Efficient Traceable Signatures in the Standard Model

Benoît Libert[1] ⋆ and Moti Yung[2] ⋆⋆

[1] Université catholique de Louvain, Crypto Group (Belgium)
[2] Google Inc. and Columbia University (USA)

Abstract. Traceable signatures (TS), suggested by Kiayias, Tsiounis and Yung (Eurocrypt'04), extend group signatures to address various basic traceability issues beyond merely identifying the anonymous signer of a rogue signature. Namely, they enable the efficient tracing of all signatures produced by a misbehaving party without opening the identity of other parties. They also allow users to provably claim ownership of a previously signed anonymous signature. To date, known TS systems all rely on the random oracle model. In this work we present the first realization of the primitive that avoids resorting to the random oracle methodology in its security proofs. Furthermore, our realization's efficiency is comparable to that of nowadays' fastest and shortest standard model group signatures.

Keywords. Traceable signatures, anonymity, standard model.

1 Introduction

Group Signatures Background. Group signatures, introduced by Chaum and van Heyst [22], allow members of a group to sign messages without revealing their identity. When the necessity arises, an authority holding some privileged piece of information can "open" signatures and uncover the signer's identity. Such primitives find applications in electronic auctions or trusted computing platforms where anonymity is a central issue.

The first scalable coalition-resistant system was proposed by Ateniese *et al.* [4]. The recent years saw a continued interest in the primitive with the appearance of pairing-based constructions (e.g. [15, 43]). In general, when it comes to signatures, pairing has been employed to achieve two goals: (1) short signatures and (2) realizations in the standard model, not relying on the random oracle idealization. Notably, Boneh, Boyen and Shacham [15] showed the first scheme featuring signatures shorter than 200 bytes. Its security was analyzed in (a relaxation of) the model of Bellare, Micciancio and Warinschi (BMW) [7], which captures the requirements of group signatures in three properties but assumes static groups. The setting of dynamic groups was formalized by Bellare-Shi-Zhang (BSZ) [9] and, independently, by Kiayias-Yung [39] while efficient systems were given in [39, 43, 29, 26].

The aforementioned practical proposals all rely on the random oracle model [8]. In the standard model, the theoretical constructions of [7, 9] were "only" proofs of concept (plausibility results), since the main interest is in getting efficient schemes. Using improved non-interactive zero-knowledge (NIZK) techniques [35, 34] inspired by an earlier homomorphic encryption scheme [16], Boyen and Waters [19] showed a fairly efficient realization with logarithmic-size signatures in the static BMW model. They subsequently improved [20] it to get rid of the dependency of signatures' size on the group cardinality. Ateniese *et al.* [3] independently constructed another scheme relying on stronger interactive assumptions. Meanwhile, Groth [32] came up with constant-size signatures

⋆ This author acknowledges the Belgian National Fund for Scientific Research (F.R.S.-F.N.R.S.) for his "Chargé de recherches" fellowship and the BCRYPT Interuniversity Attraction Pole.

⋆⋆ A preliminary version of this paper appeared at the 3rd International Conference on Pairing-Based Cryptography (Pairing 2009, LNCS 5671) under the same title.

without random oracles in the (dynamic) BSZ model but signatures remained too long for practical use. In 2007, Groth showed [33] another standard model scheme with signatures shorter than 2 kB and full anonymity in the BSZ model.

Traceable Signatures. In group signatures, if we are given a member's name and his public key, scanning all signatures and verify which ones were signed by that member is only doable by revoking the anonymity of all signatures (in particular, signatures of honest users). To overcome this and allow further tracing properties, Kiayias, Tsiounis and Yung [38] introduced traceable signatures (TS). They still allow the group manager (GM) to open signatures individually. In addition, however, the GM can reveal a trapdoor allowing clerks to trace suspicious members' signatures without having to revoke anonymity of every single signature. Misbehaving users can thus be traced without affecting the anonymity of honest ones. Moreover, such a traceability results in increased scalability since tracing agents can run in parallel whereas traditional group signatures involve a centralized tracing authority[3]. Traceable signatures also support a mechanism enabling users to claim (and prove) the authorship of their own anonymously generated signatures.

