k}[\vec{\mathbf{X}}]$ of degree 3 must have at least one root that is the vector of $\mathbf{x} = (x, y, r_1, \dots, r_{q_1})$. That is, $\mathbf{P}_k^*(\mathbf{x}) = 0$. There are two possible cases:

Claim 1. If $\mathbf{P}_k^*[\vec{\mathbf{X}}] = 0$, then $\tilde{\mathcal{E}}([f_k]_{\mathbb{F}_k}, [w_k]_{\mathbb{W}_k})$ works correctly.

Proof. This claim means that $\tilde{\mathcal{E}}$ is able to extract the correct m_k from the fact $\mathbf{P}_k^*[\vec{\mathbf{X}}] = 0$. As the equality $\mathbf{R}_k \mathbf{P}_{f_k}[\vec{\mathbf{X}}] = \mathbf{P}_{w_k}[\vec{\mathbf{X}}]$ holds, the coefficients (which are components of the representation vectors $\vec{\mathbf{f}}_k$ and $\vec{\mathbf{w}}_k$) are determined by equating coefficients of monomials in both sides as follows:

- $[f_k|g] = [w_k|u_k]$ (due to R_k).
- $[f_k|u_\ell] = 0$ for all $\ell \in \{1, \dots, q_1\}$ (due to $R_k R_\ell$).
- $[f_k|v_\ell] = 0$ for all $\ell \in \{1, \dots, k-1\}$ (due to $R_k R_\ell X$).
- $[w_k|u_\ell] = 0$ for all $\ell (\neq k) \in \{1, \dots, q_1\}$ (due to R_ℓ).
- $[w_k|v_\ell] = 0$ for all $\ell \in \{1, \dots, k-1\}$ (due to $R_\ell X$).

Eventually, the above equations indicates that $f_k = g^{[f_k|g]}$ and $w_k = u_k^{[w_k|u_k]}$, and thus $\tilde{\mathcal{E}}$ outputs $m_k := [f_k|g]$ as the discrete logarithm. \square

Claim 2. If $P_k^*[\vec{X}] \neq 0$, then the SDL problem is solved. ($\tilde{\mathcal{E}}$ does not work.)

Proof. By converting variables $X, Y, \{R_\ell\}$ into $S_0Z + T_0, S'_0Z + T'_0, \{S_\ell Z + T_\ell\}$, a univariate polynomial $Q_k^*[Z]$ can be defined from $P_k^*[\vec{X}]$. If $Q_k^*[Z]$ is a non-zero polynomial, it must have at least one root (including z as the solution of the SDL problem) since $P_k^*(\mathbf{x}) = Q_k^*(z) = 0$.

Based on this fact, \mathcal{C} first checks if $Q_k^*[Z] = 0$, and if so, \mathcal{C} aborts. In fact, the probability that $Q_k^*[Z] = 0$ is statistically negligible due to the following reason. As $P_k^*[\vec{X}]$ has degree 3, the leading coefficient a_3 of $Q_k^*[Z]$ can be represented by a polynomial in $\mathbb{Z}_p[S_0, S'_0, S_1, \dots, S_{q_1}]$ of degree 3 by lemma 3.5. Since $\mathbf{s} = (s_0, s'_0, s_1, \dots, s_{q_1})$ were perfectly hidden, \mathbf{s} is independently chosen at random from the view of \mathcal{A}_{alg} . Hence, the Schwartz-Zippel lemma (Lemma 3.4) implies that the probability of the coefficient a_3 evaluating 0 at \mathbf{s} is upper-bounded by $\frac{3}{p}$. This eventually implies that the inequality $\Pr[Q_k^*[Z] = 0] \leq \Pr[a_3 = 0] \leq \frac{3}{p}$. In this case, \mathcal{C} can obtain the solution z for the given SDL problem by finding roots of the equation $Q_k^*[Z] = 0$.

It is obvious that the event when $P_k^*[\vec{X}] \neq 0$ occurs is equivalent to the event that the extractor $\tilde{\mathcal{E}}$ does not work (denoted by $\neg\tilde{\mathcal{E}}$). When denoting $\Pr[P_k^*[\vec{X}] \neq 0] = \Pr[\neg\tilde{\mathcal{E}}]$, we have

$$\Pr[\neg\tilde{\mathcal{E}}] \leq \Pr[P_k^*[\vec{X}] \neq 0 \wedge Q_k^*[Z] \neq 0] + \Pr[Q_k^*[Z] = 0] \leq \mathbf{Adv}_{G, \mathcal{C}}^{SDL} + \frac{3}{p}. \quad \square$$

Let q_2 be the maximum number of oracle queries to \mathcal{O}_2 . Due to Claim 2, we can assume that $\tilde{\mathcal{E}}$ works correctly for all \mathcal{O}_2 queries under the SDL assumption. Finally, \mathcal{A}_{alg} outputs $([f^*]_{\vec{f}^*}, [u^*]_{\vec{u}^*}, [v^*]_{\vec{v}^*}, [w^*]_{\vec{w}^*})$, based on the given values $(g, \{u_\ell = g^{r_\ell}\}_{\ell=1}^{q_1}, \{v_\ell = g^{r_\ell(x+m_\ell y)}\}_{\ell=1}^{q_2})$ where $q_2 \leq q_1$, and m_ℓ is extracted using $\tilde{\mathcal{E}}$. For $\vec{X} = (X, Y, R_1, \dots, R_{q_1})$, consider polynomials in $\mathbb{Z}_p[\vec{X}]$ corresponding to the output of \mathcal{A}_{alg} , which can be represented by

$$\begin{aligned} P_{f^*}[\vec{X}] &= [f^*|g] + \sum_{\ell=1}^{q_1} [f^*|u_\ell]R_\ell + \sum_{\ell=1}^{q_2} [f^*|v_\ell](X + m_\ell Y)R_\ell, \\ P_{u^*}[\vec{X}] &= [u^*|g] + \sum_{\ell=1}^{q_1} [u^*|u_\ell]R_\ell + \sum_{\ell=1}^{q_2} [u^*|v_\ell](X + m_\ell Y)R_\ell, \\ P_{v^*}[\vec{X}] &= [v^*|g] + \sum_{\ell=1}^{q_1} [v^*|u_\ell]R_\ell + \sum_{\ell=1}^{q_2} [v^*|v_\ell](X + m_\ell Y)R_\ell, \\ P_{w^*}[\vec{X}] &= [w^*|g] + \sum_{\ell=1}^{q_1} [w^*|u_\ell]R_\ell + \sum_{\ell=1}^{q_2} [w^*|v_\ell](X + m_\ell Y)R_\ell. \end{aligned}$$

If \mathcal{A}_{alg} wins, the conditions (w1), (w2), (w3), and (w4) (described in Figure 7) are satisfied. In that case, we define the following events:

- Event E_1 : $\mathbf{G}_{\mathcal{BG}, \mathcal{A}_{alg}}^{GPS_2} = 1 \wedge \tilde{\mathcal{E}}$ works correctly.
- Event E_2 : $\mathbf{P}_{w_2}^*[\vec{\mathbf{X}}] = \mathbf{P}_{v^*}[\vec{\mathbf{X}}] - (\mathbf{P}_{u^*}[\vec{\mathbf{X}}] \cdot \mathbf{X} + \mathbf{P}_{w^*}[\vec{\mathbf{X}}] \cdot \mathbf{Y})$ is the zero-polynomial.
- Event E_3 : $\mathbf{P}_{w_3}^*[\vec{\mathbf{X}}] = \mathbf{P}_{w^*}[\vec{\mathbf{X}}] - \mathbf{P}_{u^*}[\vec{\mathbf{X}}] \cdot \mathbf{P}_{f^*}[\vec{\mathbf{X}}]$ is the zero-polynomial,

where the polynomials $\mathbf{P}_{w_2}^*[\vec{\mathbf{X}}]$ and $\mathbf{P}_{w_3}^*[\vec{\mathbf{X}}]$ are from the winning conditions (w2) and (w3), respectively.

Claim 3. $\Pr[E_1 \wedge E_2 \wedge E_3] = 0$.

Proof. Assume that $E_1 \wedge E_2 \wedge E_3$ happens. From $\mathbf{P}_{w_2}^*[\vec{\mathbf{X}}] = 0$, the equality $\mathbf{P}_{v^*}[\vec{\mathbf{X}}] = \mathbf{P}_{u^*}[\vec{\mathbf{X}}] \cdot \mathbf{X} + \mathbf{P}_{w^*}[\vec{\mathbf{X}}] \cdot \mathbf{Y}$ holds. We can see that the following coefficients are zero by equating the coefficients of monomials on both sides as follows:

- $[\mathbf{u}^*|v_k] = 0$ and $[\mathbf{w}^*|v_k] = 0, \forall k \in \{1, \dots, q_2\}$ (due to monomials of degree 3).
- $[\mathbf{u}^*|g] = 0$ and $[\mathbf{w}^*|g] = 0$ (due to \mathbf{X}).
- $[\mathbf{v}^*|u_\ell] = 0, \forall \ell \in \{1, \dots, q_1\}$ (due to $\{\mathbf{R}_\ell\}$).
- $[\mathbf{v}^*|g] = 0$ (due to the constant).
- $[\mathbf{u}^*|u_\ell] = [\mathbf{w}^*|u_\ell] = 0, \forall \ell \in \{q_2 + 1, \dots, q_1\}$ (due to $\{\mathbf{X}\mathbf{R}_\ell\}$),

which shows that $\mathbf{P}_{u^*}[\vec{\mathbf{X}}] = \sum_{\ell=1}^{q_2} [\mathbf{u}^*|u_\ell] \mathbf{R}_\ell$ and $\mathbf{P}_{w^*}[\vec{\mathbf{X}}] = \sum_{\ell=1}^{q_2} [\mathbf{w}^*|u_\ell] \mathbf{R}_\ell$ whose degrees should be all 1. In that case, E_3 ensures that the equality $\mathbf{P}_{u^*}[\vec{\mathbf{X}}] \mathbf{P}_{f^*}[\vec{\mathbf{X}}] = \mathbf{P}_{w^*}[\vec{\mathbf{X}}]$ holds, implying that the degree of \mathbf{P}_{f^*} must be 0 (i.e., it is a constant polynomial). We can thus consider $\mathbf{P}_{f^*}[\vec{\mathbf{X}}] = [f^*|g]$, which is now denoted by m^* . Recall that $\mathbf{P}_{v^*}[\vec{\mathbf{X}}] = \sum_{\ell=1}^{q_2} [\mathbf{v}^*|v_\ell] (\mathbf{X} + m_\ell \mathbf{Y}) \mathbf{R}_\ell$. Based on these facts, the polynomial $\mathbf{P}_{w_2}^*[\vec{\mathbf{X}}]$ can be rewritten as follows:

$$\begin{aligned} \mathbf{P}_{w_2}^*[\vec{\mathbf{X}}] &= \mathbf{P}_{v^*}[\vec{\mathbf{X}}] - \mathbf{P}_{u^*}[\vec{\mathbf{X}}] (\mathbf{X} + \mathbf{P}_{f^*}[\vec{\mathbf{X}}] \mathbf{Y}) \\ &= \sum_{\ell=1}^{q_2} c_\ell (\mathbf{X} + m_\ell \mathbf{Y}) \mathbf{R}_\ell - \sum_{\ell=1}^{q_2} c_\ell (\mathbf{X} + m^* \mathbf{Y}) \mathbf{R}_\ell, \end{aligned}$$

where $c_\ell = [\mathbf{v}^*|v_\ell] = [\mathbf{u}^*|u_\ell]$. If $\{c_\ell\}_{\ell=1}^{q_2}$ are all 0, then $\mathbf{P}_{u^*}[\vec{\mathbf{X}}]$ is the zero-polynomial, which is a contradiction to the condition (w1). Therefore, there must be at least one non-zero c_k for some $k \in \{1, \dots, q_2\}$. Say only $c_k \neq 0$, in which case $\mathbf{P}_{w_2}^*[\vec{\mathbf{X}}] = c_k (m_k - m^*) \mathbf{Y} \mathbf{R}_{k^*}$. By condition (w1), $f^* \neq 1$ leads to $m^* \neq 0$. Moreover, by condition (w4), $m_k \neq m^*$, eventually resulting in $\mathbf{P}_{w_2}^*[\vec{\mathbf{X}}] \neq 0$. This is the contradiction to the event E_2 . \square

Claim 4. $\Pr[E_1 \wedge \neg E_2] + \Pr[E_1 \wedge E_2 \wedge \neg E_3] \leq \mathbf{Adv}_{\mathcal{G}, \mathcal{C}}^{SDL} + \frac{7}{p}$.

