Skipping the q in Group Signatures

Olivier Blazy[1] Saqib A. Kakvi[2]

[1] XLim, Université de Limoges
olivier.blazy@unilim.fr
[2] Department of Computer Science, University of Bristol
saqib.kakvi@bris.ac.uk

Abstract

The notion of group signatures was introduced to allow group members to sign anonymously on behalf of a group. A group manager allows a user to join a group, and another will be able to open a signature to revoke its anonymity. Several schemes have already been proposed to fulfil these properties, however very few of them are proven in the standard model. Of those proven in the standard model, most schemes rely on a so called q-assumption. The underlying idea of a q-assumptions is that to prove the security of the scheme, we are given a challenge long enough to allow the simulator to answer queries. Another common solution is to rely on interactive hypothesis. We provide one of the first schemes proven in the standard model, requiring a constant-size non-interactive hypothesis. We then compare its efficiency to existing schemes, and show that this trade-off is acceptable as most schemes with better efficiency rely on either an interactive or a q-hypothesis. The exception to this is the recent independent of our work Libert, Peters and Yung (CRYPTO 2015), who presented an efficient group signature scheme in the standard model relying on standard assumptions.

Keywords: signatures, group signatures, standard model, q-assumptions

1 Introduction

A group signature scheme [Cv91] is a protocol which lets a member of a group individually issue signatures on behalf of the group, in an anonymous but traceable way. We have an Opener who is able to revoke anonymity of the actual signer in case of abuse. Several steps have been made in the study of these protocols: Bellare, Micciancio and Warinschi [BMW03] gave formal definitions of the security properties of group signatures (the BMW model), and proposed a scheme under general assumptions. However, this model required that the size of the group be fixed a priori and may not change, known as static groups. Later, Bellare, Shi and Zhang [BSZ05] extended this model to dynamic groups (the BSZ model), which allows the group to grow arbitrarily large, emphasizing the importance of unforgeability and anonymity. Additionally, there was a similar model proposed by Kiayias and Yung [KY06], with slightly weaker requirements on the Opener.

Group signatures primarily guarantee *anonymity*, which means that nobody can link the signature to the signer, but also *unlinkability*, which means that one cannot tell whether two signatures have been produced by the same user. The exception to this is the Opener who can use their key to revoke anonymity.

The first efficient proposed group signature schemes were proven in the Random Oracle Model. One of the first standard model schemes was proposed by Camenish and Lysyanskaya [CL04]. Despite being fairly efficient, these schemes suffered from the drawback that the signatures were non-constant and would grow related to either the size of the group or the number of revoked users.

The first group signature with constant size was due to Groth [Gro06], but with an exceptionally large size. Soon after another scheme was proposed by Boyen and Waters [BW06], with more plausible sizes. Groth then improved on the scheme of [Gro06] in [Gro07] and provided not only an efficient group signature scheme, but also presented a generic approach consisting in using a re-randomisable certificate to produce a certified signature. Kakvi then proposed in [Kak10] some improvements that lead to a more efficient SXDH instantiation. The schemes due to Groth [Gro07] and Kakvi [Kak10] all rely on similar q-assumptions. Following this, Delerablée and Pointcheval proposed another short scheme based on a q-assumption [DP06]. This scheme was improved and extended by Blazy and Pointcheval in [BP12], with comparable efficiency while relying on only one, but a somewhat unclassical, q-assumption.

We build up from previous works and present a new group signature scheme, that does not rely on q-assumptions. We do this using a transformation that has been previously applied in other areas of cryptography. Although this work focuses on the case of group signatures, there is some scope to generalise it other constructions.

1.1 Related Work

In a recent independent work, Libert, Peters and Yung [LPY15] have proposed a group signatures scheme, which is secure without any q-type assumptions. They use the recent advances in structure preserving signatures [AFG$^+$10] as a principle building block. However their scheme is not generically constructed, whereas ours is. In spite of this, we maintain a signature size that is comparable to that of Libert Peters and Yung.

There have been other works looking at removing q-type assumptions from cryptographic primitives. The question of generically removing q-assumptions was studied by Chase and Meiklejohn [CM14]. The approach of Chase and and Meiklejohn [CM14] transforms schemes in a prime order pairing groups to schemes in composite order groups, reducing to a subgroup decisional problem. In the other direction, Bresson, Monnerat and Vergnaud [BMV08] showed separation between q-type assumptions and their non-q or simple variants, for the case of algebraic reductions.

1.2 Our Contribution.

In this work, we present a simple and efficient generic construction of group signatures that can be proven under reasonable assumptions in the standard model. In this paper we combine the use of a Delerablée-Pointcheval [DP06] certificate for Waters' signature [Wat05], and the Groth-Sahai [GS08] methodology. We describe our instantiation through the framework of Groth [Gro06,Gro07] for generic group signatures.

1.3 Organization

In the next section, we present the primitive of group signature and the security model, due to Bellare, Shi and Zhang [BSZ05]. Then, we present the basic tools upon which our instantiations rely. Next, we describe our scheme, in the SXDH setting[1], with the corresponding assumptions for the security analysis that is provided. Finally, we compare the size of our signature to previous schemes.

2 Preliminaries

2.1 Dynamic Group Signatures

We prove our scheme secure in the growing group security model of Bellare, Shi and Zhang [BSZ05], known as the BSZ model. The model requires that all users have their own personal signing/verification key pairs, which are all registered in a Public Key Infrastructure (PKI) i.e. any user $\mathcal{U}_i$ wishing to join the group owns a public-secret key pair $(\mathsf{upk}[i], \mathsf{usk}[i])$, certified by the PKI. Within our group signature setting, we have two distinct[2] authorities:

- The *Issuer* who adds new uses to the group and issues them with a group signing key and the corresponding certificate,
- The *Opener*, it is able to "open" any signature and extract the identity of the signer.

A group signature scheme is defined by a sequence of (interactive) protocols, $\mathsf{GS} = (\mathsf{Setup}, \mathsf{Join}, \mathsf{Sig}, \mathsf{Verif}, \mathsf{Open}, \mathsf{Judge})$, which are defined as follows:

- $\mathsf{Setup}(1^\lambda)$: Generates the group public key gpk, the issuer key ik for the Issuer, and the opening key ok for the Opener.
- $\mathsf{Join}(\mathcal{U}_i)$: This is an interactive protocol between a user $\mathcal{U}_i$ (who has their secret key $\mathsf{usk}[i]$) and the Issuer (using their private key ik). At the end of the protocol, the user obtains their group signing key sk_i, and the group manager adds the user to the registration list, Reg.
- $\mathsf{Sig}(\mathsf{pk}, m, \mathsf{sk}_i)$: Produces a group signature σ on the message m, under user U_i's group signing key sk_i.
- $\mathsf{Verif}(\mathsf{pk}, m, \sigma)$: Verifies the validity of the group signature σ, with respect to the public key pk. This algorithm thus outputs 1 iff the signature is valid.
- $\mathsf{Open}(\mathsf{pk}, m, \sigma, \mathsf{ok})$: If σ is valid, the Opener, using ok, outputs a user identity i assumed to be the signer of the signature with a proof τ of this accusation.
- $\mathsf{Judge}(\mathsf{pk}, m, \sigma, i, \tau)$: Verifies that the opening of σ to the identity i was indeed correctly done.

2.2 Security Notions

We now recap the BSZ security model [BSZ05]. The BSZ model requires group signature schemes to satisfy the following conditions, which we state informally:

- **Correctness:** All honest users must be able to join the group, receive their group signing key and use it to generate group signatures. Furthermore all correctly and honestly generated group signature should verify under the group public key and when opened by the opener should reveal the correct signer.
- **Anonymity:** No person, except the opener, should be able to extract the identity of the signer from any honestly generated group signature. Furthermore any group signatures a user produces should not be linkable to any other group signature generated by the same user.
- **Traceability:** Any signature should be traceable to the signer who made the signature, using the opener's secret key. In particular a corrupted Opener is unable to make a "false" opening and have it accepted by the Judge algorithm. In this scenario, we assume the issuer is fully honest.
- **Non-frameability:** No set of colluding group members can make a signature the will open to another honest user, even if they collude with both the Issuer and Opener.