Kiayias, Tsiounis and Yung (KTY) formalized the security of traceable signatures via three properties termed *misidentification security*, *non-frameability* and *anonymity*. They suggested a first implementation of the primitive (using the Fiat-Shamir heuristic [28] and thus the random oracle model) and proved its security under the Strong RSA and the Decision Diffie-Hellman assumptions. Later on, efficiency improvements were suggested by Ge and Tate [30]. Meanwhile, Nguyen and Safavi-Naini [43] and Choi, Park and Yung [23] gave pairing-based constructions with shorter signatures. More recently, Benjumea *el al.* [10] considered traceable signatures with extended capabilities in the multi-group setting and implemented them in the random oracle model.

Our Contribution. Constructions with security proofs in the random oracle model are known to sometimes have realizability problems [21]. In this paper we construct the first efficient traceable signature in the standard model, where we employ the Groth-Sahai [36] non-interactive witness indistinguishable (NIWI) proof systems as part of the construction. We prove it secure in the KTY sense under non-interactive (and thus falsifiable) assumptions.

As far as efficiency goes, our scheme is on par with most efficient standard model group signatures: for recommended parameters, we obtain signatures of less than 2.6 kB, which is close to the size of Groth's signatures [33] while both schemes have similar computational complexities for signing and verification. From a security standpoint, the two constructions rely on intractability assumptions of comparable strengths. Whereas Groth's system is proved anonymous in the strong sense (*i.e.*, where the adversary has access to an oracle that "removes" the anonymity of adversarially-chosen signatures), our basic scheme is anonymous in a weaker sense but readily extends – by applying the same twist as in [33] – to achieve the same anonymity level at a quite moderate additional cost: in this case, the signature size does not exceed 3 kB.

Our traceable signature system also allows users to non-interactively claim their own signatures in an abuse-free manner. In previous TS realizations, claims consist in zero-knowledge proofs that can be made non-interactive using the Fiat-Shamir transformation. In our setting, implementing claims using the Groth-Sahai techniques requires special care to make sure that dishonest group members will not be able to copy each other's claims. As a contribution of independent interest (and a novelty w.r.t. the proceedings version of this paper [40]), we thus extend the original model

[3] Group signatures with verifier-local revocation [17] are an exception as verification entails to publicly run some implicit tracing mechanism to make sure that the signer is not revoked.

of traceable signatures [38] in order to explicitly capture that honest users' claims cannot be copied by dishonest users. We then give a convenient way for signers to claim their signatures and non-malleably link their claims to a long-term public key which they previously registered in a PKI.

Related Work. Independently and concurrently, Chow [24] considered an extension of traceable signatures where a tracing trapdoor enables the reconstruction of a set of tags, each one of which identifies a signature from the traced user. Instead of collecting and scanning all signatures using the tracing trapdoor, the tracing agent recomputes signature-specific tags, which are sent to signature holders who have to compare them with deterministically-generated tags included in each signature.

This new kind of TS schemes was analyzed in both the random oracle model and in the standard model. While the construction of [24] yields a more efficient tracing mechanism, it only provides a weaker flavor of anonymity where the anonymity adversary is limited to observe a fixed number of signatures from each honest user. In addition, [24] does not consider how to keep non-interactive claims from being copied as we do.

Organization. In the following, section 2 first describes the model of the TS primitive and the various tools and assumptions that we use. The scheme is described in section 3 and its security results are proved in section 4.

2 Background

Throughout the paper, when S is a set, $x \stackrel{\$}{\leftarrow} S$ denotes the action of choosing x uniformly at random in S. By $a \in \mathsf{poly}(\lambda)$, we mean that a is a polynomial in λ while $b \in \mathsf{negl}(\lambda)$ says that b is a negligible function of λ (*i.e.*, a function that decreases faster than the inverse of any $a \in \mathsf{poly}(\lambda)$). When a and b are binary strings, $a||b$ stands for their concatenation.

2.1 Complexity Assumptions

We use groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order p and endowed with an efficiently computable map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ such that $e(g^a, h^b) = e(g, h)^{ab}$ for any elements $(g, h) \in \mathbb{G} \times \mathbb{G}$, $a, b \in \mathbb{Z}$ and $e(g, h) \neq 1_{\mathbb{G}_T}$ whenever $g, h \neq 1_{\mathbb{G}}$.