Proof. If E_2 does not happen, \mathcal{C} can build a univariate polynomial $\mathbf{Q}_{w_2}^*[Z]$ by changing the variables of $\mathbf{P}_{w_2}^*$ into $sZ + t$ for some $s, t \in \mathbb{Z}_p$. By a similar argument to the proof of Claim 2, we see that $\mathbf{Q}_{w_2}^*(z) = 0$ because of the condition (w2), and thus one of the roots of the polynomial $\mathbf{Q}_{w_2}^*[Z]$ has to be the solution of the SDL problem. In addition, by lemmas 3.4 and 3.5, the probability that $\mathbf{Q}_{w_2}^*[Z] = 0$ (i.e., \mathcal{C} aborts) is at most $\frac{3}{p}$. Similarly, if $\mathbf{P}_{w_3}^*[\vec{\mathbf{X}}] \neq 0$ (i.e., if E_3 does not occur), then the non-zero polynomial $\mathbf{Q}_{w_3}^*[Z]$ gives the SDL solution with a probability of at least $1 - \frac{4}{p}$. That is, \mathcal{C} aborts with a probability of $\frac{4}{p}$ at most. Thereby, in both cases, we have $\Pr[\mathcal{C} \text{ aborts}] \leq \frac{7}{p}$.

As long as \mathcal{C} does not abort, \mathcal{C} can obtain the solution z of the SDL problem by finding roots of $\mathbf{Q}_{w^2}^*[Z] \neq 0$ or $\mathbf{Q}_{w^3}^*[Z] \neq 0$, which gives the following equality:

$$\begin{aligned} \Pr[E_1] &= \Pr[E_1 \wedge \neg E_2] + \Pr[E_1 \wedge E_2 \wedge \neg E_3] + \Pr[E_1 \wedge E_2 \wedge E_3] \\ &= \Pr[E_1 \wedge \neg E_2] + \Pr[E_1 \wedge E_2 \wedge \neg E_3] \leq \mathbf{Adv}_{\mathcal{G}, \mathcal{C}}^{SDL} + \frac{7}{p}. \quad \square \end{aligned}$$

Using the fact that $\Pr[E_1] = \Pr[\mathbf{G}_{\mathcal{BG}, \mathcal{A}_{alg}}^{GPS_2} = 1 \wedge \tilde{\mathcal{E}}]$, we conclude that

$$\begin{aligned} \mathbf{Adv}_{\mathcal{BG}, \mathcal{A}_{alg}}^{GPS_2} &= \Pr[\mathbf{G}_{\mathcal{BG}, \mathcal{A}_{alg}}^{GPS_2} = 1 \wedge \neg \tilde{\mathcal{E}}] + \Pr[\mathbf{G}_{\mathcal{BG}, \mathcal{A}_{alg}}^{GPS_2} = 1 \wedge \tilde{\mathcal{E}}] \\ &\leq \Pr[\neg \tilde{\mathcal{E}}] + \Pr[\mathbf{G}_{\mathcal{BG}, \mathcal{A}_{alg}}^{GPS_2} = 1 \wedge \tilde{\mathcal{E}}] \\ &\leq \mathbf{Adv}_{\mathcal{G}, \mathcal{C}}^{SDL} + \frac{3}{p} + \mathbf{Adv}_{\mathcal{G}, \mathcal{C}}^{SDL} + \frac{7}{p} \\ &= 2\mathbf{Adv}_{\mathcal{G}, \mathcal{C}}^{SDL} + \frac{10}{p}, \end{aligned}$$

as required. □

4 Our Dynamic Group Signature Scheme

4.1 Idea Behind Removing Knowledge Extractors

We explain how we can eliminate knowledge extractors throughout our security analysis without resorting to a common reference string (CRS). We first recall that knowledge extractors were mostly required to prove the security notions of traceability and non-frameability. At a high level, our \mathcal{DGS} scheme is based on a variant [40] of the PS signature. For a signing key $(x, y) \in \mathbb{Z}_p^2$, a signature on a (committed) message $\alpha \in \mathbb{Z}_p$ consists of $(u, v = u^{x+\alpha y}, w = u^\alpha) \in \mathbb{G}_1^3$, which is also randomizable as it is the case with the original PS signature. The main difference with [45] and common variants is that this signature is no longer generated from α or g^α but from a tuple (g^α, u, u^α) for some u that we will define below. In our \mathcal{DGS} scheme, the issuer uses the secrets (x, y) to issue such a (modified) PS signature (especially, the v value) for a prospective user through a joining protocol.

Roughly speaking, breaking traceability requires an adversary to generate a new (modified) PS signature $(u^*, v^* = (u^*)^{x+\alpha^* y}, w^* = (u^*)^{\alpha^*})$, where α^* has not been previously used by another user. To ensure that α is fresh during the joining protocol, we introduce an additional element $f = g^\alpha$ and ask the user to provide a NIZK proof π_1 with respect to α in order to prove that $f = g^\alpha$ and $w = u^\alpha$. Since g is a fixed public element, the issuer easily checks that f (and so α) is fresh at the time of joining. In the traceability proof, after checking the validity of π_1 and the freshness, the tuple (f, u, w) is queried to the oracle \mathcal{O}_2 provided by the \mathcal{GPS}_2 problem to get the appropriate value v . As a valid tuple against the \mathcal{GPS}_2 problem must contain the element $f^* = g^{\alpha^*}$ we need to embed this element in our group signatures. In our construction, this will be done by adding an El Gamal encryption of this element, that is, a pair $(\mathbf{c}_0^* = g^s, \mathbf{c}_1^* = f^* D_1^s)$ for some public D_1 and random secret s . An interesting outcome of this additional pair is that it leads to constant time opening if one sets D_1 as the opener's public key.

At this stage, we get a signature consisting of $(u^*, v^*, w^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \pi_2)$, where π_2 is a NIZK proof ensuring that every element is well formed. If the adversary breaks traceability and thus generates an untraceable signature with respect to a new $f^* = g^{\alpha^*}$, then we can obtain a new (modified) PS signature $(u^*, v^* =$

$(u^*)^{x+\alpha^*y}, w^* = (u^*)^{\alpha^*}$) and the relevant f^* . Especially, f^* is easily recovered, since the opener is assumed to be partially corrupted in the traceability game. The relation $v^* = (u^*)^{x+\alpha^*y}$ comes from the pairing equation test as in the PS signature, and the relations $w^* = (u^*)^{\alpha^*}$ and $f^* = g^{\alpha^*}$ come from the (statistical) simulation soundness of the NIZK proof π_2 . The important point is that, during the above analysis, neither extracting α^* nor α^* itself are required, so that we can obtain a new and valid forgery of the (modified) PS signature *without extracting the witnesses*. Eventually, this allows us to remove knowledge extractors from the traceability proof.

In the case of the non-frameability game, the adversary is asked to generate a signature that GOpen links back to some user without knowing this user's secret α^* . As explained in the introduction, non-frameability of many existing \mathcal{DGS} schemes in the ROM is proven under the hardness of the SDL problem. More specifically, the reduction extracts the discrete logarithm α^* of the SDL problem by rewinding the adversary. Our idea behind removing knowledge extractors is to extend the element $f = g^\alpha$ above (which essentially acts as the member's public key) with $f_2 = h^\alpha$, for some different (public) basis h . This means that a signature generated with α^* now contains $(u^*, w^* = (u^*)^{\alpha^*}, v^* = (u^*)^x(w^*)^y, g^s, f^*D_1^s, f_2^*D_2^s, \pi_2)$, where D_2 is another public key of the opener. Here again, π_2 proves consistency of the resulting signature and in particular that α^* used to generate w^* is such that $f^* = g^{\alpha^*}$ and $f_2^* = h^{\alpha^*}$. But now, using the lossy key technique, we can replace at some point in the proof the element f_2^* by some random element in \mathbb{G}_1 . This cannot be detected by the adversary under the $\text{XDH}_{\mathbb{G}_1}$ assumption. Once this is done, the only way for the adversary to accuse the owner of α^* of having produced a signature σ is to encrypt g^{α^*} and the random f_2^* while proving that both elements have the same discrete logarithm with respect to (g, h) . This clearly breaks the (statistical) simulation soundness of π_2 , which is assumed to be impossible. We can therefore prove non-frameability of our scheme without needing to rewind the adversary.

At first sight, our strategy to remove knowledge extractors has led us to add the elements $(g^s, f^*D_1^s, f_2^*D_2^s)$ to our signature. However, we would like to stress that this does not really increase the size of our group signature as the latter must anyway contain an IND-CCA2 encryption of the signer's identifier to achieve constant time opening while being able to answer opening queries. In our setting, the most classical solution would be to use the Naor-Yung transformation [43] on El Gamal ciphertexts, which would result in something similar to what we get with $(g^s, f^*D_1^s, f_2^*D_2^s)$ combined with π_2 . Actually, we are even more efficient than this generic transform as we use randomness-sharing of the ciphertext and merge the corresponding zero-knowledge proof with π_2 . This way, we get a scheme proven secure without knowledge extractor which is still competitive with the most efficient solutions from the state-of-the-art (see Section 6).

4.2 Non-Interactive Zero Knowledge Proofs

We employ NIZK proofs obtained by the Fiat–Shamir transformation [28] in the ROM, which do not require a CRS generation. These NIZK proofs are defined with a NP-language $\mathcal{L}_{\mathcal{R}} = \{Y \in \{0, 1\}^* \mid \exists \omega : (Y, \omega) \in \mathcal{R}\}$, where $\mathcal{R} \in \{0, 1\}^* \times \{0, 1\}^*$ is an efficiently recognizable relation. If $Y \in \mathcal{L}_{\mathcal{R}}$, then we call such a Y a *true statement* for a witness ω ; otherwise, a *false statement*. We denote a NIZK proof π by $\pi = \text{NIZK}\{\omega : (Y, \omega) \in \mathcal{R}\}$. Such a proof π is generated by the $\text{Prove}(Y, \omega)$ algorithm, and the $\text{Vf}(Y, \pi)$ algorithm outputs 1 if Y is true for ω ; otherwise, it outputs 0.

Let \mathbb{G}_1 be a cyclic group of a type-3 bilinear group, and let $\tilde{H} : \{0, 1\}^* \rightarrow \mathbb{Z}_p$ be a hash function. We use the following three NIZK instantiations that essentially rely on the well-known NIZK proof of a generalized representation [21] of discrete logarithms. For group elements in \mathbb{G}_1 , our NIZK proofs are defined as follows:

- NIZK₁ for the relation $\mathcal{R}_1 = \{(Y = (g, h, u, f_1, f_2, w), \omega = \alpha) \in \mathbb{G}_1^6 \times \mathbb{Z}_p\}$ such that $f_1 = g^\alpha \wedge f_2 = h^\alpha \wedge w =$

u^α . A proof π is composed of $(c, s) \in \mathbb{Z}_p^2$, where $c = \tilde{H}(Y, g^k, h^k, u^k)$ and $s = k - c\alpha$ for $k \leftarrow_{\mathcal{S}} \mathbb{Z}_p$. If $c = \tilde{H}(Y, g^s f_1^c, h^s f_2^c, u^s w^c)$ holds, then π is valid.

- NIZK₂ for the relation $\mathcal{R}_2 = \{(Y = (\tilde{u}, g, h, D_1, D_2, \tilde{w}, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2), \omega = (\alpha, s)) \in \mathbb{G}_1^9 \times \mathbb{Z}_p^2\}$ such that $\tilde{w} = \tilde{u}^\alpha \wedge \mathbf{c}_0 = g^s \wedge \mathbf{c}_1 = g^\alpha D_1^s \wedge \mathbf{c}_2 = h^\alpha D_2^s$. A proof is $\pi = (c, s_1, s_2) \in \mathbb{Z}_p^3$, where $c = \tilde{H}(Y, \tilde{u}^{k_1}, g^{k_2}, g^{k_1} D_1^{k_2}, h^{k_1} D_2^{k_2})$, $s_1 = k_1 - c\alpha$ and $s_2 = k_2 - cs$ for $k_1, k_2 \leftarrow_{\mathcal{S}} \mathbb{Z}_p$. The proof is valid only when $c = \tilde{H}(Y, \tilde{u}^{s_1} \tilde{w}^c, g^{s_2} \mathbf{c}_0^c, g^{s_1} D_1^{s_2} \mathbf{c}_1^c, h^{s_1} D_2^{s_2} \mathbf{c}_2^c)$ holds.
- NIZK₃ for the relation $\mathcal{R}_3 = \{(Y = (g, \mathbf{c}_0, \mathbf{c}_1, f_1, f_2, D_1, D_2), \omega = (d_1, d_2)) \in \mathbb{G}_1^7 \times \mathbb{Z}_p^2\}$ such that $\mathbf{c}_1 f_1^{-1} = \mathbf{c}_0^{d_1} \wedge D_1 = g^{d_1} \wedge \mathbf{c}_2 f_2^{-1} = \mathbf{c}_0^{d_2} \wedge D_2 = g^{d_2}$. A proof $\pi = (c, s_1, s_2) \in \mathbb{Z}_p^3$ is computed by $c = \tilde{H}(Y, (\mathbf{c}_1 f_1^{-1})^{k_1}, g^{k_1}, (\mathbf{c}_2 f_2^{-1})^{k_2}, g^{k_2})$, $s_1 = k_1 - cd_1$ and $s_2 = k_2 - cd_2$ for $k_1, k_2 \leftarrow_{\mathcal{S}} \mathbb{Z}_p$, and it is valid when $c = \tilde{H}(Y, (\mathbf{c}_1 f_1^{-1})^{s_1} \mathbf{c}_0^c, D_1^{s_1} g^c, (\mathbf{c}_2 f_2^{-1})^{s_2} \mathbf{c}_0^c, D_2^{s_2} g^c)$ holds.