For a more detailed discussion of the security requirements we refer the reader to Appendix A.

[1] For completeness and clarity, we provide a linear version of our scheme in Appendix B
[2] The BSZ model requires that both authorities must be distinct for certain notions of security. However, one could have them as the same entity in a relaxed version of the BSZ security model.

2.3 Computational Assumptions.

Our protocols will work with a pairing-friendly elliptic curve, of prime order:

- $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are multiplicative cyclic groups of finite prime order p, and g_1, g_2 are generators of $\mathbb{G}_1, \mathbb{G}_2$;
- e is a map from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$, that is bilinear and non-degenerated, such that $e(g_1, g_2)$ is generator of $\mathbb{G}_T$.

In particular we consider Type 3 group, as per the definitions of Galbraith, Paterson and Smart [GPS08]. For our purposes we will need the following assumptions.

Assumption 1 [Advanced Computational Diffie-Hellman [BFPV11]]
Let $\mathbb{G}_1, \mathbb{G}_2$ be multiplicative cyclic groups of order p generated by g_1, g_2 respectively, and e an admissible bilinear map $\mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The CDH^+ assumption states that given $(g_1, g_2, g_1^a, g_2^a, g_1^b)$, for random $a, b \in \mathbb{Z}_p$, it is hard to compute g_1^{ab}.

Assumption 2 [q-Double Hidden Strong Diffie-Hellman [FPV09]] Let $\mathbb{G}_1, \mathbb{G}_2$ be multiplicative cyclic groups of order p generated by g_1, g_2 respectively. The q-DHSDH problem consists given $(g_1, k_1, g_2, g_2^{\gamma})$ and several tuples of the form $(g_1^{x_i}, g_2^{x_i}, g_1^{y_i}, g_2^{y_i}, (k_1 g_1^{y_i})^{1/(\gamma+x_i)})_{i\in[1,q]}$ in computing $(g_1^x, g_2^x, g_1^y, g_2^y, (k_1 g_1^y)^{1/\gamma+x})$ for a new pair (x, y).

Assumption 3 [Double Hidden Strong Diffie-Hellman in $\mathbb{G}_1, \mathbb{G}_2$ [FPV09]] Let $\mathbb{G}_1, \mathbb{G}_2$ be multiplicative cyclic groups of order p generated by g_1, g_2 respectively. The DHSDH problem consists of, given $(g_1, k_1, g_2, g_2^{\gamma})$ in computing a tuple of the form $(g_1^x, g_2^x, g_1^y, g_2^y, (k_1 g_1^y)^{1/\gamma+x})$ for any pair (x, y).

We recall some computational assumptions used in other group signature schemes for completeness.

Assumption 4 [Decisional Diffie-Hellman Assumption in $\mathbb{G}$ [DH76]] Let $\mathbb{G}$ be a cyclic group of prime order p generated by g. The DDH assumption states it is infeasible to distinguish between the tuples $(g^a, g^b, g^a b)$ and (g^a, g^b, g^c) for random a, b, c.

Assumption 5 [Symmetric eXternal Diffie-Hellman [BBS04]] Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. The SXDH assumption states that the DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Assumption 6 [q-Strong Diffie-Hellman Assumption in $\mathbb{G}$ [BB04]] Let $\mathbb{G}$ be a cyclic group of order p generated by g. The q-SDH problem consists, given $(g, g^{\gamma}, g^{\gamma^2}, \ldots, g^{\gamma^q})$, in computing a pair $(x, g^{1/\gamma+x})$.

Assumption 7 [Decision Linear Assumption in $\mathbb{G}$ [BBS04]] Let $\mathbb{G}$ be a cyclic group of prime order, with generator g. The DLin assumption states that given $(g, g^x, g^y, g^{ax}, g^{by}, g^c)$, it is hard to decide if $c = a + b$ or not, for random $a, b, x, y \in \mathbb{Z}_p$.

Assumption 8 [Symmetric eXternal Decision Linear [BBS04]] Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order, $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. The SXDLin assumption states that the DLin assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

Assumption 9 [eXternal Decision Linear 1 Assumption [AFG$^+$10]] Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order, with generators (g_1, g_2), and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map.The XDLin_1 assumption states that given a tuple of the form $(g_1, g_1^x, g_1^y, g_1^{ax}, g_1^{by}, g_2, g_2^x, g_2^y, g_2^{ax}, g_2^{by}, g_1^c)$, it is hard to decide if $c = a + b$ or not, for random $a, b, x, y \in \mathbb{Z}_p$.

Assumption 10 [eXternal Decision Linear 2 Assumption [AFG$^+$10]] Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order, with generators (g_1, g_2), and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map.The XDLin_2 assumption states that given a tuple of the form $(g_1, g_1^x, g_1^y, g_1^{ax}, g_1^{by}, g_2, g_2^x, g_2^y, g_2^{ax}, g_2^{by}, g_2^c)$, it is hard to decide if $c = a + b$ or not, for random $a, b, x, y \in \mathbb{Z}_p$.

Assumption 11 [q-Simultaneous Flexible Pairing Assumption [AFG+10]] Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order, with generators (g_1, g_2), and $e\colon \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map.The q-SFP assumption states that given a tuple of the form $(g_1, g_1^{a_1}, g_1^{b_1}, g_2, g_2^{a_2}, g_2^{b_2}, g_2^{\zeta_1}, g_2^{\zeta_2}, g_2^{\rho}, g_2^{\eta})$, and several tuples of the form $(g_1^{z_i}, g_1^{r_i}, g_1^{t_i}, g_2^{s_i}, g_1^{h_i}, g_1^{v_i}, g_2^{w_i})_{i\in[1,q]}$, with:

$$e(g_1^{a_1}, g_2^{a_2}) = e(g_1^{z_i}, g_2^{\zeta_1})e(g_1^{r_i}, g_2^{\rho})e(g_1^{t_i}, g_2^{s_i})$$
$$e(g_1^{b_1}, g_2^{b_2}) = e(g_1^{z_i}, g_2^{\zeta_2})e(g_1^{h_i}, g_2^{\eta})e(g_1^{w_i}, g_2^{V_i})$$

it is hard to find a new tuple $(g_1^{z^*}, g_1^{r^*}, g_1^{t^*}, g_2^{s^*}, g_1^{h^*}, g_1^{v^*}, g_2^{w^*})$, satisfying these equations where we have that $z^* \notin \{0, z_1, \ldots, z_q\}$.

2.4 Certified Signatures

We use a primitive known as a certified signature scheme which was introduced by Boldyreva et al. [BFPW07]. A certified signature scheme is signature scheme where the well-formedness of the pubic key is verifiable due to an additional certificate. We use the BBS-like certification [BBS04] proposed by Delerablée and Pointcheval [DP06] to certify a Waters public key [Wat05]. When a receiver wishes to verify a certified signature, they will not only verify the signature, as per usual, but also verify the certificate of the well-formedness of the public key.

The security requirements for certified signatures is that we should neither be able to create a signature using a faked certificate key nor forge a signature for an already issued certificate. Although Boldyreva et al. provide more general security requirements, we use slightly simpler definitions, as in previous works. For a certified signature scheme to be secure, we require it to satisfy the following conditions:

- **Unfakeability:** No adversary should be able to produce a valid certificate for a key pair generated of their choice, even after having seen a polynomial number of certificates
- **Unforgeablity:** We require that the basic signature scheme satisfies at least the notion of existential unforgeability under weak message attack.