In this algebraic setting, we rely on hardness assumptions that are all non-interactive and thus falsifiable [42]. The first one, introduced by Boneh, Boyen and Shacham [15], allows constructing NIWI proofs as pointed out in [36].

Definition 1. *In a group $\mathbb{G} = \langle g \rangle$ of prime order $p > 2^\lambda$, the* **Decision Linear Problem** *(DLIN) is to distinguish the distributions $(g^a, g^b, g^{ac}, g^{bd}, g^{c+d})$ and $(g^a, g^b, g^{ac}, g^{bd}, g^z)$, with $a, b, c, d \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$, $z \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. The* **Decision Linear Assumption** *asserts that, for any PPT distinguisher $\mathcal{D}$,*

$$\begin{aligned}\mathbf{Adv}^{\mathrm{DLIN}}_{\mathbb{G},\mathcal{D}}(\lambda) = |\Pr[\mathcal{D}(g^a, g^b, g^{ac}, g^{bd}, g^{c+d}) = 1 | a, b, c, d \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*] \\ - \Pr[\mathcal{D}(g^a, g^b, g^{ac}, g^{bd}, g^z) = 1 | a, b, c, d \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*,\ z \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*]| \in \mathsf{negl}(\lambda).\end{aligned}$$

This problem amounts to deciding whether vectors $\vec{g_1} = (g^a, 1, g)$, $\vec{g_2} = (1, g^b, g)$ and $\vec{g_3}$ are linearly dependent or not.

We also use a variant, first considered by Boyen and Waters [20], of the Strong Diffie-Hellman assumption [13].

Definition 2 ([20]). *In a group $\mathbb{G}$ of prime order p, the* **ℓ-Hidden Strong Diffie-Hellman problem** *(ℓ-HSDH) is, given elements $(g, \Omega = g^{\omega}, u) \xleftarrow{\$} \mathbb{G}^3$ and ℓ triples $(g^{1/\omega+s_i}, g^{s_i}, u^{s_i})$ with $s_1, \ldots, s_\ell \in \mathbb{Z}_p^*$, to find another triple $(g^{1/\omega+s}, g^s, u^s)$ such that $s \neq s_i$ for $i = 1, \ldots, \ell$.*

We finally need a variant of the problem, called Triple Diffie-Hellman, recently considered by Belenkiy *et al.* [6].

Definition 3. *Let $\mathbb{G}$ be a group of prime order p. The* **(modified) ℓ-Triple Diffie-Hellman Problem** *(ℓ-mTDH) is, given $(g, g^a, g^b) \in \mathbb{G}^3$, for randomly chosen $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, and ℓ distinct pairs $(g^{1/(a+c_i)}, c_i)$ with $c_1, \ldots, c_\ell \in \mathbb{Z}_p^*$, to output a triple $(g^{\mu}, g^{b\mu}, g^{ab\mu})$ for some non-zero $\mu \in \mathbb{Z}_p^*$.*

The original Triple Diffie-Hellman problem [6] was to find a triple $(g^{a\mu}, g^{b\mu}, g^{ab\mu})$ given the same inputs. In the paper, we only need these inputs to comprise a single pair $(c, g^{1/(a+c)})$ (*i.e.*, $\ell = 1$). A related assumption, named BB-CDH [5], asserts the infeasibility of finding g^{ab} on input of (g^a, g^b) as well as pairs $(g^{1/(a+c_i)}, c_i)$ with $c_1, \ldots, c_\ell \in \mathbb{Z}_p^*$. Under the knowledge of exponent assumption (KEA)[4] [25], the ℓ-mTDH problem is equivalent to the BB-CDH problem. The hardness of ℓ-mTDH in generic groups is thus implied by the generic intractability of KEA [27, 1] and BB-CDH.

2.2 Model and Security Notions

As in [9, 39], we assume a PKI and require each user i to hold a private/public key pair $(\mathsf{usk}[i], \mathsf{upk}[i])$ for an ordinary signature scheme. The public key $\mathsf{upk}[i]$ must be properly certified before the user registers as a group member.