Notice that our NIZK proofs above are formalized as the true or false statements by which the *simulation soundness* property can be defined. According to the Faust et al.'s work (especially, Theorem 2 in [27]), under the assumption that \tilde{H} is modeled as a random oracle, those NIZK proofs satisfy the following properties: informally, (*completeness*) a proof generated by an honest prover should be verified successfully; (*zero-knowledge*) there is a zero-knowledge simulator \mathcal{S} that, not knowing the witness, simulates a valid proof statistically indistinguishable from a real one; and (*simulation soundness*) a cheating prover is unable to generate a new proof for a false statement even when receiving simulated proofs for true or false statements. The latter property is proved in [27] without rewinding the adversary.

In our second construction, the second NIZK proof will be transformed into a signature variant of the NIZK proof by additionally including a message m into the hash function \tilde{H} . To avoid confusion, we denote this variant by $\text{sNIZK}_2\{\omega : (Y, \omega) \in \mathcal{R}_2\}(m)$ for a message $m \in \{0, 1\}^*$.

4.3 Construction

Our \mathcal{DGS} scheme makes use of a digital signature scheme $\mathcal{DS} = \{\text{DSKg}, \text{DSSig}, \text{DSVf}\}$ and NIZK proofs $\{\text{NIZK}_1, \text{sNIZK}_2, \text{NIZK}_3\}$ described in Section 2 as follows:

Setup. The setup algorithm works as follows:

1. It runs GrGen to produce a type-3 bilinear group $\mathcal{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, e)$.
2. It chooses a cryptographic hash function $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ and sets $h = H(g)$.
3. The public parameter is $pp = (\mathcal{BG}, H, h)$, which is *publicly verifiable*.⁴

IKg. The issuer key generation works as follows:

1. It chooses $x, y \leftarrow_{\mathcal{S}} \mathbb{Z}_p$.
2. It computes $\hat{X} = \hat{g}^x$ and $\hat{Y} = \hat{g}^y$.
3. The issuer's secret key is $ik = (x, y) \in \mathbb{Z}_p^2$ and public key is $ipk = (\hat{X}, \hat{Y}) \in \mathbb{G}_2^2$.

OKg. The opener key generation works as follows:

1. It picks $d_1, d_2 \leftarrow_{\mathcal{S}} \mathbb{Z}_p$.

⁴According to Bellare et al. [5], our setup algorithm can be non-trusted if a bilinear-group generation algorithm is deterministic and public. Also, the element h is computable by anyone and thus publicly verifiable from the generated bilinear group.

2. It computes $D_1 = g^{d_1}$ and $D_2 = g^{d_2}$.
3. The opener's secret and public keys are $ok = (d_1, d_2) \in \mathbb{Z}_p^2$ and $opk = (D_1, D_2) \in \mathbb{G}_1^2$, respectively.

The group public key is $gpk = (pp, ipk, opk)$.

UKg. User i obtains its key pair (upk_i, usk_i) by running DSKg.

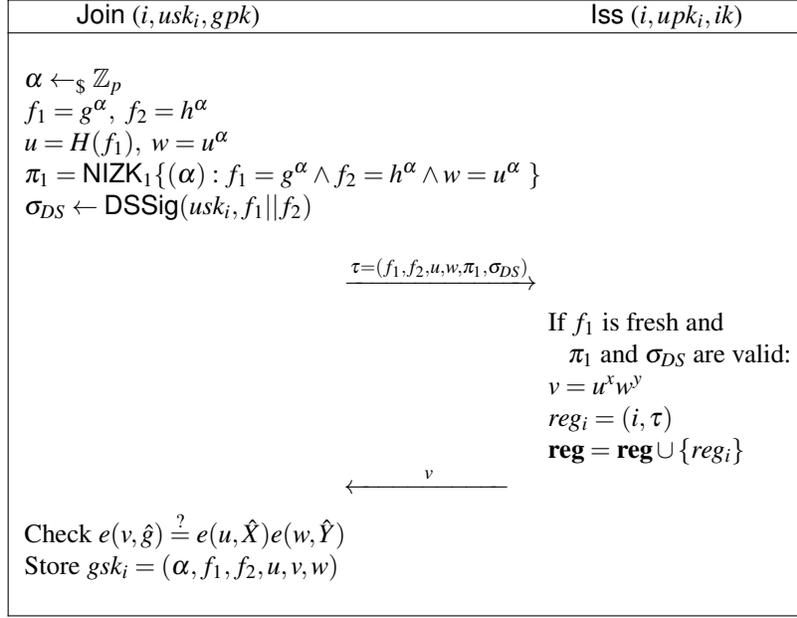


Figure 8: Overview of the GJoin protocol

GJoin. As shown in Figure 8, the user i and the issuer execute the group-joining protocol using the **Join** and **Iss** algorithms, respectively.

1. **Join:** The user performs the following:

- (1) After choosing $\alpha \leftarrow_{\$} \mathbb{Z}_p$, the user computes $f_1 = g^\alpha, f_2 = h^\alpha, u = H(f_1)$ and $w = u^\alpha$.
- (2) The proof π_1 is generated as

$$\pi_1 = \text{NIZK}_1\{(\alpha) : f_1 = g^\alpha \wedge f_2 = h^\alpha \wedge w = u^\alpha\} \in \mathbb{Z}_p^2.$$

- (3) The user obtains σ_{DS} by running $\text{DSSig}(usk_i, f_1 || f_2)$.
- (4) The tuple $\tau = (f_1, f_2, u, w, \pi_1, \sigma_{DS})$ is sent to the issuer.
2. **Iss:** Given the above tuple, the issuer computes $u = H(f_1)$ and checks that:
 - (1) f_1 has never appeared in a previous or current joining session.
 - (2) π_1 is valid with respect to (g, h, u, f_1, f_2, w) .
 - (3) σ_{DS} is valid on $f_1 || f_2$ under upk_i .
 - (4) If all the above conditions are satisfied, it computes $v = u^x w^y$ and adds $reg_i = (i, \tau)$ to the list **reg**.

(5) v is sent to the user.

3. **Join:** If the following equation $e(v, \hat{g}) = e(u, \hat{X})e(w, \hat{Y})$ holds, the user stores its group signing key

$$gsk_i = (\alpha, f_1 = g^\alpha, f_2 = h^\alpha, u, v = u^{x+y\alpha}, w = u^\alpha) \in \mathbb{Z}_p \times \mathbb{G}_1^5.$$

GSig. Given a message m to be signed, the user invokes this algorithm as follows:

1. It picks $r \leftarrow_{\S} \mathbb{Z}_p$, and it computes $\tilde{u} = u^r, \tilde{v} = v^r$ and $\tilde{w} = w^r$.
2. It uses $s \leftarrow_{\S} \mathbb{Z}_p$ to compute $(\mathbf{c}_0 = g^s, \mathbf{c}_1 = f_1 D_1^s, \mathbf{c}_2 = f_2 D_2^s) \in \mathbb{G}_1^3$.
3. The proof π_2 is generated as

$$\pi_2 = \text{SNIZK}_2\{(\alpha, s) : \tilde{w} = \tilde{u}^\alpha \wedge \mathbf{c}_0 = g^s \wedge \mathbf{c}_1 = g^\alpha D_1^s \wedge \mathbf{c}_2 = h^\alpha D_2^s\}(m) \in \mathbb{Z}_p^3.$$

4. The group signature is $\sigma = (\tilde{u}, \tilde{v}, \tilde{w}, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \pi_2) \in \mathbb{G}_1^6 \times \mathbb{Z}_p^3$.

GVf. A verifier checks if a signature $\sigma = (\tilde{u}, \tilde{v}, \tilde{w}, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \pi_2)$ is valid on a message m as follows:

1. It checks if π_2 is valid with respect to $(\tilde{u}, g, h, D_1, D_2, \tilde{w}, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2)$ and m .
2. If so, it checks if $e(\tilde{v}, \hat{g}) = e(\tilde{u}, \hat{X})e(\tilde{w}, \hat{Y})$.
3. If all the above conditions hold, it outputs 1; otherwise, 0.

GOpen. Given a valid σ on a message m , the opener does the following:

1. For $(\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2)$ in σ , it obtains $f_1' = \mathbf{c}_1 \mathbf{c}_0^{-d_1}$ and $f_2' = \mathbf{c}_2 \mathbf{c}_0^{-d_2}$ by using $ok = (d_1, d_2)$.
2. If there is no $reg_i = (i, \tau) \in \mathbf{reg}$, where $\tau = (f_1, f_2, u, w, \pi_1, \sigma_{DS})$ such that $f_1 = f_1', f_2 = f_2'$ and a valid π_1 with respect to (g, h, u, f_1, f_2, w) , then it outputs \perp .
3. If there is one more such $\{reg_i\}$, then it outputs \perp .
4. Otherwise, the proof π_3 is generated as

$$\pi_3 = \text{NIZK}_3\{(d_1, d_2) : \mathbf{c}_1 \cdot f_1^{-1} = \mathbf{c}_0^{d_1} \wedge D_1 = g^{d_1} \wedge \mathbf{c}_2 \cdot f_2^{-1} = \mathbf{c}_0^{d_2} \wedge D_2 = g^{d_2}\} \in \mathbb{Z}_p^3.$$

5. It outputs the identity i along with $\Pi = (f_1, f_2, \sigma_{DS}, \pi_3)$.

GJudge. Given $(m, \sigma, gpk, i, upk_i, \Pi)$ where σ is a valid signature and $\Pi = (f_1, f_2, \sigma_{DS}, \pi_3)$, the judge algorithm does the following:

1. It checks if π_3 is valid with respect to $(f_1, f_2, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2)$ in σ and (D_1, D_2) in gpk .
2. It also checks if $\text{DSVf}(upk_i, f_1 || f_2, \sigma_{DS}) = 1$.
3. It outputs 1 if all the above conditions hold; otherwise, 0.

It is easy to see that our \mathcal{DGS} works correctly because of the completeness of the NIZKs and the correctness of the (variant of) PS signature and \mathcal{DS} . In addition, as in [40], our \mathcal{DGS} scheme can provide pairing-batch computations (i.e., the pairing equation of Step 2 in GVf can be batched) to verify n group signatures from distinct signers. We omit the details of the correctness and the pairing-batch verification algorithm in this paper.

5 Security Proofs

Let $\mathcal{L}_h, \mathcal{L}_c, \mathcal{L}_{sk}, \mathcal{L}_\sigma$ and \mathcal{L}_{ch} be initialized as empty sets. Let $\mathcal{BG} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g, \hat{g}, e)$ be published as a public parameter. For each security proof, we create a sequence of hybrid games where the original game \mathbf{G}_0 is gradually changed into the final game \mathbf{G}_F .

5.1 Non-frameability

Theorem 5.1. *Let $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ be modeled as a random oracle. Our \mathcal{DGS} is non-frameable under the zero-knowledge property of both NIZK_1 and sNIZK_2 , the simulation soundness of both sNIZK_2 and NIZK_3 , the EUF-CMA of \mathcal{DS} , and the $\text{XDH}_{\mathbb{G}_1}$ assumption.*

Proof. We show how to change \mathbf{G}_0 into \mathbf{G}_F under the security properties of underlying primitives. We next show that, in \mathbf{G}_F , the probability that \mathcal{A} breaks the non-frameability becomes zero.