We will use a slight variant of the signature scheme due to Waters [Wat05] using a certificates as described by Delerablée and Pointcheval [DP06], which we refer to as the DPW scheme from here on in. We describe the scheme in Figure 1

algorithm $\mathsf{KeyGen}(1^k)$	**algorithm** Issue
$\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow_\$ \mathsf{Gen}(1^k)$ $\ell = poly(\lambda)$ $\gamma \in_R \mathbb{Z}_p$, $\mathit{\Gamma} = g_2^{\gamma}$ $k_1 \in_R \mathbb{G}_1$ $\mathbf{u} \in_R \mathbb{G}_1^{\ell}$ //description of $\mathcal{F}$ return $(\mathsf{ak}, \mathsf{ck}) = ((\mathsf{gk}, \mathit{\Gamma}, k_1, \mathcal{F}), (\mathsf{ak}, \gamma))$	User / Issuer User: $y' \in_R \mathbb{Z}_p$ $\xrightarrow{g_1^{y'}, g_2^{y'}}$ Issuer: $x, y'' \in_R \mathbb{Z}_p$ Issuer: $A = (k_1 g_1^{y'} g_1^{y''})^{\frac{1}{x+\gamma}}$ Issuer: $\mathsf{cert} = (g_1^x, g_2^x, A)$ $\xleftarrow{y'', \mathsf{cert}}$ User: $\mathsf{sk} = y' + y''$ User: $\mathsf{pk} = (g_1^{\mathsf{sk}}, g_2^{\mathsf{sk}})$ User: return $(\mathsf{pk}, \mathsf{cert}, \mathsf{sk})$ — Issuer: return $(\mathsf{pk}, \mathsf{cert})$
algorithm $\mathsf{Sign}(\mathsf{pk}, \mathsf{sk}, m)$	**algorithm** $\mathsf{Verify}(\mathsf{pk}, \mathsf{ak}, \mathsf{cert}, m, \sigma)$
$s \in_R \mathbb{Z}_p$ $\sigma_1 = h^{\mathsf{sk}}\mathcal{F}(m)^s$ $\sigma_2 = g_1^s$ $\sigma_3 = g_2^s$ return $\sigma = (\sigma_1, \sigma_2, \sigma_3)$	return 1 if $e(\mathsf{cert}_1, g_2) = e(g_1, \mathsf{cert}_2)\wedge$ $e(\mathsf{cert}_3, \mathsf{ak}_2\mathsf{pk}_2) = e(k_1, g_2)e(g_1, \mathsf{pk}_2)\wedge$ $e(\mathsf{cert}_2, g_2) = e(g_1, \sigma_3)\wedge$ $e(\sigma_2, g_2) = e(g_1, \sigma_3)\wedge$ $e(\sigma_1, g_2) = e(h_1, \mathsf{pk}_2)e(\mathcal{F}(m), \sigma_3)$ else return 0

Fig. 1. The Delerablée-Pointcheval Certified Waters Signature Scheme.

The DPW Scheme was shown to be secure under the q-DHSDH, and CDH^+ assumptions. In Section 3 we will present a modification of this scheme that such that we can prove the security under the DHSDH and CDH^+ assumptions.

2.5 Groth-Sahai Commitments.

We will follow the Groth-Sahai methodology for SXDH-based commitment in the SXDH setting. The commitment key consists of $\mathbf{u} \in \mathbb{G}_1^{2\times2}$ and $\mathbf{v} \in \mathbb{G}_2^{2\times2}$. There exist two initialisations of the parameters; either in the perfectly binding setting, or in the perfectly hiding one. Those initialisations are indistinguishable under the SXDH assumption which will be used in the simulation. We denote by $\mathcal{C}(X)$ a commitment of a group element X. An element is always committed in the group ($\mathbb{G}_1$ or $\mathbb{G}_2$) it belongs to. If one knows the commitment key in the perfectly binding setting, one can extract the value of X, else it is perfectly hidden. We note $\mathcal{C}^{(1)}(x)$ a commitment of a scalar x embedded in $\mathbb{G}_1$ as g_1^x. If one knows the commitment key in the perfectly binding setting, on can extract the value of g_1^x else x is perfectly hidden, and analogously for $\mathbb{G}_2$.

Under the SXDH assumption, the two initializations of the commitment key (perfectly binding or perfectly hiding) are indistinguishable. The former provides perfectly sound proofs, whereas the latter provides perfectly witness hiding proofs. A Groth-Sahai proof, is a pair of elements $(\pi, \theta) \in \mathbb{G}_1^{2\times2} \times \mathbb{G}_2^{2\times2}$. These elements are constructed to help verifying pairing relations on committed values. Being able to produce a valid pair implies knowing plaintexts verifying the appropriate relation.

We will use three kinds of relations:

- pairing products equation which require 4 extra elements in each group;
- multi-scalar multiplication which require 2 elements in one group and 4 in the other;
- quadratic equations which require 2 elements in each group.

In the following, we will generate two Common Reference Strings to handle commitments and proofs under this methodology through the following algorithm:

- $\mathsf{GS.KeyGen}(\mathsf{gk})$: generates two commitment keys, and the associated extraction key xk. In our protocol, ck_B will provide perfectly binding commitments in both groups and while ck_H will provide perfectly hiding commitments in $\mathbb{G}_2$. Both commitment keys are added to crs.
- $\mathsf{C.Commit}(\mathsf{ck}_*, A)$: this produces a commitment to an element A under the key ck_* using some randomness r.
- $\mathsf{GS.Prove}(E, (\mathcal{C}, \mathsf{ck}_*))$: generates a Zero Knowledge Groth-Sahai Proof of Knowledge, the plaintexts committed in $\mathcal{C}$ under ck_* verifies some equation described in E. Such proofs, requires the of the previous randomness r, and can only be done directly if elements in a designated group are committed under the same ck_*. (This means that we have to be careful that, for a given equation, our commitments in $\mathbb{G}_2$ are done solely with ck_H or solely with ck_B.). This generates a proof π composed of several group elements.
- $\mathsf{GS.Verify}(\pi)$: verifies the validity of the proof π. To lighten the notation, we suppose that a proof π includes the previous $E, (\mathcal{C}, \mathsf{ck}_*)$. In other words, is π a valid proof that the plaintexts committed in $\mathcal{C}$ under ck_* are a valid solution to the equation described in E. Once again, to lighten the notation, we will denote $\mathsf{GS.Verify}(\pi_1, \pi_2, \ldots)$ the verification of several proofs, this can be done sequentially or using a batch technique, as presented in [BFI+10].
- $\mathsf{GS.Re\text{-}Randomize}(\mathsf{ck}_*, \mathcal{C}, \pi)$: rerandomizes the commitment $\mathcal{C}$ and then adapts the proof π. This step does not require the knowledge of the commitment randomness. When committing to a value previously public in an equation, it can be seen as randomizing a previous commitment to this variable where the randomness used was 0.
- $\mathsf{C.Extract}(\mathcal{C}, \mathsf{xk})$: extracts the plaintext A from $\mathcal{C}$ if $\mathcal{A}$ was committed in $\mathcal{C}$ under a binding key. The soundness of proof generated by Groth-Sahai methodology implies that if $\mathsf{GS.Verify}(E, (\mathcal{C}, \mathsf{ck}_B), \pi)$ holds, then we have that $\mathsf{C.Extract}(\mathcal{C}, \mathsf{xk})$ verifies the equation E.

2.6 A Classical Trick

Our construction will rely on a classical trick used on Groth-Sahai proof. In many e-cash papers, such as [CG07,BCKL09,LV09,FV10], the construction needs an anonymity property where the adversary should not be able to get any information on a coin while a judge should be able to extract information while in the same CRS. Another application around this idea was presented by Fischlin, Libert and Manulis in [FLM11] where the authors used it to provide a non-interactive technique to commit to elements in the UC framework.

In those cases, the solution proposed, is to commit twice to the value X, once with a perfectly binding commitment key, and once with a perfectly hiding key, and then proving the committed value X is the same in both. (While this is necessarily true because of the perfectly hiding commitment, under the Co-Soundness of Groth-Sahai proof, this is hard to do without the knowledge of trapdoors in the commitment key). To then use this X in the rest of the scheme, one simply builds proof using the perfectly hiding commitment.

We will employ exactly this trick in the context of group signatures. Most schemes rely on a q-assumption, or even an interactive assumption, to prove anonymity of the scheme. We use this trick to be able to prove anonymity without using either, thus achieving our goal.