A traceable signature [38] consists of the following algorithms or protocols.

Setup: given a security parameter $\lambda \in \mathbb{N}$, this algorithm (possibly run by a trusted party) generates a group public key $\mathcal{Y}$, that is widely distributed, and the matching private key $\mathcal{S}$ which is handed to the group manager.

Join$^{(\mathrm{GM},\mathcal{U}_i)}$ **:** is an interactive protocol, between the group manager GM and the prospective user $\mathcal{U}_i$, whereby the latter obtains a membership secret sec_i, that nobody else knows, and a membership certificate cert_i. The GM stores the whole transcript in a database called $\mathsf{transcripts}$, which is a private database also containing the coin tosses that were used by the GM.

Sign: given a certificate membership cert_i, a membership secret sec_i and a message M, this algorithm outputs a traceable signature σ of M.

Verify: on input of a signature σ, a message M and a group public key $\mathcal{Y}$, this deterministic algorithm returns 0 or 1.

Open: takes as input a signature σ that verifies under the group public key $\mathcal{Y}$, the corresponding private key $\mathcal{S}$ and the database $\mathsf{transcripts}$ of all transcripts of join protocols. It outputs the identity i of a group member.

Reveal: takes in the group manager's private key $\mathcal{S}$, the index i of a group member and the join transcript $\mathsf{transcript}_i$ of user i. It outputs the latter's tracing trapdoor trace_i.

Trace: on input of a valid traceable signature σ, the group public key $\mathcal{Y}$ and a tracing trapdoor trace_i for user i, this algorithm outputs either 0 or 1.

Claim: takes as input the group public key $\mathcal{Y}$, a valid message-signature pair (M, σ) issued by user i, the latter's membership secret sec_i and certificate cert_i as well as his private key $\mathsf{usk}[i]$. The output is an authorship claim τ of user i for σ.

[4] This assumption states that, given $g, g^a \in \mathbb{G}$, the only way to generate a pair $(h, h^a) \in \mathbb{G}^2$ is to raise g and g^a to some power and thus know $x = \log_g(h)$.

Claim-Verify: given a group public key $\mathcal{Y}$, a message-signature pair (M, σ), a claim τ and the public key $\mathsf{upk}[i]$ of user i, this deterministic algorithm outputs 0 or 1.

Security properties are formalized by experiments where the adversary is granted access to oracles sharing certain variables:

- state: contains the join transcripts, membership certificates and secrets that have been defined so far.
- N is the number of users in the group.
- Sigs: is the database of signatures issued by the Q_{sig} oracle.
- Claims: is the database of signatures that were issued by the Q_{sig} oracle and subsequently claimed by the signer.
- Revs: is the set of members that have been the input of a Q_{reveal} query.
- U^p: is the set of honest users introduced in the system via a $Q_{\mathsf{p\text{-}join}}$ query.
- U^a: is the set of adversarially-controlled users in the system.
- U^b: is the set of users that were introduced by the adversary acting as a dishonest group manager. For such users, the transcript of the join protocol is leaked to the adversary.

For reasons that will become apparent in security definitions (more precisely, when defining security against framing attacks), we will consider an equivalence class for message-signature pairs. The model of non-frameability considered in [39, 23] implicitly captures a flavor of strong unforgeability [2] in that it can only be satisfied when adversaries are unable to randomize existing signatures and turn them into other signatures on the same message. Here, due to the use of NIWI proof systems where non-interactive proofs are publicly re-randomizable, we will need to consider a slightly relaxed flavor of non-frameability. To this end, we define an equivalence relation over the signature space. In our scheme, each signature will consist of a number of traceability values, several commitments and a set of proofs elements. We say that two message-signature pairs $(M_1, \sigma_1), (M_2, \sigma_2)$ belong to the *same* equivalence class, which we denote by $(M_1, \sigma_1) \equiv_s (M_2, \sigma_2)$, if they pertain to the same message (*i.e.*, $M_1 = M_2$) *and* they comprise identical traceability values.

The various oracles that adversaries are given access to are listed below.