- \mathbf{G}_0 : This is the original non-frameability game $\mathbf{G}_{\mathcal{DGS}, \mathcal{A}}^{\text{Nf}}$.
- \mathbf{G}_1 : This is the same game as \mathbf{G}_0 , except that \mathcal{C} simulates NIZK_1 and sNIZK_2 proofs using the corresponding simulators. By the (statistical) zero-knowledge property of the NIZKs, \mathbf{G}_0 is indistinguishable from \mathbf{G}_1 .
- \mathbf{G}_2 : This is the same game as \mathbf{G}_1 , except that \mathcal{C} aborts if \mathcal{A} generates a proof of either sNIZK_2 or NIZK_3 on a false statement. By the (statistical) simulation soundness property of the sNIZK_2 or NIZK_3 , \mathbf{G}_1 is indistinguishable from \mathbf{G}_2 .
- \mathbf{G}_3 : This is the same game as \mathbf{G}_2 , except that \mathcal{C} guesses a target (honest) user i^* among the q number of the SndToU queries. \mathcal{C} chooses $k \leftarrow_{\$} \{1, \dots, q\}$ and considers the k -th appeared identity in SndToU queries as i^* . If \mathcal{A} queries i^* to USK oracle or outputs a forgery σ^* that cannot be linked to i^* , then \mathcal{C} aborts. The probability that \mathcal{C} does not abort is exactly $1/q$.
- \mathbf{G}_4 : This is the same game as \mathbf{G}_3 , except that $f_1^* = g^{\alpha^*}$ and $f_2^* = h^{\alpha^*}$ initially stored in the i^* -th entry reg^* are not modified by WReg queries. Under the EUF-CMA property of \mathcal{DS} , \mathbf{G}_3 is indistinguishable from \mathbf{G}_4 .
- \mathbf{G}_F : This is the same game as \mathbf{G}_4 , except that reg_{i^*} now contains a pair (f_1^*, f_2^*) such that $\log_g f_1^* \neq \log_h f_2^*$. Since \mathbf{G}_2 we have excluded proofs of false statements. Therefore, a successful adversary against non-frameability must have produced a group signature σ^* along with a proof Π^* such that
 - σ^* contains an El Gamal encryption of (f_1^*, f_2^*) (otherwise GJudge would reject Π^*).
 - σ^* contains a proof that $\log_g f_1^* = \log_h f_2^*$.

As it is impossible to fulfil both requirements, \mathcal{A} cannot succeed in this last game.

Claim 1: Under the $\text{XDH}_{\mathbb{G}_1}$ assumption, \mathbf{G}_4 is indistinguishable from \mathbf{G}_F .

Proof of Claim 1. Let $(g, A = g^a, B = g^b, T)$ be an instance of the $\text{XDH}_{\mathbb{G}_1}$ problem. \mathcal{C} first sets up the public parameter that includes $h = H(g)$ as it is in the real scheme, and then receives $\text{ipk} = (\hat{X}, \hat{Y})$ and $\text{opk} = (D_1, D_2)$ from \mathcal{A} . All oracle queries are managed by \mathcal{C} as follows:

- **Hash- H .** For a new input $n (\neq g)$, \mathcal{C} chooses $\delta_n \in \mathbb{Z}_p$ and computes $\mu = g^{\delta_n}$. For the special input g , \mathcal{C} sets $H(g) = B = h$, leading to $\mu = B$. \mathcal{C} returns μ and adds either (n, δ_n, μ) or (g, \perp, μ) to \mathcal{L}_H . If n has queried before, \mathcal{C} returns μ stored in \mathcal{L}_H .

- **SndToU**. For $i \neq i^*$, \mathcal{C} obtains a pair (usk_i, upk_i) by **DSKg**. While executing the **GJoin** protocol, i 's secret $\alpha \leftarrow_{\mathcal{S}} \mathbb{Z}_p$ is chosen by \mathcal{C} . At the end of the protocol, \mathcal{C} obtains $gsk_i = (\alpha, f_1, f_2, u, v, w)$. For $i = i^*$, \mathcal{C} implicitly sets $\alpha^* = a = \log_g A$ and sends \mathcal{A} the following tuple:

$$(f_1^* = A, f_2^* = T, u^* = g^{\delta_{f_1^*}}, w^* = A^{\delta_{f_1^*}}, \pi_1^*, \sigma_{DS}^*),$$

where π_1^* is a simulated proof with respect to the unknown α^* , and σ_{DS}^* is a digital signature on the message $f_1^* || f_2^*$. After \mathcal{A} sends v^* to \mathcal{C} in response to the tuple, \mathcal{C} stores $gsk_{i^*} = (\perp, f_1^*, f_2^*, u^*, v^*, w^*)$. Note that if $T = g^{ab}$, $\log_g f_1^* = \log_h f_2^*$, which is \mathbf{G}_4 , and if $T \neq g^{ab}$, $\log_g f_1^* \neq \log_h f_2^*$, which is \mathbf{G}_F . Finally, the list \mathcal{L}_h is updated by adding i .

- **USK**. For $i \neq i^*$, \mathcal{C} returns (usk_i, gsk_i) and updates \mathcal{L}_{sk} by adding i .
- **Sig**. For $i \neq i^*$, \mathcal{C} uses gsk_i to generate σ on m as is the case in the real scheme, except for a simulated proof π_2 . For $i = i^*$, \mathcal{C} computes $((u^*)^r, (v^*)^r, (w^*)^r)$ by using gsk_{i^*} and $r \leftarrow_{\mathcal{S}} \mathbb{Z}_p$. In addition, it computes $\mathbf{c}_0 = g^s, \mathbf{c}_1 = A \cdot D_1^s$ and $\mathbf{c}_2 = T \cdot D_2^s$ for the randomly chosen $s \leftarrow_{\mathcal{S}} \mathbb{Z}_p$. Then, σ contains those values along with π_2 that is a simulated proof. Note that the returning σ passes the **GVf** algorithm and is opened with reg_{i^*} even though T is random. Finally, the list \mathcal{L}_σ is updated by adding (i, m, σ) .

\mathcal{A} finally outputs $(m^*, \sigma^*, i^*, \Pi^*)$, where $\sigma^* = (\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \pi_2^*)$ and $\Pi^* = (f_1^*, f_2^*, \sigma_{DS}^*, \pi_3^*)$. If $T = g^{ab}$, \mathcal{C} simulates \mathbf{G}_4 ; otherwise, \mathcal{C} simulates \mathbf{G}_F . Any change of behaviour of \mathcal{A} can thus be used to break the $\text{XDH}_{\mathbb{G}_1}$ assumption.

We now show that \mathcal{A} can never win the security game in \mathbf{G}_F . Depending on whether either π_2^* or π_3^* is valid, there are two possible cases of the forgery:

- (1) Case when π_2^* is valid: Because of the simulation-soundness of **sNIZK₂**, the validity indicates that $\tilde{w}^* = \tilde{u}^{\alpha^*} \wedge \mathbf{c}_0^* = g^s \wedge \mathbf{c}_1^* = g^{\alpha^*} D_1^s \wedge \mathbf{c}_2^* = h^{\alpha^*} D_2^s$, where $\alpha^* = \log_g A$ is the secret exponent of i^* and s is chosen by \mathcal{A} . Recall that $D_2^s = (\mathbf{c}_0^*)^{d_2}$, where $d_2 = \log_g D_2$. In that case, for the values $(f_1^*, f_2^* (= T \neq h^{\alpha^*}))$ registered for i^* , the statement for **NIZK₃** becomes false, because $\mathbf{c}_2^* (f_2^*)^{-1} = h^{\alpha^*} D_2^s \cdot T^{-1} \neq D_2^s = (\mathbf{c}_0^*)^{d_2}$ holds. That is, we see that $\mathbf{c}_2^* (f_2^*)^{-1} \neq (\mathbf{c}_0^*)^{d_2}$ and $D_2 = g^{d_2}$ with respect to the witness d_2 . We notice that even though \mathcal{A} knows the witness d_2 , the simulation-soundness property of **NIZK₃** means that it is statistically infeasible for \mathcal{A} to generate a valid proof π_3 for a false statement.
- (2) Case when π_3^* is valid: Because of the simulation-soundness of **NIZK₃**, the validity shows that $\mathbf{c}_1^* (f_1^*)^{-1} = (\mathbf{c}_0^*)^{d_1}, D_1 = g^{d_1}, \mathbf{c}_2^* (f_2^*)^{-1} = (\mathbf{c}_0^*)^{d_2}$, and $D_2 = g^{d_2}$ should hold for the opener's secret d_1 and d_2 . Especially, these relations lead to the fact that \mathbf{c}_1^* and \mathbf{c}_2^* are constructed by $\mathbf{c}_1^* = f_1^* (\mathbf{c}_0^*)^{d_1}$ and $\mathbf{c}_2^* = f_2^* (\mathbf{c}_0^*)^{d_2}$. In that case, for the values $(f_1^*, f_2^* (= T \neq h^{\alpha^*}))$ registered for i^* , the statement for **sNIZK₂** becomes false, since $\mathbf{c}_1^* = g^{\alpha^*} (\mathbf{c}_0^*)^{d_1}$ and $\mathbf{c}_2^* = T (\mathbf{c}_0^*)^{d_2} \neq h^{\alpha^*} (\mathbf{c}_0^*)^{d_2}$ with respect to the witness α^* . \square

Overall, if \mathcal{A} is an adversary succeeding against the non-frameability of our construction with probability $\varepsilon (= \text{Adv}_{\text{DS}, \mathcal{A}}^{\text{Nf}})$, then we have proven the following bound:

$$\varepsilon \leq \text{Adv}_{\text{NIZK}_1, \mathcal{A}}^{\text{ZK}} + \text{Adv}_{\text{sNIZK}_2, \mathcal{A}}^{\text{ZK}} + \text{Adv}_{\text{sNIZK}_2, \mathcal{A}}^{\text{SS}} + \text{Adv}_{\text{NIZK}_3, \mathcal{A}}^{\text{SS}} + q \text{Adv}_{\text{DS}, \mathcal{A}}^{\text{EUF-CMA}} + q \text{Adv}_{\mathbb{G}_1, \mathcal{A}}^{\text{XDH}},$$

where $\text{Adv}_{(\text{s})\text{NIZK}, \mathcal{A}}^{\text{ZK}}$ (resp. $\text{Adv}_{(\text{s})\text{NIZK}, \mathcal{A}}^{\text{SS}}$) is the advantage of \mathcal{A} against the zero-knowledge property (resp. the simulation soundness) of **(s)NIZK**, $\text{Adv}_{\text{DS}, \mathcal{A}}^{\text{EUF-CMA}}$ is the advantage of \mathcal{A} against the unforgeability of the signature scheme DS and finally $\text{Adv}_{\mathbb{G}_1, \mathcal{A}}^{\text{XDH}}$ is the advantage of \mathcal{A} against the $\text{XDH}_{\mathbb{G}_1}$ assumption. \square

Table 1: Hybrid games for anonymity

Game	SndToU(i_0^*)	Challenge Signature σ^*	π_2 Statement by \mathcal{A}	Challenge Identities	Opening Trapdoor	Note
	SndToU(i_1^*)					
\mathbf{G}_0	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\alpha_0^*}, \pi_{1,real}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \pi_{2,real}^*$	True or False	Any/any	ok	Original Game $\mathbf{G}_{\mathcal{DGS}, \mathcal{A}}^{Anon-0}$
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\alpha_1^*}, \pi_{1,real}'$					
\mathbf{G}_1	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\alpha_0^*}, \pi_{1,sim}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \pi_{2,sim}^*$	True or False	Any/any	ok	Zero-knowledge
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\alpha_1^*}, \pi_{1,sim}'$					
\mathbf{G}_2	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\alpha_0^*}, \pi_{1,sim}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \pi_{2,sim}^*$	True	Any/any	ok	Simulation-soundness
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\alpha_1^*}, \pi_{1,sim}'$					
\mathbf{G}_3	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\alpha_0^*}, \pi_{1,sim}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \pi_{2,sim}^*$	True	i_0^* / i_1^*	ok	Statistical argument
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\alpha_1^*}, \pi_{1,sim}'$					
\mathbf{G}_4	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\alpha_0^*}, \pi_{1,sim}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \pi_{2,sim}^*$	True	i_0^* / i_1^*	ok, \mathcal{L}_σ	Non-frameability
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\alpha_1^*}, \pi_{1,sim}'$					
\mathbf{G}_5	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\alpha_0^*}, \pi_{1,sim}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, R_2, \pi_{2,sim}^*$	True	i_0^* / i_1^*	ok, \mathcal{L}_σ	XDH $_{\mathbf{G}_1}$ Assumption
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\alpha_1^*}, \pi_{1,sim}'$					
\mathbf{G}_6	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\alpha_0^*}, \pi_{1,sim}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, R_1, R_2, \pi_{2,sim}^*$	True	i_0^* / i_1^*	ok, \mathcal{L}_σ	XDH $_{\mathbf{G}_1}$ Assumption
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\alpha_1^*}, \pi_{1,sim}'$					
\mathbf{G}_7	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\beta^*}, \pi_{1,sim}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, R_1, R_2, \pi_{2,sim}^*$	True	i_0^* / i_1^*	ok, \mathcal{L}_σ	XDH $_{\mathbf{G}_1}$ Assumption
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\alpha_1^*}, \pi_{1,sim}'$					
\mathbf{G}_8	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\beta^*}, \pi_{1,sim}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, R_1, R_2, \pi_{2,sim}^*$	True	i_0^* / i_1^*	ok, \mathcal{L}_σ	XDH $_{\mathbf{G}_1}$ Assumption
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\gamma^*}, \pi_{1,sim}'$					
\mathbf{G}_F	$g^{\alpha_0^*}, h^{\alpha_0^*}, u_{i_0^*}, (u_{i_0^*})^{\gamma^*}, \pi_{1,sim}^*$	$\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, R_1, R_2, \pi_{2,sim}^*$	True	i_0^* / i_1^*	ok, \mathcal{L}_σ	XDH $_{\mathbf{G}_1}$ Assumption
	$g^{\alpha_1^*}, h^{\alpha_1^*}, u_{i_1^*}, (u_{i_1^*})^{\gamma^*}, \pi_{1,sim}'$					