3 Our Construction

3.1 Certified Signatures

We first present a variant of the Delerablée-Pointcheval Certified Waters Signature Scheme, using commitments, which we will call the DPWC scheme from here on in. In the DPWC scheme, instead of sending the certificate, the certificate authority will send commitments to the certificate, and a proof that the certificate is well-formed. The receiver must now verify the proof of well-formedness instead of the certificate. We can now show that the hardness of forging a certificate can be reduced to the soundness of the commitment scheme, which in turn is based upon the SXDH. Due to technical reasons, we need two common reference strings, one which is perfectly hiding and one which is perfectly binding. We present the DPWC scheme in Figure 2.

algorithm $\mathsf{KeyGen}(1^k)$

$\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow_\$ \mathsf{Gen}(1^\lambda)$
$\gamma \in_R \mathbb{Z}_p$, $\Gamma = g_2^\gamma$
$k_1 \in_R \mathbb{G}_1, h_2, \mathcal{F} \in_R \mathbb{G}_2^{\ell+2}$
$(\mathsf{ck}_B, \mathsf{ck}_H, \mathsf{xk}) \leftarrow_\$ \mathsf{GS.KeyGen}(\mathsf{gk})$
$(\mathsf{ak}) = (\mathsf{gk}, \Gamma, k, \mathcal{F}, \mathsf{ck}_B, ck_H, \mathsf{crs})$
$(\mathsf{ck}) = (\mathsf{ak}, \gamma)$
return $(\mathsf{ak}, \mathsf{ck})$

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

$s \in_R \mathbb{Z}_p$
$\sigma_1 = h_2^{\mathsf{sk}} \mathcal{F}(m)^s$
$\sigma_2 = g_1^s$
$\sigma_3 = g_2^s$
return $\sigma = (\sigma_1, \sigma_2, \sigma_3)$

algorithm $\mathsf{Verify}(\mathsf{pk}, \mathsf{ak}, \mathsf{cert}, m, \sigma)$

return 1 if
  $\mathsf{GS.Verify}(\pi) == 1 \wedge$
  $e(\sigma_2, g_2) == e(g_1, \sigma_3) \wedge$
  $e(g_1, \sigma_1) == e(\mathsf{pk}_1, h_2) e(\sigma_2, \mathcal{F}(m))$
return 0 else

algorithm Join/Issue

User		Issuer
$y' \in_R \mathbb{Z}_p$	$\xrightarrow{g_1^{y'}}$	
		$y'', x \in_R \mathbb{Z}_p$
		$A = (k_1 g_1^{y'} g_1^{y''})^{\frac{1}{x+\gamma}}$
		$\alpha = \mathsf{C.Commit}(\mathsf{ck}_B, A)$
		$\chi = \mathsf{C.Commit}(\mathsf{ck}_H, g_2^x)$
		$X_1 = \mathsf{C.Commit}(ck_B, g_1^x)$
		$X_2 = \mathsf{C.Commit}(ck_B, g_2^x)$
		$\pi_1 = \mathsf{GS.Prove}(\alpha, \chi)$
		$\pi_2 = \mathsf{GS.Prove}(X_1, \chi)$
		$\pi_3 = \mathsf{GS.Prove}(X_1, X_2)$
		$X = (X_1, X_2)$
		$\pi = (\pi_1, \pi_2, \pi_3)$
		$\mathsf{cert} = (\alpha, \chi, X, \pi)$
	$\xleftarrow{y'', \mathsf{pk}, \mathsf{cert}}$	$\mathsf{pk} = (g_1^{y'_i} g_1^{y''_i}, g_2^{y'_i} g_2^{y''_i})$
$\mathsf{sk} = y' + y''$		
if $\mathsf{pk} \neq (g_1^{\mathsf{sk}}, g_2^{\mathsf{sk}})$		
return $\perp$		
if $\mathsf{GS.Verify}(\pi) \neq 1$		
return $\perp$		
else		
return $(\mathsf{pk}, \mathsf{cert}, \mathsf{sk})$		return $(\mathsf{pk}, \mathsf{cert})$

Fig. 2. The Delerablée-Pointcheval Certified Waters Signature Scheme with Commitments.

Theorem 1. *The DPWC scheme is a certified signature scheme with perfect correctness for all messages $m \in \{0,1\}^\ell$. It is unfakebale under the DHSDH assumption and unforgeable under the CDH^+ assumption.*

Proof. The correctness of the scheme follows from the correctness of the Waters signatures, the Delerablée-Pointcheval certification and the correctness of the Groth-Sahai NIZK scheme.

We now prove unfakeability, using the following lemma:

Lemma 1. *If an adversary can (q', t', ε')-break the unfakeability of the scheme, then we can (t, ε)-solve the Double Hidden Strong Diffie-Hellman (DHSDH) problem, with*

$$t \approx t' \qquad \textit{and} \qquad \varepsilon = \varepsilon.$$

Proof. We receive as an initial input the DHSDH challenge of the form $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, k_1, g_2, g_2^{\gamma}, e)$. We then generate new commitment keys and keys for the proof system, thus giving us the extraction keys for the commitments and the ability to simulate the proofs, using the CRS trapdoor. We send the challenge along with the commitment keys and public parameters for the proof system to the adversary. Note the these form a valid DPWC public key. The adversary will then make q queries to the KeyReg oracle, which will allow it to act as a user and receive a key and certificate. We pick random values y'', x as before. Now since we do not posses a valid certification key, we must simulate the certificate. To simulate a certificate, we pick a random value A and commit to it. We then simulate the zero-knowledge proofs of well-formedness. Since we never send the A values in the clear, the adversary will not realise this, hence we have a perfect simulation of the scheme. The adversary will then submit a faked certificate cert^*, a public key pk^*, message m^*, signature σ^*. We first verify the certified signature. If the adversary has produced a valid certified signature, then both the certificate and signature must be correctly formed. Using the extraction key on the binding commitments, we are able to extract the value A^* from faked certificate, along with the values $g_1^{x^*}, g_2^{x^*}, g_1^{y^*}, g_2^{y^*}$. We then submit $(g_1^{x^*}, g_2^{x^*}, g_1^{y^*}, g_2^{y^*}, A^*)$ our solution to the DHSDH problem. We note that we win with exactly the same probability as the adversary.

The unforgeability of the DPW scheme was shown to hold by Blazy et al. in [BFPV11]. We include their statement in here for completeness.

Lemma 2. *Given an adversary can (q', t', ε')-break the unforgeability of the scheme, then we can (t, ε)-solve the Advanced Computational Diffie-Hellman (CDH^+) problem, with*

$$t \approx t' \qquad \textit{and} \qquad \varepsilon = \Theta(\varepsilon / q' \sqrt{\ell})$$

where ℓ is the length of our messages.

Proof. The proof can be found in [BFPV11, Appendix D].

This concludes the proof.

3.2 Group Signature

Now that we have the DPWC scheme, we can begin to construct our group signature scheme. The naïve approach would be to simply to provide each user with a DPWC certificate and key pair and use those to produce in the normal manner. However, we can immediately see that these signatures are no longer unlinkable, as all the signatures from any user would have their DPWC certificate attached to it, along with the corresponding public key. This is remedied by treating the DPWC public key as part of the certificate and committing it as well during the Join/Issue protocol. When signing the user will re-randomize these commitments and the proofs.

However, the signatures are still linkable. This is due to the fact that given a pair of Waters signatures, one can check if they are signed using the same key or not. To resolve this problem, we use an idea due to Fischlin [Fis06] that a commitment to a signature and proof of well-formedness implies a signature. We apply this idea to the Waters signature and hence get commitments of our signature elements and proofs of their well-formedness. This "committed" signature and our re-randomized committed certificate and the relevant proofs are then sent as the group signature.