- $Q_{\mathcal{Y}}$: returns the public information $(\mathsf{N}, \mathcal{Y})$ of the system.
- $Q_{\mathcal{S}}$: returns the group manager's private key and thereby allows the adversary to corrupt the latter.
- $Q_{\mathsf{p\text{-}join}}$: is an oracle that privately introduces new honest users in the group. It simulates the join protocol in private, adds index N into U^p, increases N by 1 and finally updates variables state and transcripts as $\mathsf{state} \leftarrow \mathsf{state}||(\mathsf{N}, \mathsf{transcript}_{\mathsf{N}}, \mathsf{cert}_{\mathsf{N}}, \mathsf{sec}_{\mathsf{N}})$ and $\mathsf{transcripts} \leftarrow \mathsf{transcripts}||(\mathsf{N}, \mathsf{transcript}_{\mathsf{N}})$.
- $Q_{\mathsf{a\text{-}join}}$: allows the adversary to introduce users under his control in the group. The oracle, acting as the group manager, interacts with the malicious prospective user in the join protocol. If the protocol successfully terminates, the oracle increments N and finally sets $\mathsf{state} \leftarrow \mathsf{state}||(\mathsf{N}, \mathsf{transcript}_{\mathsf{N}}, \mathsf{cert}_{\mathsf{N}}, \perp)$, $\mathsf{transcripts} \leftarrow \mathsf{transcripts}||(\mathsf{N}, \mathsf{transcript}_{\mathsf{N}})$ and adds N into U^a.
- $Q_{\mathsf{b\text{-}join}}$: allows the adversary, acting as a dishonest group manager, to introduce new group members. The oracle, acting on behalf of the prospective user, interacts with the malicious group manager in the join protocol. If the latter successfully terminates, the oracle increases N by 1, sets $\mathsf{state} \leftarrow \mathsf{state}||(\mathsf{N}, \mathsf{transcript}_{\mathsf{N}}, \mathsf{cert}_{\mathsf{N}}, \perp)$, and adds N into U^b.

- Q_{sig}: on input of a message M and a user index i, the oracle checks if state contains an entry of the form $(i, \cdot, \mathsf{cert}_i, \mathsf{sec}_i)$. If no such record is found or if $i \in U^a$, it returns $\perp$. Otherwise, it generates and returns a traceable signature on behalf of user i using cert_i and sec_i. It also sets $\mathsf{Sigs} \leftarrow \mathsf{Sigs}||(i, M, \sigma)$.
- Q_{Claim}: on input of a triple (i, M, σ), this oracle first checks whether i belongs to the set of good users (which is either U^p or U^b depending on the considered security notion) and whether a triple (i, M', σ') such that $(M', \sigma') \equiv_v (M, \sigma)$ appears in Sigs. If either of these conditions fails to hold (*i.e.*, if user i is not an honest user or did not generate (M, σ)), it returns $\perp$ and sets $\mathsf{Claims} \leftarrow \mathsf{Claims}||(i, M, \sigma)$. Otherwise, it outputs a non-interactive authorship claim τ for the pair (M, σ) on behalf of user i and also sets $\mathsf{Claims} \leftarrow \mathsf{Claims}||(i, M, \sigma)$.
- Q_{reveal}: on input of a user index i, this oracle returns $\perp$ if user i does not exist or if $i \in U^b$. Otherwise, it returns the output of $\mathsf{Reveal}(i, \mathsf{transcripts})$ and adds i to Revs.

Misidentification Attacks. In a misidentification attack, the adversary is allowed to control a number of group members, which are introduced by invoking the $Q_{\mathsf{a\text{-}join}}$ oracle. Through the $Q_{\mathsf{p\text{-}join}}$ and Q_{sig} oracles, he can observe operations while users are added and generate signatures. She is also given access to users' tracing information via the Q_{reveal} oracle. His goal is to produce a non-trivial valid signature that does not open to any of the users under his control or that cannot be traced back to one of them.