5.2 Anonymity

Theorem 5.2. *Let $H : \{0, 1\}^* \rightarrow \mathbb{G}_1$ be modeled as a random oracle. Our DGS is anonymous under the zero-knowledge of all NIZKs, the simulation soundness of $sNIZK_2$, DGS non-frameability, and the $XDH_{\mathbb{G}_1}$ assumption.*

Proof. As shown in Table 1, we begin with \mathbf{G}_0 that is the original game of $\mathbf{G}_{DGS, \mathcal{A}}^{Anon-0}$. At the end, we arrive at \mathbf{G}_F where the challenged σ^* is valid for both i_0^* and i_1^* . Thus, from the adversary standpoint, the signature σ^* becomes perfectly indistinguishable. Note that it is required to create a sequence of the next hybrid games in order to change \mathbf{G}_F into the original game of $\mathbf{G}_{DGS, \mathcal{A}}^{Anon-1}$. However, we omit the details because it is easy to consider it by reversing the first hybrid games.

- \mathbf{G}_0 : This is the original game $\mathbf{G}_{DGS, \mathcal{A}}^{Anon-0}$ with respect to the target identity i_0^* , where the challenge signature $\sigma^* = (\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \pi_2^*)$ is generated under the group signing key of i_0^* .
- \mathbf{G}_1 : This is the same as \mathbf{G}_0 , except that \mathcal{C} simulates all proofs of the NIZKs without witnesses. By the (statistical) zero-knowledge property of all NIZKs, \mathbf{G}_0 is indistinguishable from \mathbf{G}_1 .
- \mathbf{G}_2 : This is the same as \mathbf{G}_1 , except that \mathcal{C} aborts if \mathcal{A} produces a proof of $sNIZK_2$ on a false statement. By the (statistical) simulation soundness property of the $sNIZK_2$, \mathbf{G}_1 is indistinguishable from \mathbf{G}_2 .
- \mathbf{G}_3 : This is the same as \mathbf{G}_2 , except that \mathcal{C} guesses target identities i_0^* and i_1^* that will be queried to the CH_b oracle. Let q be the number of the SndToU queries. \mathcal{C} picks $k_0, k_1 \leftarrow_{\mathcal{S}} \{1, \dots, q\}$ and then considers the k_0 -th (and k_1 -th) appeared identity in SndToU queries to be i_0^* (and i_1^* , respectively). If the guess of \mathcal{C} is wrong, then \mathcal{C} aborts but this reduces the overall advantage by a factor, at worse, $1/q^2$.
- \mathbf{G}_4 : This is the same as \mathbf{G}_3 , except that \mathcal{A} is not allowed to create a new valid signature for i_0^* and i_1^* without going through the Sig oracle. This means that \mathcal{C} is able to answer opening oracle queries with respect to i_0^* and i_1^* using only the list \mathcal{L}_σ : if $(i_b^*, m, \sigma) \in \mathcal{L}_\sigma$, \mathcal{C} returns i_b^* . By the DGS non-frameability (Theorem 5.1), \mathbf{G}_3 is indistinguishable from \mathbf{G}_4 .
- \mathbf{G}_5 : This is the same as \mathbf{G}_4 , except that the element \mathbf{c}_2^* of $\sigma^* = (\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \pi_2^*)$ is generated as a random element of \mathbb{G}_1 .

Claim 1. Under the $XDH_{\mathbb{G}_1}$ assumption, \mathbf{G}_4 is indistinguishable from \mathbf{G}_5 .

Proof. \mathcal{C} is given an $XDH_{\mathbb{G}_1}$ instance $(g, A = g^a, B = g^b, T)$. The group public key is set as $gpk = (h = H(g), D_1 = g^{d_1}, D_2 = A, \hat{X}, \hat{Y})$ where $x, y, d_1 \leftarrow_{\mathcal{S}} \mathbb{Z}_p$, $d_2 = a = \log_g A$, and \hat{X}, \hat{Y} are output by \mathcal{A} so that $ok = (d_1, \perp)$. \mathcal{C} then sends opk to \mathcal{A} , and all oracles are handled as follows:

- **Hash- H .** If a new n is queried as input, \mathcal{C} chooses $\delta_n \in \mathbb{Z}_p$ and computes $\mu = g^{\delta_n}$. \mathcal{C} returns μ and adds (n, δ, μ) to \mathcal{L}_H . If n has queried before, \mathcal{C} returns the μ stored in \mathcal{L}_H .
- **SndToU.** For any i (including i_0^* and i_1^*), \mathcal{C} obtains a key pair (usk_i, upk_i) from UKg and chooses a random $\alpha_i \leftarrow_{\mathcal{S}} \mathbb{Z}_p$. \mathcal{C} invokes the Join algorithm except that π_1 is simulated by using the zero-knowledge simulator \mathcal{S} . \mathcal{C} finally stores gsk_i . The list of honest users \mathcal{L}_h is updated by adding i .
- **USK.** As a queried i must not be i_0^* and i_1^* , \mathcal{C} can send (usk_i, gsk_i) for all queries. \mathcal{C} updates \mathcal{L}_{sk} by adding i .
- **WReg.** For a queried i and ρ , \mathcal{C} sets $reg_i = \rho$.
- **Sig.** For all $i \in \mathcal{L}_h$, \mathcal{C} uses α_i to generate σ on a message m by performing GSig , except when simulating π_2 . The list \mathcal{L}_σ is updated by adding (i, m, σ) .

- **Open.** To answer an opening query about $\sigma = (u, v, w, \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \pi_2)$, \mathcal{C} first uses \mathcal{L}_σ for i_b^* , where $b \in \{0, 1\}$, by checking whether or not $(i_b^*, m, \sigma) \in \mathcal{L}_\sigma$. If not, for other identities, \mathcal{C} uses only the d_1 , which it knows, by checking whether or not $f_1 = \mathbf{c}_1 \mathbf{c}_0^{d_1}$ is included in some $reg_i \in \mathbf{reg}$. Note that f_1 and f_2 are bounded to the secret exponent α_i by the simulation soundness of π_2 . Namely, f_1 and f_2 never indicate different users.

- **CH₀.** Given challenge identities i_0^*, i_1^* and m^* , \mathcal{C} uses $ok = (d_1, \perp)$ and $gsk_{i_0^*} = (\alpha_0^*, f_1^*, f_2^*, u_{i_0^*}, v_{i_0^*}, w_{i_0^*})$ to output the challenge signature

$$\sigma^* = (\tilde{u}^* = (u_{i_0^*}^*)^{r^*}, \tilde{v}^* = (v_{i_0^*}^*)^{r^*}, \tilde{w}^* = (w_{i_0^*}^*)^{r^*}, \mathbf{c}_0^* = B, \mathbf{c}_1^* = f_1^* \cdot B^{d_1}, \mathbf{c}_2^* = f_2^* \cdot T, \pi_2^*),$$

where $r^* \leftarrow_{\S} \mathbb{Z}_p$ and the random exponent for \mathbf{c}_0^* is implicitly set as $s^* = b = \log_g B$, and π_2^* is a simulated proof. The list \mathcal{L}_{ch} is updated by adding (i_0^*, m^*, σ^*) and (i_1^*, m^*, σ^*) .

If $T = g^{ab}$, \mathcal{C} simulates \mathbf{G}_4 ; otherwise, \mathcal{C} simulates \mathbf{G}_5 . Thus, the ability of \mathcal{A} to distinguish between \mathbf{G}_4 and \mathbf{G}_5 can be transformed into the ability of \mathcal{C} to solve the $\text{XDH}_{\mathbb{G}_1}$ problem. \square

• **G₆:** This is the same as \mathbf{G}_5 , except that the element \mathbf{c}_1^* of $\sigma^* = (\tilde{u}^*, \tilde{v}^*, \tilde{w}^*, \mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*, \pi_2^*)$ is generated as a random element of \mathbb{G}_1 .

Claim 2. Under the $\text{XDH}_{\mathbb{G}_1}$ assumption, \mathbf{G}_5 is indistinguishable from \mathbf{G}_6 .

Proof. The proof is almost analogous to the preceding proof for $\mathbf{G}_4 \approx \mathbf{G}_5$, except that d_1 is implicitly set as $a = \log_g A$ of an $\text{XDH}_{\mathbb{G}_1}$ instance $(g, A = g^a, B = g^b, T)$, and $d_2 \leftarrow_{\S} \mathbb{Z}_p$ is now chosen at random by \mathcal{C} . With this setting, when $\text{SndToU}(i)$ is called for any i , \mathcal{C} uses a key pair (usk_i, upk_i) generated by UKg and $\alpha_i \leftarrow_{\S} \mathbb{Z}_p$ to execute the Join algorithm, in which π_1 is simulated. For all i (including i_0^* and $i_1^*) \in \mathcal{L}_h$, \mathcal{C} is able to respond to all USK (unless either i_0^* or i_1^* is queried) and Sig queries by using α_i . \mathcal{C} now uses \mathcal{L}_σ and d_2 to deal with Open queries. This is also valid due to the simulation soundness of π_2 .

In CH_0 , \mathcal{C} having $gsk_{i_0^*} = (\alpha_0^*, f_1^*, f_2^*, u_{i_0^*}, v_{i_0^*}, w_{i_0^*})$ implicitly sets the random exponent as $s^* = \log_g B$ for the challenge signature

$$\sigma^* = (\tilde{u}^* = (u_{i_0^*}^*)^{r^*}, \tilde{v}^* = (v_{i_0^*}^*)^{r^*}, \tilde{w}^* = (w_{i_0^*}^*)^{r^*}, \mathbf{c}_0^* = B, \mathbf{c}_1^* = f_1^* \cdot T, \mathbf{c}_2^* = R, \pi_2^*),$$

where $r^* \leftarrow_{\S} \mathbb{Z}_p$, $R \leftarrow_{\S} \mathbb{G}_1$ and the proof π_2^* is simulated. If $T = g^{ab}$, \mathcal{C} simulates \mathbf{G}_5 ; otherwise, \mathcal{C} simulates \mathbf{G}_6 . Thus, the ability of \mathcal{A} to distinguish between \mathbf{G}_5 and \mathbf{G}_6 can be transformed into the ability of \mathcal{C} to solve the $\text{XDH}_{\mathbb{G}_1}$ problem. \square

• **G₇:** This is the same as \mathbf{G}_6 , except that, when \mathcal{C} generates $\tau_0^* = (f_1^*, f_2^*, u_{i_0^*}, w_{i_0^*}, \pi_1^*, \sigma_{DS}^*)$ to handle $\text{SndTol}(i_0^*)$, the value $w_{i_0^*}$ is computed by $w_{i_0^*} = (u_{i_0^*}^*)^{\beta^*}$ where β^* is distinct from $\alpha_0^* = \log_g f_1^* = \log_h f_2^*$.

Claim 3. Under the $\text{XDH}_{\mathbb{G}_1}$ assumption, \mathbf{G}_6 is indistinguishable from \mathbf{G}_7 .