The Open procedure will use the extraction key xk to extract the certificate from a signature and then check if there is a registry entry with the same certificate. If a matching certificate is found, we know that this user must have made that signature and thus output this index. To prove that the opening was done correctly, we simply prove that the commitment stored in the registry and the one commitment from the signature contain the same certificate.

algorithm $\mathsf{KeyGen}(1^k)$

$\mathsf{gk} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e) \leftarrow_\$ \mathsf{Gen}(1^k)$
$\gamma \in_R \mathbb{Z}_p,\ \Gamma = g_2^\gamma$
$k_1 \in_R \mathbb{G}_1, h_2, \mathcal{F} \in_R \mathbb{G}_2^{\ell+2}$
$(\mathsf{ck}_B, \mathsf{ck}_H, \mathsf{xk}) \leftarrow_\$ \mathsf{GS.KeyGen}(\mathsf{gk})$
$\mathsf{gpk} = (\mathsf{gk}, \Gamma, k, \mathcal{F}, \mathsf{ck}_B, ck_H, \mathsf{crs})$
$\mathsf{ik} = (\mathsf{gpk}, \gamma)$
$\mathsf{ok} = (\mathsf{gpk}, \mathsf{xk})$
return $(\mathsf{gpk}, \mathsf{ik}, \mathsf{ok})$

algorithm $\mathsf{Sign}(\mathsf{gpk}, \mathsf{sk}_i, m)$

$s \in_R \mathbb{Z}_p$
$\mathsf{cert}'_i \leftarrow_\$ \mathsf{GS.Re\text{-}Randomize}(\mathsf{cert}_i)$
$Y_1 = \mathsf{GS.Re\text{-}Randomize}(ck_B, g_1^{sk_i}, \pi'_{i,1})$
$Y_2 = \mathsf{C.Commit}(ck_B, g_2^{sk_i})$
$\sigma_1 = \mathsf{C.Commit}(h_2^{\mathsf{sk}_i}\mathcal{F}(m)^s)$
$\sigma_2 = g_1^s$
$\sigma_3 = g_2^s$
$\tilde{\pi}_1 = \mathsf{GS.Prove}(Y_1, Y_2)$
$\tilde{\pi}_2 = \mathsf{GS.Prove}(\sigma_1, \sigma_2, Y_2)$
return $\sigma = (\sigma_1, \sigma_2, \sigma_3, \mathsf{cert}'_i, Y_1, Y_2, \tilde{\pi}_1, \tilde{\pi}_2)$

algorithm $\mathsf{Issue}(\mathsf{usk}[i], ik)$

User		Issuer
$y'_i \in_R \mathbb{Z}_p$	$\xrightarrow{g^{y'_i}}$	
		$y''_i, x_i \in_R \mathbb{Z}_p$
		$A_i = (k_1 g_1^{y'} g_1^{y''})^{\frac{1}{x_i+\gamma}}$
		$\alpha_i = \mathsf{C.Commit}(\mathsf{ck}_B, A_i)$
		$\chi_i = \mathsf{C.Commit}(\mathsf{ck}_H, g_2^{x_i})$
		$X_{i,1} = \mathsf{C.Commit}(ck_B, g_1^{x_i})$
		$X_{i,2} = \mathsf{C.Commit}(ck_B, g_2^{x_i})$
		$\pi_{i,1} = \mathsf{GS.Prove}(\alpha_i, \chi_i)$
		$\pi_{i,2} = \mathsf{GS.Prove}(X_{i,1}, \chi_i)$
		$\pi_{i,3} = \mathsf{GS.Prove}(X_{i,1}, X_{i,2})$
		$X_i = (X_{i,1}, X_{i,2})$
		$\pi_i = (\pi_{i,1}, \pi_{i,2}, \pi_{i,3})$
		$\mathsf{cert}_i = (\alpha_i, \chi_i, X_i, \pi_i)$
$\mathsf{sk}_i = y'_i + y''_i$	$\xleftarrow{\mathsf{cert}_i, y''_i}$	
if $\mathsf{GS.Verify}(\pi_i) \neq 1$		
return $\perp$		
$s_i = \mathsf{Sign}(\mathsf{usk}[i], \mathsf{cert}_i)$	$\xrightarrow{s_i}$	
return $(\mathsf{cert}_i, \mathsf{sk}_i)$		$\mathsf{Reg}[i] = (i, \mathsf{pk}[i], \mathsf{cert}_i, s_i)$

algorithm $\mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \sigma)$

$\mathsf{cert}^* \leftarrow \mathsf{C.Extract}(\mathsf{xk}, cert'_i)$
for$(i \in [1, n])$
  $\widehat{cert} \leftarrow \mathsf{C.Extract}(\mathsf{ok}, \mathsf{cert}_i)$
  $\widehat{x} \leftarrow \mathsf{C.Extract}(\mathsf{ok}, \mathsf{Reg}[i]_4)$
  if $\widehat{cert} == \mathsf{cert}^*_1$
    $\tau = \mathsf{GS.Prove}(cert, \mathsf{cert}^*)$
    return (i, τ)
endfor
return $(0, \perp)$

algorithm $\mathsf{Verify}(\mathsf{gpk}, m, \sigma)$

return $\mathsf{GS.Verify}(\pi'_i, \tilde{\pi}_1, \tilde{\pi}_2) \wedge e(\sigma_2, g_2) == e(g_1, \sigma_3)$

algorithm $\mathsf{Judge}(\mathsf{pk}, \mathsf{ak}, \mathsf{cert}, m, \sigma, \tau)$

return $\mathsf{GS.Verify}(\tau)$

Fig. 3. The Group Signature Scheme.

Theorem 2. *The scheme described in Figure 3 is a group signature scheme with perfect correctness. The scheme satisfies anonymity, traceablity and non-frameability under the* SXDH, DHSDH *and* CDH^+ *assumptions.*

Proof. We will now prove each of the statements individually. We begin with correctness. The correctness follows directly from the correctness of the DPWC certified signature scheme and the Groth-Sahai proof system.

Lemma 3. *The scheme described in Figure 3 is a group signature scheme with anonymity, under the Symmetric External Diffie-Hellman Assumption.*

Proof. The identifying information in the group signature is the commitment to the user's certificate and public key. If an adversary would be able to distinguish signatures made by one user from the other, then they would effectively have distinguished between the commitments of two known values, hence breaking the hiding property of our commitment scheme, and thus the Symmetric External Diffie-Hellman Assumption.

Lemma 4. *The scheme described in Figure 3 is a group signature scheme with traceability, under the Double Strong Hidden Diffie-Hellman Assumption.*

Proof. For an adversary to be able to succeed in the traceability game, a corrupted user must produce a *valid* group signature such that the Opener is unable to trace the signer, or is unable to prove that they have traced the signer. Since the Opener in this came is only partially corrupt, and hence follows the algorithm in Figure 3. By the soundness of our NIZK, there must exist a certificate cert^*, which we can extract using ok. By the unfakebility of the DPWC scheme, that is to say the DHSDH assumption, this certificate must be one that was created by the issuer, thus allowing us to trace the signer.

Once the signer has been traced, the Opener will the produce a proof that certificate contained in the signature does indeed belong to the i^{th} user. Recall that the the Opener is only partially corrupted and therefore follows the algorithm as set out. By completeness of our NIZK proof system, the Just will output 1. Hence any valid signature can be traced and this opening will be judged to be valid.

Lemma 5. *The scheme described in Figure 3 is a group signature scheme with non-frameability, under the Symmetric External Diffie-Hellman, Double Strong Hidden Diffie-Hellman and Advanced Computational Diffie-Hellman Assumptions.*

Proof. For an adversary to win the non-framebility game, they must produce a signature which will be correctly attributed to an honest user who did not produce this signature. To achieve this, an adversary must provide:

1. A valid signature under the user's public key
2. A valid committed certificate, with proofs
3. A valid proof that the signature is valid under the public key in the committed certificate

Item 2 can be easily obtained by the adversary as they are able to fully corrupt both the Opener and Issuer and obtain the correct certificates and the corresponding proof from there. Thus we now need to only consider how the Adversary produces the other two components. To this end, we consider two types of Adversaries, namely $\mathcal{NF}_1$ and $\mathcal{NF}_2$.

In addition to the above types of adversary, we must also consider an adversary who fakes a certificate for the targeted user and then performs a Type I or Type II attack. The adversary in this game has the capability to write to the registry and hence can replace the user's old certificate with their faked one. After this the user must perform a Type I or Type II attack as described above. Here we see that the adversary must first fake a certificate, hence breaking the unfakeability of the DPWC certified signature scheme, and thus the Double String Hidden Diffie-Hellman Assumption. After this, the adversary will proceed as a $\mathcal{NF}_1$ or $\mathcal{NF}_2$ and thus additionally break the Advanced Computational Diffie-Hellman Assumption or the Symmetric External Diffie-Hellman Assumption.

This concludes the proof.