Definition 4. *A traceable signature is secure against misidentification attacks if, for any PPT adversary $\mathcal{A}$, it holds that $\mathbf{Adv}^{\text{mis-id}}_{\mathcal{A}}(\lambda) = \Pr[\mathbf{Expt}^{\text{mis-id}}_{\mathcal{A}}(\lambda) = 1] \in \mathsf{negl}(\lambda)$ in the experiment below.*

Experiment $\mathbf{Expt}^{\text{mis-id}}_{\mathcal{A}}(\lambda)$
1. $(\mathcal{Y}, \mathcal{S}) \leftarrow \mathsf{Setup}(\lambda)$;
2. $(M^\star, \sigma^\star) \leftarrow \mathcal{A}(Q_{\mathcal{Y}}, Q_{\mathsf{p\text{-}join}}, Q_{\mathsf{a\text{-}join}}, Q_{\mathsf{sig}}, Q_{\mathsf{reveal}})$;
3. *If* $\mathsf{Verify}(M^\star, \sigma^\star, \mathcal{Y}) = 0$ *then return* 0;
4. *If* $\big((\mathsf{Open}(\sigma^\star, \mathcal{Y}, \mathcal{S}) \notin U^a) \vee (\bigwedge_{i \in U^a} \mathsf{Trace}(\sigma^\star, \mathsf{Reveal}(i)) = 0)\big)$ $\wedge \big(\bigwedge_{i \in U^p} (i, M^\star, *) \notin \mathsf{Sigs}\big)$ *then return* 1;
5. *Return* 0;

Framing Attacks. In a framing attack, the adversary can corrupt the group manager (via the $Q_{\mathcal{S}}$ oracle) and observe the system while users are added and produce signatures. Two kinds of framing attacks are considered. First, the adversary is deemed successful if he manages to produce a signature that opens or traces to an innocent group member. Second, he also wins if he can either (1) forge an honest signer's claim w.r.t. that signer's long term public key; (2) successfully claim a signature produced (and possibly claimed) by another user as his own using her own long term public key.

Definition 5. *A traceable signature is secure against framing attacks if, for any PPT adversary $\mathcal{A}$ involved in the experiment hereafter, $\mathbf{Adv}^{\text{fra}}_{\mathcal{A}}(\lambda) = \Pr[\mathbf{Expt}^{\text{fra}}_{\mathcal{A}}(\lambda) = 1]$ is negligible.*

Experiment $\mathbf{Expt}^{\mathrm{fra}}_{\mathcal{A}}(\lambda)$
1. $(\mathcal{Y}, \mathcal{S}) \leftarrow \mathsf{Setup}(\lambda)$;
2. $(M^\star, \sigma^\star, \tau^\star, \mathsf{upk}^\star) \leftarrow \mathcal{A}(Q_{\mathcal{Y}}, Q_{\mathcal{S}}, Q_{\mathsf{b\text{-}join}}, Q_{\mathsf{sig}})$;
3. *If* $\mathsf{Verify}(M^\star, \sigma^\star, \mathcal{Y}) = 0$ *then return* 0;
4. *If* $\big((\mathsf{Open}(\sigma^\star, \mathcal{Y}, \mathcal{S}) = i \in U^b) \vee (\exists i \in U^b \ s.t. \ \mathsf{Trace}(\sigma^\star, \mathsf{Reveal}(i)) = 1)\big)$
$\wedge\big(\nexists (i, M, \sigma) \in \mathsf{Sigs} \ s.t. \ (M^\star, \sigma^\star) \equiv_s (M, \sigma)\big)$ *then return* 1;
5. *If* $\big(\exists i \in U^b \ s.t. \ (i, M, \sigma) \in \mathsf{Sigs} \wedge (M^\star, \sigma^\star) \equiv_s (M, \sigma)\big)$
$\wedge\big(\nexists (j, M, \sigma) \in \mathsf{Claims} \ s.t. \ (j = i) \wedge (M^\star, \sigma^\star) \equiv_s (M, \sigma)\big)$
$\wedge\big(\mathsf{Claim\text{-}Verify}(M^\star, \sigma^\star, \tau^\star, \mathsf{upk}[i], \mathcal{Y}) = 1 \wedge \mathsf{upk}^\star = \mathsf{upk}[i]\big)$ *then return* 1 ;
6. *If* $\big(\exists i \in U^b \ s.t. \ (i, M, \sigma) \in \mathsf{Sigs} \wedge (M^\star, \sigma^\star) \equiv_s (M, \sigma)\big)$
$\wedge\big(\mathsf{Claim\text{-}Verify}(M^\star, \sigma^\star, \tau^\star, \mathsf{upk}^\star, \mathcal{Y}) = 1 \wedge \mathsf{upk}^\star \neq \mathsf{upk}[i]\big)$ *then return* 1 ;
7. *Return* 0;