Proof. Let $(g, A = g^a, B = g^b, T)$ be an $\text{XDH}_{\mathbb{G}_1}$ instance that \mathcal{C} is given. \mathcal{C} sets $h = g^{\delta_g}$ for some random scalar δ_g and the opener key pair as $ok = (d_1, d_2)$ and $opk = (D_1 = g^{d_1}, D_2 = g^{d_2})$, where $d_1, d_2 \leftarrow_{\S} \mathbb{Z}_p$. It receives the issuer's public key $ipk = (\hat{X}, \hat{Y})$ from \mathcal{A} . All oracles are managed as follows:

- **Hash- H .** If a new n is queried as input, \mathcal{C} chooses $\delta_n \in \mathbb{Z}_p$ and computes $\mu = g^{\delta_n}$. \mathcal{C} returns μ and adds (n, δ_n, μ) to \mathcal{L}_H . If n has queried before, \mathcal{C} returns μ stored in \mathcal{L}_H .

- **SndToU.** If $i \neq i_0^*$, \mathcal{C} invokes the Join algorithm by obtaining a key pair (usk_i, upk_i) from DKg and choosing its secret $\alpha \leftarrow_{\S} \mathbb{Z}_p^*$, except for simulating a proof π_1 . If $i = i_0^*$, \mathcal{C} simulates the proof π_1^* with respect to the following elements:

$$(f_1^* = A, f_2^* = A^{\delta_g}, u_{i_0^*} = B, w_{i_0^*} = T).$$

Here, $u_{i_0^*} = H(f_1^*)$ is set as B by programming the random oracle accordingly. This defines the unknown α_0^* as $a = \log_g A$. Note that if $T = g^{ab}$, $w_{i_0^*} = (u_{i_0^*})^{\alpha_0^*}$. Otherwise, if $T \neq g^{ab}$, then $\beta^* = \log_{u_{i_0^*}} w_{i_0^*}$ is set as a random value distinct from α_0^* . \mathcal{A} is given the above elements along with (π_1^*, σ_{DS}^*) , where σ_{DS}^* is a digital signature on the message $f_1^* || f_2^*$. In response to those elements, \mathcal{C} receives a relevant $v_{i_0^*}$ which satisfies $e(v_{i_0^*}, \hat{g}) = e(u_{i_0^*}, \hat{X})e(w_{i_0^*}, \hat{Y})$. \mathcal{C} finally stores $gsk = (\alpha, f_1, f_2, u, v, w)$ for $i \neq i_0^*$, and it stores $gsk_{i_0^*} = (\perp, f_1^*, f_2^*, u_{i_0^*}, v_{i_0^*}, w_{i_0^*})$ for $i = i_0^*$. The list of honest users \mathcal{L}_h is updated by adding i .

- **USK.** As a queried i must not be either i_0^* or i_1^* , \mathcal{C} can send (usk_i, gsk_i) for all queries. \mathcal{C} updates \mathcal{L}_{sk} by adding i .
- **WReg.** For a queried i and ρ , \mathcal{C} sets $reg_i = \rho$.
- **Sig.** For any i , \mathcal{C} can generate σ on a message m by performing **GSig** and simulating the proof π_2 therein. The list \mathcal{L}_σ is updated by adding (i, m, σ) .
- **Open.** \mathcal{C} uses either \mathcal{L}_σ or ok to answer an opening query.
- **CH₀.** Given challenge identities i_0^*, i_1^* and m^* , \mathcal{C} uses $gsk_{i_0^*} = (\perp, f_1^*, f_2^*, u_{i_0^*}, v_{i_0^*}, w_{i_0^*})$ to output the challenge signature

$$\sigma^* = (\tilde{u}^* = (u_{i_0^*}^*)^{r^*}, \tilde{v}^* = (v_{i_0^*}^*)^{r^*}, \tilde{w}^* = (w_{i_0^*}^*)^{r^*}, \mathbf{c}_0^* = R_0, \mathbf{c}_1^* = R_1, \mathbf{c}_2^* = R_2, \pi_2^*),$$

where $r^* \leftarrow_{\$} \mathbb{Z}_p$ and $R_0, R_1, R_2 \leftarrow_{\$} \mathbb{G}_1$, and π_2^* is a simulated proof. The list \mathcal{L}_{ch} is updated by adding (i_0^*, m^*, σ^*) and (i_1^*, m^*, σ^*) .

If $T = g^{ab}$, \mathcal{C} simulates **G₆**, and otherwise, \mathcal{C} simulates **G₇**. Thus, the ability of \mathcal{A} to distinguish between **G₆** and **G₇** can be transformed into the ability of \mathcal{C} to solve the $\text{XDH}_{\mathbb{G}_1}$ problem. \square

- **G₈:** This is the same as **G₇**, except that, when \mathcal{C} deals with **SndTol**(i_1^*), the value $w_{i_1^*}$ of $\tau_1^* = (f_1', f_2', u_{i_1^*}, w_{i_1^*}, \pi_1', \sigma_{DS}')^*$ is computed by $(u_{i_1^*}^*)^{\gamma^*}$ that is distinct from $\alpha_1^* = \log_g f_1' = \log_h f_2'$.

Claim 4. Under the $\text{XDH}_{\mathbb{G}_1}$ assumption, **G₇** is indistinguishable from **G₈**.

Proof. A simulation of \mathcal{C} is the same as that of **G₆** \approx **G₇**, except for the following case in **SndToU**(i): For $i = i_0^*$, \mathcal{C} chooses $\alpha_0^*, \beta^* \leftarrow_{\$} \mathbb{Z}_p$ and simulates π_1^* for elements $(f_1^* = g^{\alpha_0^*}, f_2^* = (f_1^*)^{\delta_g}, u_{i_0^*} = g^{\delta_{f_1^*}}, w_{i_0^*} = (u_{i_0^*})^{\beta^*})$, where δ_g and $\delta_{f_1^*}$ are random values associated with $H(g) = h = g^{\delta_g}$ and $H(f_1^*) = u_{i_0^*} = g^{\delta_{f_1^*}}$, respectively. \mathcal{A} is then given $(f_1^*, f_2^*, u_{i_0^*}, w_{i_0^*}, \pi_1^*)$ and a digital signature σ_{DS}^* on $f_1^* || f_2^*$ together. For $i = i_1^*$, \mathcal{C} computes $(f_1' = A, f_2' = A^{\delta_g}, u_{i_1^*} = B, w_{i_1^*} = T)$ by using an $\text{XDH}_{\mathbb{G}_1}$ instance $(g, A = g^a, B = g^b, T)$ under the implicit setting of $\alpha_1^* = a = \log_g A$. As the response, the tuple $\tau_1^* = (f_1', f_2', u_{i_1^*}, w_{i_1^*}, \pi_1', \sigma_{DS}')^*$ is given to \mathcal{A} where π_1' is a simulated proof, and $\sigma_{DS}'^*$ is a digital signature on $f_1' || f_2'$. Under this simulation, if $T = g^{ab}$, \mathcal{C} simulates **G₇**, and otherwise, \mathcal{C} simulates **G₈**. Therefore, the ability of \mathcal{A} to distinguish between **G₇** and **G₈** can be transformed into the ability of \mathcal{C} to solve the $\text{XDH}_{\mathbb{G}_1}$ problem. \square

- **G_F:** This is the same as **G₈**, except that, in response to **SndTol**(i_0^*) and **SndTol**(i_1^*), \mathcal{C} sets the same exponent γ^* such that $\log_{u_{i_0^*}} w_{i_0^*} = \gamma^* = \log_{u_{i_1^*}} w_{i_1^*}$. The same exponent γ^* in **G_F** shows that the values $(\tilde{u}^*, \tilde{v}^*, \tilde{w}^*)$ of σ^* could be possibly computed by the group signing key for i_0^* and i_1^* with the same possibility.

Claim 5. Under the $\text{XDH}_{\mathbb{G}_1}$ assumption, **G₈** is indistinguishable from **G_F**.

Proof. Given an $\text{XDH}_{\mathbb{G}_1}$ instance $(g, A = g^a, B = g^b, T)$, \mathcal{C} first receives issuer's public key $ipk = (\hat{X}, \hat{Y})$ that is generated by \mathcal{A} . Then, \mathcal{C} generates an opener key pair (ok, opk) to provide \mathcal{A} with opk . All oracles are managed as follows:

- **Hash-H.** If a new n is queried as input, \mathcal{C} chooses $\delta_n \in \mathbb{Z}_p$ and computes $\mu = g^{\delta_n}$. \mathcal{C} returns μ and adds (n, δ_n, μ) to \mathcal{L}_H . If n has queried before, \mathcal{C} returns μ stored in \mathcal{L}_H .
- **SndToU.** For each query i , \mathcal{C} first obtains a key pair (usk_i, upk_i) by running **DKG**. To execute **Join**, \mathcal{C} works as follows. If $i = i_0^*$, \mathcal{C} generates a simulated proof π_1^* with respect to the elements:

$$(f_1^* = g^{\alpha_0^*}, f_2^* = (f_1^*)^{\delta_g}, u_{i_0^*} = B, w_{i_0^*} = T),$$

where $\alpha_0^* \leftarrow_{\mathcal{S}} \mathbb{Z}_p$ and $h = g^{\delta_g}$ and $B = u_{i_0^*} = g^{\delta_{f_1^*}} = H(f_1^*)$. Note that $\delta_{f_1^*}$ and β^* are implicitly set as unknown exponents $\delta_{f_1^*} = b = \log_g B$ and $\beta^* = \log_B T$, respectively. If $i = i_1^*$, \mathcal{C} sets γ^* as $a = \log_g A$, to which the proof π_1' is generated with respect to the values:

$$(f_1' = g^{\alpha_1^*}, f_2' = (f_1')^{\delta_g}, u_{i_1^*} = g^{\delta_{f_1'}}, w_{i_1^*} = A^{\delta_{f_1'}}),$$

where $\alpha_1^* \leftarrow_{\mathcal{S}} \mathbb{Z}_p$ and $\delta_{f_1'} \leftarrow_{\mathcal{S}} \mathbb{Z}_p$ and $u_{i_1^*} = g^{\delta_{f_1'}} = H(f_1')$. We notice that if $T = g^{ab}$, β^* is equal to γ^* and otherwise, if $T \neq g^{ab}$, β^* is distinct from γ^* . In both cases, \mathcal{C} receives $v_{i_0^*}$ and $v_{i_1^*}$ from \mathcal{A} , respectively.

If $i \neq i_0^*$ or $i \neq i_1^*$, such a tuple is computed by $\alpha_i \leftarrow_{\mathcal{S}} \mathbb{Z}_p$ as it is in the real scheme. The list \mathcal{L}_h is updated by adding i .

- **USK.** Since $i \notin \{i_0^*, i_1^*\}$, \mathcal{C} is able to give (usk_i, gsk_i) to \mathcal{A} . The list \mathcal{L}_{sk} is updated by adding i .
- **WReg.** For all i and ρ , \mathcal{C} sets $reg_i = \rho$.
- **Sig.** For all i , \mathcal{C} can respond to \mathcal{A} by invoking **GSig**, except for simulating π_2 . The list \mathcal{L}_σ is updated by adding (i, m, σ) .
- **Open.** \mathcal{C} uses either \mathcal{L}_σ or ok to answer an opening query.
- **CH₀.** For challenge identities i_0^*, i_1^* and m^* , \mathcal{C} generates the challenge signature

$$\sigma^* = (\tilde{u}^* = (u_{i_0^*})^{r^*}, \tilde{v}^* = (v_{i_0^*})^{r^*}, \tilde{w}^* = (w_{i_0^*})^{r^*}, \mathbf{c}_0^* = R_0, \mathbf{c}_1^* = R_1, \mathbf{c}_2^* = R_2, \pi_2^*),$$

where $r^* \leftarrow_{\mathcal{S}} \mathbb{Z}_p$ and $R_0, R_1, R_2 \leftarrow_{\mathcal{S}} \mathbb{G}_1$, and π_2^* is a simulated proof. The list \mathcal{L}_{ch} is updated by adding (i_0^*, m^*, σ^*) and (i_1^*, m^*, σ^*) .