4 Efficiency Comparison

We now look at the efficiency of our scheme in comparison to the state of the art in signature schemes. We begin with a look at the exact size of our signatures. We list the size of each component of our signature in the table below.

Component	σ_1	σ_2	σ_3	α	χ	X_1	X_2	π_1	π_2	π_3	Y_1	Y_2	$\tilde{\pi}_1$	$\tilde{\pi}_2$	TOTAL
$\mathbb{G}_1$	0	1	0	2	0	2	0	2	2	2	2	0	2	4	19
$\mathbb{G}_2$	2	0	1	0	2	0	2	4	2	2	0	2	2	4	23

Table 1. Group elements required for each signature component.

In our comparison we only consider the schemes in Type 3 groups secure in the BSZ model. The first efficient constant size group signature scheme secure in the BSZ model was proposed by Groth [Gro07], based on the DLin assumption in Type 1 groups. The generic construction of Groth [Gro07] was adapted to Type 2 and 3 groups by Kakvi [Kak10] and independently adapted to Type 3 groups by Libert, Peters and Yung [LPY15]. Blazy and Pointcheval [BP12] presented a special case of group signatures, building on the scheme of Delerablée and Pointcheval [DP06]. Additionally, Bernhard, Fuchsbauer and Ghadafi [BFG13] also presented an adapted version of group signatures for attestation. We consider in [LPY15] using DLin-based chameleon hash due to Hofheinz and Jager [HJ12], explicitly stated by Blazy et al. [BKKP15, Appendix A].

Similar to the work of Libert, Peters and Yung [LPY15], we compare not only the number of group elements, but the bit sizes. We consider the minimal bitsize when one group representation is 256 bits and the other is 512 bits.

Scheme	Assumptions	$\mathbb{G}_1$	$\mathbb{G}_2$	Signature Size Elements	Signature Size Bits
Adapated Groth [Gro07,LPY15]	SXDLin, q-SDH, q-U′	27	12	39	13056
Kakvi [Kak10] (Scheme 3)	SXDLin, q-SDH, q-U3a	24	15	39	13824
Kakvi [Kak10] (Scheme 4)	SXDLin, q-SDH, q-U3b	16	23	39	14080
Blazy and Pointcheval [BP12]	CDH^+, q-DDHI, q-DHSDH	21	16	37	13568
Bernhard, Fuchsbauer and Ghadafi [BFG13]	SXDH, CDH^+, q-SDH, q-DDHI, q-SFP	24	15	39	13824
Libert, Peters and Yung [LPY15]	SXDH, XDLin_2, DLin	30	14	44	14848
This Work	SXDH, CDH^+, DHSDH	19	23	42	15616

Table 2. Comparison of Group Signature Schemes secure in the Standard Model.

As we can see from the table, our signature sizes are comparable to that of the other schemes, but under standard assumptions. In particular, we have fewer elements than the scheme of Libert, Peters and Yung [LPY15], albeit with a marginally larger signature in one case. We have slightly larger signatures across the board when compared to the other schemes, but with the advantage of relying on standard assumptions. We believe that this trade-off between size and security is an acceptable one to make.

Acknowledgements

This work has been supported in part by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO. Part of this work was done while both authors were at the Ruhr-Universität Bochum.

References

AFG$^+$10. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, August 2010. (Cited on page 1, 3, 4.)

BB04. Dan Boneh and Xavier Boyen. Short signatures without random oracles. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 56–73. Springer, May 2004. (Cited on page 3.)

BBS04. Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 41–55. Springer, August 2004. (Cited on page 3, 4.)

BCKL09. Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. Compact e-cash and simulatable VRFs revisited. In Hovav Shacham and Brent Waters, editors, *PAIRING 2009*, volume 5671 of *LNCS*, pages 114–131. Springer, August 2009. (Cited on page 5.)

BFG13. David Bernhard, Georg Fuchsbauer, and Essam Ghadafi. Efficient signatures of knowledge and DAA in the standard model. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *ACNS 13*, volume 7954 of *LNCS*, pages 518–533. Springer, June 2013. (Cited on page 9, 10.)

BFI$^+$10. Olivier Blazy, Georg Fuchsbauer, Malika Izabachène, Amandine Jambert, Hervé Sibert, and Damien Vergnaud. Batch Groth-Sahai. In Jianying Zhou and Moti Yung, editors, *ACNS 10*, volume 6123 of *LNCS*, pages 218–235. Springer, June 2010. (Cited on page 5.)

BFPV11. Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Signatures on randomizable ciphertexts. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 403–422. Springer, March 2011. (Cited on page 3, 7.)

BFPW07. Alexandra Boldyreva, Marc Fischlin, Adriana Palacio, and Bogdan Warinschi. A closer look at PKI: Security and efficiency. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450 of *LNCS*, pages 458–475. Springer, April 2007. (Cited on page 4.)

BKKP15. Olivier Blazy, Saqib A. Kakvi, Eike Kiltz, and Jiaxin Pan. Tightly-secure signatures from chameleon hash functions. In Jonathan Katz, editor, *PKC 2015*, volume 9020 of *LNCS*, pages 256–279. Springer, March / April 2015. (Cited on page 9.)

BMV08. Emmanuel Bresson, Jean Monnerat, and Damien Vergnaud. Separation results on the "one-more" computational problems. In Tal Malkin, editor, *CT-RSA 2008*, volume 4964 of *LNCS*, pages 71–87. Springer, April 2008. (Cited on page 1.)

BMW03. 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 *LNCS*, pages 614–629. Springer, May 2003. (Cited on page 1.)

BP12. Olivier Blazy and David Pointcheval. Traceable signature with stepping capabilities. In David Naccache, editor, *Quisquater Festschrift*, LNCS. Springer, 2012. (Cited on page 1, 9, 10.)

BSZ05. 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 *LNCS*, pages 136–153. Springer, February 2005. (Cited on page 1, 2.)

BW06. Xavier Boyen and Brent Waters. Compact group signatures without random oracles. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 427–444. Springer, May / June 2006. (Cited on page 1.)

CG07. Sébastien Canard and Aline Gouget. Divisible e-cash systems can be truly anonymous. In Moni Naor, editor, *EUROCRYPT 2007*, volume 4515 of *LNCS*, pages 482–497. Springer, May 2007. (Cited on page 5.)

CL04. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, August 2004. (Cited on page 1.)

CM14. Melissa Chase and Sarah Meiklejohn. Déjà Q: Using dual systems to revisit q-type assumptions. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 622–639. Springer, May 2014. (Cited on page 1.)

Cv91. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, April 1991. (Cited on page 1.)

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976. (Cited on page 3.)

DP06. Cécile Delerablée and David Pointcheval. Dynamic fully anonymous short group signatures. In Phong Q. Nguyen, editor, *Progress in Cryptology - VIETCRYPT 06*, volume 4341 of *LNCS*, pages 193–210. Springer, September 2006. (Cited on page 1, 4, 9.)

Fis06. Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 60–77. Springer, August 2006. (Cited on page 7.)

FLM11. Marc Fischlin, Benoît Libert, and Mark Manulis. Non-interactive and re-usable universally composable string commitments with adaptive security. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 468–485. Springer, December 2011. (Cited on page 6.)

FPV09. Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 226–247. Springer, December 2009. (Cited on page 3.)

FV10. Georg Fuchsbauer and Damien Vergnaud. Fair blind signatures without random oracles. In Daniel J. Bernstein and Tanja Lange, editors, *AFRICACRYPT 10*, volume 6055 of *LNCS*, pages 16–33. Springer, May 2010. (Cited on page 5.)

GPS08. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008. (Cited on page 3.)

Gro06. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, December 2006. (Cited on page 1.)

Gro07. Jens Groth. Fully anonymous group signatures without random oracles. In Kaoru Kurosawa, editor, *ASIACRYPT 2007*, volume 4833 of *LNCS*, pages 164–180. Springer, December 2007. (Cited on page 1, 9, 10.)

GS08. Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer, April 2008. (Cited on page 1.)

HJ12. Dennis Hofheinz and Tibor Jager. Tightly secure signatures and public-key encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 590–607. Springer, August 2012. (Cited on page 9.)