If $T \neq g^{ab}$, we see that $\beta^* \neq \gamma^*$, where \mathcal{C} simulates \mathbf{G}_8 . Otherwise, if $T = g^{ab}$, we see that $\beta^* = \gamma^*$ and \mathcal{C} simulates \mathbf{G}_F . The ability of \mathcal{A} to distinguish between \mathbf{G}_8 and \mathbf{G}_F can then be transformed into the ability of \mathcal{C} to solve the $\text{XDH}_{\mathbb{G}_1}$ problem. \square

To conclude the proof, we note that, in \mathbf{G}_F , the certificate $(\tilde{u}^*, \tilde{v}^*, \tilde{w}^*)$ contained in σ^* is valid for both users i_0^* and i_1^* . The other elements of σ^* are the random (since \mathbf{G}_6) tuple $(\mathbf{c}_0^*, \mathbf{c}_1^*, \mathbf{c}_2^*) \in \mathbb{G}_1^3$ and the NIZK proof π_2^* that do not carry any information about i_0^* or i_1^* . The advantage of an adversary in this game is then, at best, negligible. Overall, if \mathcal{A} is an adversary succeeding against the anonymity of our construction with probability $\varepsilon (= \text{Adv}_{DGS, \mathcal{A}}^{\text{Anon}})$, then we have proven the following bound:

$$\varepsilon \leq 2(\text{Adv}_{\text{NIZK}_{1, \mathcal{A}}}^{\text{ZK}} + \text{Adv}_{\text{sNIZK}_{2, \mathcal{A}}}^{\text{ZK}} + \text{Adv}_{\text{NIZK}_{3, \mathcal{A}}}^{\text{ZK}} + \text{Adv}_{\text{sNIZK}_{2, \mathcal{A}}}^{\text{SS}} + q^2 \text{Adv}_{DGS, \mathcal{A}}^{\text{Nf}} + 5q^2 \text{Adv}_{\mathbb{G}_1, \mathcal{A}}^{\text{XDH}}),$$

where $\text{Adv}_{(\text{s})\text{NIZK}, \mathcal{A}}^{\text{ZK}}$ (resp. $\text{Adv}_{(\text{s})\text{NIZK}, \mathcal{A}}^{\text{SS}}$) is the advantage of \mathcal{A} against the zero-knowledge property (resp. the simulation soundness) of $(\text{s})\text{NIZK}$, $\text{Adv}_{DGS, \mathcal{A}}^{\text{Nf}}$ is the advantage of \mathcal{A} against the non-frameability of DGS and finally $\text{Adv}_{\mathbb{G}_1, \mathcal{A}}^{\text{XDH}}$ is the advantage of \mathcal{A} against the $\text{XDH}_{\mathbb{G}_1}$ assumption. \square

5.3 Traceability

Theorem 5.3. *Let $H : \{0,1\}^* \rightarrow \mathbb{G}_1$ be modeled as a random oracle. Our \mathcal{DGS} is traceable under the simulation soundness of both NIZK_1 and sNIZK_2 and the GPS_2 assumption.*

Proof. In the following hybrid argument, we aim to show that the probability that \mathcal{A} breaks the traceability becomes zero in the last game \mathbf{G}_F .

- \mathbf{G}_0 : This is the original traceability game $\mathbf{G}_{\mathcal{DGS}, \mathcal{A}}^{\text{Trace}}$.
- \mathbf{G}_1 : This is the same game as \mathbf{G}_0 , except that \mathcal{C} aborts if a proof of a false statement is generated for NIZK_1 or sNIZK_2 . Because of the (statistical) simulation soundness of these NIZKs, \mathbf{G}_0 is indistinguishable from \mathbf{G}_1 .
- \mathbf{G}_2 : This is the same game as \mathbf{G}_1 , except that \mathcal{C} aborts if the proof Π generated by GOpen for some identity $i (\neq \perp)$ is rejected by GJudge .

Claim 1: \mathbf{G}_1 is identical to \mathbf{G}_2 .

As we assume that GOpen does not return \perp , it necessarily returns $\Pi = (f_1, f_2, \sigma_{DS}, \pi_3)$ accusing i (we recall that GOpen is always invoked honestly in the traceability game). As NIZK_3 has perfect completeness, the first condition tested by GJudge holds. Moreover, an invalid signature σ_{DS} on $f_1 || f_2$ would have been rejected by \mathcal{C} during the corresponding AddU or SndTol queries. Therefore, Π also passes the second test of GJudge , which concludes our proof.

- \mathbf{G}_F : This is the same game as \mathbf{G}_2 , except that GOpen never outputs \perp when run on a valid group signature. In \mathbf{G}_F , the probability that \mathcal{A} breaks traceability is zero.

Claim 2: Under the GPS_2 assumption, \mathbf{G}_2 is indistinguishable from \mathbf{G}_F .

Proof. We show that \mathcal{C} can solve the GPS_2 problem using \mathcal{A} . For generators $g \in \mathbb{G}_1$ and $\hat{g} \in \mathbb{G}_2$, \mathcal{C} is given an instance of the GPS_2 problem as the tuple (\hat{X}, \hat{Y}) equipped with the $\mathcal{O}_1^{\text{GPS}_2}$ and $\mathcal{O}_2^{\text{GPS}_2}$ oracles as described in Section 3. \mathcal{C} first sets $gpk = (h = H(g), D_1 = g^{d_1}, D_2 = g^{d_2}, \hat{X}, \hat{Y})$ and $ok = (d_1, d_2)$, where $d_1, d_2 \leftarrow_{\$} \mathbb{Z}_p$. Then, the issuer's secret key is implicitly set as $ik = (x, y)$ such that $\hat{X} = \hat{g}^x$ and $\hat{Y} = \hat{g}^y$. \mathcal{C} sends gpk and ok to \mathcal{A} . All oracle queries are managed as follows:

- **Hash- H .** For an input n where $n \in \mathbb{G}_1$, if n is new, \mathcal{C} obtains $\mu \in \mathbb{G}_1$ by calling $\mathcal{O}_1^{\text{GPS}_2}$. Then, μ is returned as output, and (n, μ) is added to \mathcal{L}_H . If n was queried before, \mathcal{C} returns μ stored in \mathcal{L}_H .
- **AddU.** For a queried i , \mathcal{C} initializes the honest user i by generating (usk_i, upk_i) . To generate gsk_i without knowing ik , \mathcal{C} computes $f_1 = g^\alpha$, $f_2 = H(g)^\alpha$, $u = H(f_1)$ and $w = u^\alpha$, where $\alpha \leftarrow_{\$} \mathbb{Z}_p$. In addition, \mathcal{C} simulates a proof π_1 with respect to α . \mathcal{C} sends (f_1, u, w) to $\mathcal{O}_2^{\text{GPS}_2}$ and obtains $v = u^x w^y$. \mathcal{C} then sets $gsk = (\alpha, f_1, f_2, u, v, w)$ and $reg_i = (i, \tau)$, where $\tau = (f_1, f_2, u, w, \pi_1, \sigma_{DS})$. In the end, \mathcal{C} sends upk_i to \mathcal{A} . The list \mathcal{L}_h is updated by adding i .
- **CrptU.** For a queried (i, upk) , \mathcal{C} initializes the dishonest user i by setting the user public key of i as upk . The list \mathcal{L}_c is updated by adding (i, cont) .
- **SndTol.** For a queried i such that $(i, \text{cont}) \in \mathcal{L}_c$, \mathcal{C} executes the GJoin protocol without ik . Given a tuple $(f_1, f_2, u, w, \pi_1, \sigma_{DS})$ from \mathcal{A} , \mathcal{C} checks the freshness of f_1 and also checks that π_1 and σ_{DS} are both valid. If so, $\alpha = \log_g f_1 = \log_u w$ holds for some unknown α , because of the simulation soundness of π_1 . With this fact, \mathcal{C} sends the tuple (f_1, u, w) to $\mathcal{O}_2^{\text{GPS}_2}$ and obtains $v = u^x w^y$. Note that u is obtained from $\mathcal{O}_1^{\text{GPS}_2}$ on input f_1 . \mathcal{C} finally sends v to \mathcal{A} .

- **USK.** For a user i as input, \mathcal{C} outputs gsk_i and usk_i . The list \mathcal{L}_{sk} is updated by adding i .

- **RReg.** For a user i as input, \mathcal{C} outputs reg_i .

\mathcal{A} finally outputs a forgery $\sigma^* = (u^*, v^*, w^*, c_0^*, c_1^*, c_2^*, \pi_2^*)$ on m^* . Here, the simulation soundness of π_2^* leads to the equality of discrete logarithms such that $w^* = (u^*)^{\alpha^*}$, $f_1^* = g^{\alpha^*}$ and $f_2^* = h^{\alpha^*}$ for some unknown α^* , where f_1^* and f_2^* are obtained by $c_1^*(c_0^*)^{d_1^{-1}}$ and $c_2^*(c_0^*)^{d_2^{-1}}$, respectively. If \mathcal{A} wins, σ^* is valid (i.e., $e(u^*, \hat{X})e(w^*, \hat{Y}) = e(v^*, \hat{g})$) but the opening of σ^* fails, which is the only distinction between \mathbf{G}_2 and \mathbf{G}_F . The opening failure indicates that, for all (f_1, f_2) from $reg_i = (i, f_1, f_2, u, w, \pi_1, \sigma_{DS}) \in \mathbf{reg}$, (f_1^*, f_2^*) is different from (f_1, f_2) , inducing that α^* is new. This implies that both $f_1^* \neq f_1$ and $f_2^* \neq f_2$ so that f_1^* is not queried to $\mathcal{O}_1^{GPS_2}$. Based on this fact, \mathcal{C} outputs (f_1^*, u^*, v^*, w^*) as a solution for the GPS_2 problem. \square

Overall, if \mathcal{A} is an adversary succeeding against the traceability of our construction with probability $\varepsilon (= \mathbf{Adv}_{DGS, \mathcal{A}}^{Trace})$, then we have proven the following bound:

$$\varepsilon \leq \mathbf{Adv}_{\text{NIZK}_1, \mathcal{A}}^{SS} + \mathbf{Adv}_{\text{NIZK}_2, \mathcal{A}}^{SS} + \mathbf{Adv}_{\mathcal{BG}, \mathcal{A}}^{GPS_2},$$

where $\mathbf{Adv}_{(s)\text{NIZK}, \mathcal{A}}^{SS}$ is the advantage of \mathcal{A} against the simulation soundness of (s)NIZK, and $\mathbf{Adv}_{\mathcal{BG}, \mathcal{A}}^{GPS_2}$ is the advantage of \mathcal{A} against the GPS_2 assumption. \square

6 Comparison

Tables 2 and 3 provide a comparison between previous DGS schemes and ours in terms of security and efficiency, respectively. In the comparison, we consider several well-known practical DGS schemes proven in the random oracle model (ROM), including DP06 by Delerablée and Pointcheval [25], BCN+10 by Bichsel et al. [11], PS16 by Pointcheval and Sanders [45], LMP+16 by Libert et al. [41], DS18 by Derler and Slamanig [26], and KLA+20 by Kim et al. [40]. We additionally consider the very recent CS20 and CS20* by Clarisse and Sanders [23], which can be proven secure without ROM.

Table 2: Security comparison between DGS schemes and our schemes

Scheme	Rew. Ext.	CRS	ROM	Opening	Security Model (Anonymity)	Assumption for Traceability
DP06	$\mathcal{O}(1)$	Yes	Yes	Sound	BSZ (CCA2)	$(q\text{-DL} \Rightarrow_{alg})$ $q\text{-SDH}$
BCN+10	$\mathcal{O}(q)$	No	Yes	Sound	BMW* (CCA-)	$(\text{SDL} \Rightarrow_{alg})$ LRSW
LMP+16	$\mathcal{O}(q)$	No	Yes	Sound	KY (CCA2)	SXDH
PS16	$\mathcal{O}(q)$	No	Yes	Sound	BMW* (CCA-)	$(\text{SDL} \Rightarrow_{alg})$ PS
DS18	$\mathcal{O}(1)$	Yes	Yes	Weak Sound	BSZ (CCA2)	GGM
KLA+20	$\mathcal{O}(1)$	No	Yes	Sound	BSZ (CCA-)	$(\text{SDL} \Rightarrow_{alg})$ GPS_1
CS20	No	Yes	No	Weak Sound	BSZ (CCA-)	GGM
CS20*	No	Yes	No	Sound	BMW* (CCA2)	GGM
Ours	No	No	Yes	Sound	BSZ* (CCA-)	$(\text{SDL} \Rightarrow_{alg})$ GPS_2

◦ Rew. Ext. stands for Rewinding Extraction.

◦ q is the number of queries to the JOIN oracle in the traceability proof.

◦ \Rightarrow_{alg} represents the algebraic reduction via the AGM.

As shown in Table 2, we emphasize that the security of CS20, CS20*, and our schemes can be proven without any rewinding extraction, meaning that these are free from the rewinding problems. In contrast, the other DGS schemes require at least one or more rewinding extractions in either their non-frameability

or traceability security analyses. All their non-frameability proofs are justified under the hardness of some computational problems such as the (S)DL [11, 25, 41, 45] or the co-Diffie-Hellman Inversion [26], and like the security analysis of the Schnorr signature, the witness used as the user secret key must be extracted as the solution of their underlying computational hardness problem. Furthermore, regarding the traceability proofs, BCN+10, LMP+16, and PS16 require sequential $O(q)$ rewinding extractions for dealing with $O(q)$ joining queries that are issued by an adversary. However, in case of DP06, DS18, and KLA+20, such sequential $O(q)$ extractions are ruled out by respectively employing a straight-line extractable commitment [25], a structure-preserving signature on equivalence classes [26], and a new assumption that is redefined as the GPS_1 assumption in this paper.

Regarding a common reference string (CRS), DP06, DS18, CS20 and CS20* need a CRS that has an embedded trapdoor during their security analyses. More concretely, a CRS in DP06 is necessary to remove sequential $O(q)$ rewinding extractions resulting from joining queries, thereby forcing a prospective user to use an extractable commitment based on the CRS. However, in the case of a subverted CRS, the entity possessing the trapdoor can possibly extract the witness as the user secret key. A CRS in DS18 is used to provide full anonymity (CCA2), by forcing a signer to generate an El Gamal encryption based on the CRS. This means that in the subversion of a CRS, the entity with the trapdoor (i.e., the El Gamal decryption key) can extract user-specific opening information from the El Gamal ciphertext and identify the signer without the opener’s secret key. The CRS in CS20 and CS20* is essential for proving non-frameability, by forcing a signer to use two CRS elements ($X = g^x, \tilde{X} = \tilde{g}^x$) based on the CRS. In case of a subverted CRS, however, the entity knowing the discrete logarithm x can forge a signature on an arbitrary message, given a valid signature generated by a target user. As a result, those CRS-based \mathcal{DGS} schemes work securely only in contexts where CRS subversion never happens.

We recall that opening soundness, introduced in [47], is the property satisfied by schemes whose group signatures cannot be opened to two distinct users. The soundness is *weak* if this property only holds for honest users. Essentially, a scheme is opening sound if a user secret key cannot be shared by different users.⁵ For most schemes, this can be achieved by checking the freshness of the user’s key at the joining time. The case of DS18 and CS20 is more complex as certificates are issued on equivalence classes, which makes key sharing more difficult to detect. Comparing \mathcal{DGS} security models, DP06, LMP+16, DS18, KLA+20, and CS20* are all based on either the BSZ model [8] or the KY model [39] that considers a dynamic group and separable group managers. In particular, our scheme is proven to be secure in the security model, denoted by ‘BSZ*’ [13], which captures the stronger security than the BSZ by allowing an adversary to generate group managers’ keys maliciously. BCN+10, PS16, and CS20 are proven in the weaker model as denoted by BMW* and placed between [6] and [8], where dynamic joining is allowed but the group managers are not separated. Regarding anonymity, DP06, LMP+16, DS18 and CS20* provide fully anonymity (CCA2), meaning that a target signature is anonymous even though the signing key of the target user is given to the adversary. In DCN+10, PS16, KLA+20, and our scheme, leaking such a signing key leads to breaking anonymity so that a slightly weaker notion (CCA-) called selfless anonymity is justified by disallowing USK (i.e., user secret key) queries for the target user. CS20 also provides selfless anonymity. For traceability proofs, only LMP+16 relies on the standard SXDH assumption. BCN+10, PS16, KLA+20, and our scheme are all based on the interactive LRSW, PS, GPS_1 , and GPS_2 assumptions, respectively, but notably the LRSW, PS, GPS_1 and GPS_2 (Section 3.2) assumptions in the AGM are all proven to be at least as hard as the standard SDL assumption [4]. DP06 is based on the q -SDH assumption, which is still implied by the non-standard q -DL assumption even in the AGM [4]. Also, DS18, CS20, and CS20* rely on a signature scheme whose security is proven directly in the generic group model (GGM).

⁵This notion is slightly different from that of [47].

Table 3: Efficiency comparison between \mathcal{DGS} schemes and our schemes

Scheme	Signature			Sign				Verify				Opening
	\mathbb{G}_2	\mathbb{G}_1	\mathbb{Z}_p	P	E_T	E_2	E_1	P	E_T	E_2	E_1	
DP06		4	5		4		7	4		4	11	$O(1)$
BCN+10		3	2		1		3	4			3	$O(N)$
PS16		2	2		1		2	3			3	$O(N)$
LMP+16		7	3	4		2	13	4		4	11	$O(1)$
DS18	4	3	3			5	6	5		2	4	$O(N)$
KLA+20		3	2				4	3		2		$O(N)$
CS20	1	4				1	5	5		1		$O(N)$
CS20*	1	4				1	5	5		1		$O(N)$
Ours		6	3				12	3			10	$O(1)$

◦ For $i \in \{1, 2, T\}$, E_i indicates an exponentiation in \mathbb{G}_i .

◦ P is a pairing operation. ◦ N is the number of users in \mathcal{DGS} .

Table 3 illustrates that our \mathcal{DGS} scheme has comparable efficiency. We emphasize that, except for CS20, CS20* and our schemes, the efficiency of all other schemes should be considered by taking into account the fact that they use knowledge extractors, which eventually causes security degradation. Regarding computational cost, our **GSig** algorithm only requires 12 exponentiations in \mathbb{G}_1 , and our **GVf** algorithm requires 3 pairing operations plus 10 exponentiations in \mathbb{G}_1 . In particular, contrarily to CS20 and CS20*, we manage to keep all signer’s computations in \mathbb{G}_1 , which avoid implementing the complex arithmetic of \mathbb{G}_2 . Similarly, one must take into account that the elements of \mathbb{G}_2 are at least twice as large as those in \mathbb{G}_1 for usual type-3 curves like BN [3]. Finally, we achieve constant time (and fast) opening contrarily to schemes that exclusively use rerandomizable signatures.

References

- [1] Michel Abdalla, Pierre-Alain Fouque, Vadim Lyubashevsky, and Mehdi Tibouchi. Tightly-secure signatures from lossy identification schemes. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *Lecture Notes in Computer Science*, pages 572–590. Springer, 2012.
- [2] Masayuki Abe, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Optimal structure-preserving signatures in asymmetric bilinear groups. In *CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 649–666. Springer, 2011.
- [3] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford E. Tavares, editors, *SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer, 2005.
- [4] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *Lecture Notes in Computer Science*, pages 121–151. Springer, 2020.
- [5] Mihir Bellare, Georg Fuchsbauer, and Alessandra Scafuro. Nizks with an untrusted CRS: security in the face of parameter subversion. In Jung Hee Cheon and Tsuyoshi Takagi, editors, *ASIACRYPT 2016, Part II*, volume 10032 of *Lecture Notes in Computer Science*, pages 777–804, 2016.

- [6] Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.
- [7] Mihir Bellare and Adriana Palacio. The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 273–289. Springer, 2004.
- [8] Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005.
- [9] Eli Ben-Sasson, Alessandro Chiesa, Matthew Green, Eran Tromer, and Madars Virza. Secure sampling of public parameters for succinct zero knowledge proofs. In *SP 2015*, pages 287–304. IEEE Computer Society, 2015.
- [10] David Bernhard, Marc Fischlin, and Bogdan Warinschi. Adaptive proofs of knowledge in the random oracle model. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *Lecture Notes in Computer Science*, pages 629–649. Springer, 2015.
- [11] Patrik Bichsel, Jan Camenisch, Gregory Neven, Nigel P. Smart, and Bogdan Warinschi. Get shorty via group signatures without encryption. In Juan A. Garay and Roberto De Prisco, editors, *SCN 2010*, volume 6280 of *Lecture Notes in Computer Science*, pages 381–398. Springer, 2010.
- [12] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
- [13] Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. volume 33, pages 1822–1870, 2020.
- [14] Xavier Boyen and Brent Waters. Full-domain subgroup hiding and constant-size group signatures. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.
- [15] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick D. McDaniel, editors, *CCS 2004*, pages 132–145. ACM, 2004.
- [16] Ernie Brickell and Jiangtao Li. Enhanced privacy ID: A direct anonymous attestation scheme with enhanced revocation capabilities. In Peng Ning and Ting Yu, editors, *WPES 2007*, pages 21–30. ACM, 2007.
- [17] Jan Camenisch, Liqun Chen, Manu Drijvers, Anja Lehmann, David Novick, and Rainer Urian. One TPM to bind them all: Fixing TPM 2.0 for provably secure anonymous attestation. In *SP 2017*, pages 901–920. IEEE Computer Society, 2017.
- [18] Jan Camenisch, Manu Drijvers, Anja Lehmann, Gregory Neven, and Patrick Towa. Short threshold dynamic group signatures. In Clemente Galdi and Vladimir Kolesnikov, editors, *SCN 2020*, volume 12238 of *Lecture Notes in Computer Science*, pages 401–423. Springer, 2020.

- [19] Jan Camenisch and Jens Groth. Group signatures: Better efficiency and new theoretical aspects. In Carlo Blundo and Stelvio Cimato, editors, *SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 120–133. Springer, 2004.
- [20] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew K. Franklin, editor, *CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.
- [21] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *CRYPTO 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.
- [22] David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.
- [23] Rémi Clarisse and Olivier Sanders. Group signature without random oracles from randomizable signatures. In Khoa Nguyen, Wenling Wu, Kwok-Yan Lam, and Huaxiong Wang, editors, *ProvSec 2020*, volume 12505 of *Lecture Notes in Computer Science*, pages 3–23. Springer, 2020.
- [24] Ivan Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 1991.
- [25] Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *VIETCRYPT 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.
- [26] David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *AsiaCCS 2018*, pages 551–565. ACM, 2018.
- [27] Sebastian Faust, Markulf Kohlweiss, Giorgia Azzurra Marson, and Daniele Venturi. On the non-malleability of the fiat-shamir transform. In Steven D. Galbraith and Mridul Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *Lecture Notes in Computer Science*, pages 60–79. Springer, 2012.
- [28] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.
- [29] Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2005.
- [30] Marc Fischlin, Patrick Harasser, and Christian Janson. Signatures from sequential-or proofs. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 212–244. Springer, 2020.
- [31] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62. Springer, 2018.

- [32] Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discret. Appl. Math.*, 156(16):3113–3121, 2008.
- [33] Juan A. Garay, Philip D. MacKenzie, and Ke Yang. Strengthening zero-knowledge protocols using signatures. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 177–194. Springer, 2003.
- [34] Eu-Jin Goh, Stanislaw Jarecki, Jonathan Katz, and Nan Wang. Efficient signature schemes with tight reductions to the diffie-hellman problems. *J. Cryptol.*, 20(4):493–514, 2007.
- [35] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
- [36] Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007.
- [37] Christian Hanser and Daniel Slamanig. Structure-preserving signatures on equivalence classes and their application to anonymous credentials. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014*, volume 8873 of *Lecture Notes in Computer Science*, pages 491–511. Springer, 2014.
- [38] Intel. A cost-effective foundation for end-to-end iot security, white paper. <https://www.intel.in/content/www/in/en/internet-of-things/white-papers/iot-identity-intel-epid-iot-security-white-paper.html>, 2016.
- [39] Aggelos Kiayias and Moti Yung. Group signatures with efficient concurrent join. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 198–214. Springer, 2005.
- [40] Hyoseung Kim, Youngkyung Lee, Michel Abdalla, and Jong Hwan Park. Practical dynamic group signature with efficient concurrent joins and batch verifications. Cryptology ePrint Archive, Report 2020/921, 2020. <https://eprint.iacr.org/2020/921>.
- [41] Benoît Libert, Fabrice Mouhartem, Thomas Peters, and Moti Yung. Practical “signatures with efficient protocols” from simple assumptions. In Xiaofeng Chen, XiaoFeng Wang, and Xinyi Huang, editors, *AsiaCCS 2016*, pages 511–522. ACM, 2016.
- [42] Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In Rosario Gennaro and Matthew Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 296–316. Springer, 2015.
- [43] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In Harriet Ortiz, editor, *STOC 1990*, pages 427–437. ACM, 1990.
- [44] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT 1999*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
- [45] David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2016.

- [46] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptol.*, 13(3):361–396, 2000.
- [47] Yusuke Sakai, Jacob C. N. Schuldt, Keita Emura, Goichiro Hanaoka, and Kazuo Ohta. On the security of dynamic group signatures: Preventing signature hijacking. In Marc Fischlin, Johannes A. Buchmann, and Mark Manulis, editors, *PKC 2012*, volume 7293 of *Lecture Notes in Computer Science*, pages 715–732. Springer, 2012.
- [48] TCG. <https://trustedcomputinggroup.org/authentication/>, 2015.