Kak10. Saqib A. Kakvi. Eficienct fully anonymous group signatures based on the Groth group signature scheme. Master's thesis, University College London, 2010. (Cited on page 1, 9, 10.)

KY06. Aggelos Kiayias and Moti Yung. Secure scalable group signature with dynamic joins and separable authorities. *IJSN*, 1(1/2):24–45, 2006. (Cited on page 1.)

LPY15. 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 J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 296–316. Springer, August 2015. (Cited on page 1, 9, 10.)

LV09. Benoît Libert and Damien Vergnaud. Group signatures with verifier-local revocation and backward unlinkability in the standard model. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 498–517. Springer, December 2009. (Cited on page 5.)

Wat05. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 114–127. Springer, May 2005. (Cited on page 1, 4.)

A BSZ Security Model for Group Signatures

We now recall the BSZ security model for group signatures. As stated before, group signatures guarantee *anonymity, unlinkability* and *non-frameability*, which we will explain below. We require that we have two group authorities, namely the *Issuer*, who will issue certificates to grant access to the group, and an *Opener* who will be able to revoke anonymity, and thus trace back the actual signers. For different security notions, we allow each authority to be independently corrupted to some extent. We say an authority is fully corrupted if it reveals its key and potentially deviates from its algorithm. We say an authority is partially corrupted if it reveals its key but does not deviate from its algorithm. A uncorrupted authority neither reveals its key nor deviates from its algorithm. We give the maximum corruption levels allowed for each security property in Table 3 below

	Issuer	Opener
Traceability	Uncorrupted	Partially Corrupted
Anonymity	Fully Corrupted	Uncorrupted
Non-Frameabiity	Fully Corrupted	Fully Corrupted

Table 3. Trust levels of managers for security requirements.

Additionally, we assume that each user $\mathcal{U}_i$ owns a pair $(\mathsf{usk}[i], \mathsf{upk}[i])$ certified by a Public Key Infrastructure (PKI). We now recall the security notions.

A.1 Correctness

The *correctness* notion guarantees that honest users should be able to generate valid signatures, and the opener should then be able to get the identity of the signers, and provide a convincing proof for the judge. In the following experiments that formalize the security notions, the adversary can run the Join protocol:

- either through the joinP-oracle (passive join), which means that it creates an honest user for whom it does not know the secret keys: the index i is added to the HU (Honest Users) list. The adversary gets back the public part of the certificate $\mathsf{pk}[i]$;
- or through the joinA-oracle (active join), which means that it interacts with the group manager to create a user it will control: the index i is added to the CU (Corrupted Users) list. The adversary gets back the whole certificate $\mathsf{pk}[i]$, and $\mathsf{sk}[i]$.

For users whose secret keys are known to the adversary, we let the adversary play on their behalf. For honest users, the adversary can interact with them, granted some oracles:

- $\mathsf{corrupt}(i)$, if $i \in \mathsf{HU}$, provides the secret key $\mathsf{sk}[i]$ of this user. The adversary can now control it. The index i is then moved from HU to CU;
- $\mathsf{Sig}(i, m)$, if $i \in \mathsf{HU}$, plays as the honest user i would do in the signature process. Then i is appended to the list $\S[m]$.

A.2 Traceability

Traceability asserts that nobody should be able to produce a valid signature that cannot be opened in a valid and convincing way. We detail the tracebility experiment in Figure 4 below.

We define the advantage of adversary against traceability as:

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

and we say that a group signature scheme is *traceable* if, for any polynomial adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{tr}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ is negligible.

$$
\begin{array}{|l|}
\hline
\text{Experiment } \mathsf{Exp}^{\mathsf{tr}}_{\mathsf{GS},\mathcal{A}}(\lambda) \\
1.\ (\mathsf{pk}, \mathsf{msk}, \mathsf{skO}) \leftarrow \mathsf{Setup}(1^\lambda) \\
2.\ (m, \sigma) \leftarrow \mathcal{A}(\mathsf{pk} : \mathsf{joinA}, \mathsf{joinP}, \mathsf{corrupt}, \mathsf{Sig}, \mathsf{open}) \\
3.\ \texttt{IF } \mathsf{Verif}(\mathsf{pk}, m, \sigma) = 0, \texttt{ RETURN } 0 \\
4.\ \texttt{IF } \exists j \notin \mathsf{CU} \cup \S[m], \\
\qquad\quad \mathsf{Open}(\mathsf{pk}, m, \sigma, \mathsf{skO}) = (j, \Pi) \\
\qquad \texttt{RETURN } 1 \\
5.\ \texttt{ELSE RETURN } 0 \\
\hline
\end{array}
$$

Fig. 4. Traceability Experiment

A.3 Non-Frameability

Non-frameability guarantees that no dishonest player (even the authorities, i.e. the Issuer and the Opener, hence the keys msk and skO provided to the adversary) will be able to frame an honest user. That is to say an honest user that does not sign a message M should not be convincingly declared as a possible signer We detail the non-frameability experiment in Figure 5 below.

$$
\begin{array}{|l|}
\hline
\text{Experiment } \mathsf{Exp}^{\mathsf{nf}}_{\mathsf{GS},\mathcal{A}}(\lambda) \\
1.\ (\mathsf{pk}, \mathsf{msk}, \mathsf{skO}) \leftarrow \mathsf{Setup}(1^\lambda) \\
2.\ (m, \sigma) \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{msk}, \mathsf{skO} : \mathsf{joinP}, \mathsf{corrupt}, \mathsf{Sig}) \\
3.\ \texttt{IF } \mathsf{Verif}(\mathsf{pk}, m, \sigma) = 0 \texttt{ RETURN } 0 \\
4.\ \texttt{IF } \exists i \in \mathsf{HU} \setminus \S[m], \\
\qquad\quad \mathsf{Open}(\mathsf{pk}, m, \sigma, \mathsf{skO}) = (i, \Pi) \\
\qquad \texttt{RETURN } 1 \\
5.\ \texttt{ELSE RETURN } 0 \\
\hline
\end{array}
$$

Fig. 5. Non-Frameability Experiment

We define the advantage of an adversary against non-frameability as

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

and we say a group signature scheme is *non-frameable* if, for any polynomial adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{nf}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ is negligible.

A.4 Anonymity

Anonymity states that the signer of a message remains anonymous. In particular, given two of honest users i_0 and i_1, the adversary should not have any significant advantage in guessing which one of them have issued a valid signature. The adversary can interact with honest users as before (with Sig and $\mathsf{corrupt}$), but the challenge signature is generated using the interactive signature protocol Sign, where the adversary plays the role of the corrupted users, but honest users are activated to play their roles.

We define the advantage of an adversary against anonymity as:

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

and we say that a group signature scheme is *anonymous* for any polynomial adversary $\mathcal{A}$, the advantage $\mathsf{Adv}^{\mathsf{anon}}_{\mathsf{GS},\mathcal{A}}(\lambda)$ is negligible.

Experiment $\mathsf{Exp}^{\mathsf{anon}-b}_{\mathsf{GS},\mathcal{A}}(\lambda)$
1. $(\mathsf{pk}, \mathsf{msk}, \mathsf{skO}) \leftarrow \mathsf{Setup}(1^\lambda)$
2. $(m, i_0, i_1) \leftarrow \mathcal{A}(\mathtt{FIND}, \mathsf{pk}, \mathsf{msk} : \mathsf{joinP}, \mathsf{corrupt}, \mathsf{Sig})$
3. $\sigma \leftarrow \mathsf{Sign}(\mathsf{pk}, i_b, m, \mathsf{sk}[i])$
4. $b' \leftarrow \mathcal{A}(\mathtt{GUESS}, \sigma : \mathsf{joinP}, \mathsf{corrupt}, \mathsf{Sig})$
5. $\mathtt{IF}\ i_0 \notin \mathsf{HU}\ \mathtt{OR}\ i_1 \notin \mathsf{HU}\ \mathtt{RETURN}\ 0$
6. $\mathtt{RETURN}\ b'$

Fig. 6. Anonymity Experiment

B A Linear Version of Our Construction

Our construction can be directly transposed in a symmetric group, with Linear Commitments. Before we describe the group signature scheme, we briefly recall the asymmetric Waters signature scheme:

- $\mathsf{Setup}(1^k)$: The scheme needs a (asymmetric) pairing-friendly environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an admissible bilinear map, for groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$, of prime order p, generated by g_1, g_2 and $g_t = e(g_1, g_2)$ respectively. We will sign messages $M = (M_1, \ldots, M_k) \in \{0,1\}^k$. To this aim, we need a vector $\boldsymbol{u} = (u_0, \ldots, u_k) \stackrel{\$}{\leftarrow} \mathbb{G}_1^{k+1}$, and for convenience, we denote the *Waters Hash* as $\mathcal{F}(M) = u_0 \prod_{i=1}^k u_i^{M_i}$. We also need an additional generator $h_1 \stackrel{\$}{\leftarrow} \mathbb{G}_1$. The global parameters param consist of all these elements $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2, h_1, \boldsymbol{u})$.
- $\mathsf{KeyGen}(\mathsf{param})$: Chooses a random scalar $x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, which defines the public key as $(X_1, X_2) = (g_1^x, g_2^x)$, and the secret key as $\mathsf{sk} = Y = h_1^x$.
- $\mathsf{Sign}(\mathsf{sk} = Y, M; s)$: For some random $s \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, define the signature as $\sigma = \big(\sigma_1 = Y(\mathcal{F}(M))^s, \sigma_2 = g_1^{-s}, \sigma_3 = g_2^{-s}\big)$.
- $\mathsf{Verif}((X_1, X_2), M, \sigma)$: Checks whether $e(\sigma_1, g_2) \cdot e(\mathcal{F}(M), \sigma_3) = e(h_1, X_2)$, and $e(\sigma_2, g_2) = e(g_1, \sigma_3)$.

Now that we have the requisite components, we present the linear version of our group signatures in Figure 7.

Theorem 3. *The scheme described in Figure 7 is a group signature scheme with perfect correctness. The scheme satisfies anonymity, traceablity and non-frameability under the* DHSDH, DLin *and* CDH *assumptions.*

This can be proven following the idea of the asymmetric instantiations. We omit the proofs, as they are of minimal interest.

On the efficiency of this scheme There is always a trade-off in efficiency while instantiating on a symmetric group a scheme designed for an asymmetric one. verifying that two elements have the same discrete logarithm is way more efficient in a DLin setting because this becomes a linear equation while being a quadratic one in SXDH. However we will have equations with two CRS involved for the same group, and that is quite inefficient (approximately 13 elements for each proof).

Component	σ_1	σ_2	α	χ	X_1	X_2	π_1	π_2	π_3	Y_1	Y_2	$\tilde{\pi}_1$	$\tilde{\pi}_2$	TOTAL
$\mathbb{G}$	3	1	3	3	3	3	13	13	2	3	3	2	3	55

The table above gives a rough estimation of the cost of the symmetric instantiation of our scheme, while not being so efficient it is still in the same order of magnitude as existing group signatures schemes, but once again our hypotheses are neither interactive nor relying on q-assumptions.

algorithm $\mathsf{KeyGen}(1^k)$

$\mathsf{gk} = (p, \mathbb{G}, \mathbb{G}_T, g, e) \leftarrow_\$ \mathsf{Gen}(1^k)$
$\gamma \in_R \mathbb{Z}_p,\ \mathit{\Gamma} = g^\gamma$
$k, h, g_2 \in_R \mathbb{G}, \mathcal{F} \in_R \mathbb{G}^{\ell+1}$
$(\mathsf{ck}_B, \mathsf{ck}_H, \mathsf{xk}) \leftarrow_\$ \mathsf{GS.KeyGen}(\mathsf{gk})$
$(\mathsf{ak}) = (\mathsf{gk}, \mathit{\Gamma}, k, h, g_2, \mathcal{F}, \mathsf{ck}_B, ck_H, \mathsf{crs})$
$(\mathsf{ck}) = (\mathsf{ak}, \gamma)$
return $(\mathsf{ak}, \mathsf{ck})$

algorithm $\mathsf{Sign}(\mathsf{gpk}, \mathsf{sk}, m)$

$s \in_R \mathbb{Z}_p$
$\mathsf{cert}'_i \leftarrow_\$ \mathsf{GS.Re\text{-}Randomize}(\mathsf{cert}_i)$
$Y_1 = \mathsf{GS.Re\text{-}Randomize}(ck_B, g^{sk_i}, \pi'_{i,1})$
$Y_2 = \mathsf{C.Commit}(ck_B, g_2^{sk_i})$
$\sigma_1 = \mathsf{C.Commit}(h^y \mathcal{F}(m)^s)$
$\sigma_2 = g^s$
$\tilde{\pi}_1 = \mathsf{GS.Prove}(Y_1, Y_2)$
$\tilde{\pi}_2 = \mathsf{GS.Prove}(\sigma_1, \sigma_2, Y_2)$
return $\sigma = (\sigma_1, \sigma_2, \mathsf{cert}'_i, Y_1, Y_2, \tilde{\pi}_1, \tilde{\pi}_2)$

algorithm Issue

User		Issuer
$y'_1 \in_R \mathbb{Z}_p$	$\xrightarrow{g^{y'_1}}$	
		$y''_i, x_i \in_R \mathbb{Z}_p$
		$A_i = (k_1 g_1^{y'} g_1^{y''})^{\frac{1}{x_i+\gamma}}$
		$\alpha_i = \mathsf{C.Commit}(\mathsf{ck}_B, A_i)$
		$\chi_i = \mathsf{C.Commit}(\mathsf{ck}_H, g^{x_i})$
		$X_{i,1} = \mathsf{C.Commit}(\mathsf{ck}_B, g^{x_i})$
		$X_{i,2} = \mathsf{C.Commit}(\mathsf{ck}_B, g_2^{x_i})$
		$\pi_{i,1} = \mathsf{GS.Prove}(\alpha_i, \chi_i)$
		$\pi_{i,2} = \mathsf{GS.Prove}(X_{i,1}, \chi_i)$
		$\pi_{i,3} = \mathsf{GS.Prove}(X_{i,1}, X_{i,2})$
		$X_i = (X_{i,1}, X_{i,2})$
		$\pi_i = (\pi_{i,1}, \pi_{i,2}, \pi_{i,3})$
		$\mathsf{cert}_i = (\alpha_i, \chi_i, X_i, \pi_i)$
$\mathsf{sk}_i = y' + y''$	$\xleftarrow{\mathsf{cert}_i, y''_i}$	
if $\mathsf{GS.Verify}(\pi_i) \neq 1$		
return $\perp$		
$s_i = \mathsf{Sign}(\mathsf{sk}[i], \mathsf{cert}_i)$	$\xrightarrow{s_i}$	
return $(\mathsf{cert}_i, \mathsf{sk}_i)$		$\mathsf{Reg}[i] = (i, \mathsf{pk}[i], \mathsf{cert}_i, s_i)$

algorithm $\mathsf{Open}(\mathsf{gpk}, \mathsf{ok}, \sigma)$

$\mathsf{cert}^* \leftarrow \mathsf{C.Extract}(\mathsf{xk}, cert'_i)$
for$(i \in [1, n])$
    $\widehat{cert} \leftarrow \mathsf{C.Extract}(\mathsf{ok}, \mathsf{cert}_i)$
    $\widehat{x} \leftarrow \mathsf{C.Extract}(\mathsf{ok}, \mathsf{Reg}[i]_4)$
    if $\widehat{cert} == \mathsf{cert}^*_1$
        $\tau = \mathsf{GS.Prove}(cert, \mathsf{cert}^*)$
        return (i, τ)
endfor
return $(0, \perp)$

algorithm $\mathsf{Verify}(\mathsf{gpk}, m, \sigma)$

return $\mathsf{GS.Verify}(\pi'_i, \tilde{\pi}_1, \tilde{\pi}_2)$

algorithm $\mathsf{Judge}(\mathsf{pk}, \mathsf{ak}, \mathsf{cert}, m, \sigma, \tau)$

return $\mathsf{GS.Verify}(\tau)$

Fig. 7. The Symmetric Group Signature Scheme.