In the above experiment, condition 5 captures the infeasibility of claiming signatures without knowing the appropriate key material. Condition 6 deals with the situation of the adversary illegally claiming the authorship of some honest group member's signature regardless of whether that member has already claimed his signature or not. This notably implies that no two members should be able to successfully claim the same signature.

Anonymity. An anonymity adversary runs in two stages called play and guess. In the first one, the adversary is allowed to join the system via $Q_{\mathsf{a\text{-}join}}$-queries on polynomially-many occasions. Using the $Q_{\mathsf{p\text{-}join}}$, Q_{sig} oracles, he can observe the system while users are privately introduced and sign messages. He can finally obtain tracing trapdoors for users of his choice. At the end of the play stage, he chooses two privately introduced users $i_0^\star, i_1^\star$ that were not the input of a Q_{reveal}-query and obtains a signature on behalf of one of them. In the guess stage, he aims at finding out who the signer was among $i_0^\star$ and $i_1^\star$.

In comparison with the definition of anonymity in [38], we introduce a claiming oracle Q_{claim} and enable the adversary to request claims for honestly generated signatures. Of course, the adversary is not allowed to obtain claims on behalf of $i_0^\star$ and $i_1^\star$ for signatures that belong to the same equivalence class as the challenge pair $(M^\star, \sigma^\star)$.

Definition 6. *A traceable signature scheme provides anonymity if, for any PPT adversary $\mathcal{A}$, we have* $\mathbf{Adv}^{\mathrm{anon}}(\mathcal{A}) := |\Pr[\mathbf{Expt}^{\mathrm{anon}}_{\mathcal{A}}(\lambda) = 1] - 1/2| \in \mathsf{negl}(\lambda)$, *where*

Experiment $\mathbf{Expt}^{\mathrm{anon}}_{\mathcal{A}}(\lambda)$
1. $(\mathcal{Y}, \mathcal{S}) \leftarrow \mathsf{Setup}(\lambda)$;
2. $(aux, M^\star, i_0^\star, i_1^\star) \leftarrow \mathcal{A}(\mathsf{play} : Q_{\mathcal{Y}}, Q_{\mathsf{p\text{-}join}}, Q_{\mathsf{a\text{-}join}}, Q_{\mathsf{sig}}, Q_{\mathsf{reveal}}, Q_{\mathsf{claim}})$;
3. *If* $(i_0^\star \notin U^p) \vee (i_1^\star \notin U^p) \vee (i_0^\star \in \mathsf{Revs}) \vee (i_1^\star \in \mathsf{Revs})$ *then return* 0;
4. $d^\star \xleftarrow{\$} \{0,1\}$; $\sigma^\star \leftarrow \mathsf{Sign}(M^\star, \mathcal{Y}, \mathsf{cert}_{i_{d^\star}^\star}, \mathsf{sec}_{i_{d^\star}^\star})$;
5. $d' \leftarrow \mathcal{A}(\mathsf{guess}, \sigma^\star, aux : Q_{\mathcal{Y}}, Q_{\mathsf{p\text{-}join}}, Q_{\mathsf{a\text{-}join}}, Q_{\mathsf{sig}}, Q_{\mathsf{reveal}}, Q_{\mathsf{claim}})$;
6. *If* $\Big(\big((i_0^\star \in \mathsf{Revs}) \vee (i_1^\star \in \mathsf{Revs})\big)$
$\vee\big(\exists (j, M, \sigma) \in \mathsf{Claims} \ s.t. \ j \in \{i_0^\star, i_1^\star\} \wedge (M, \sigma) \equiv_s (M^\star, \sigma^\star)\big)\Big)$ *then return* 0;
7. *If* $d' = d^\star$ *then return* 1;
8. *Return* 0;

The KTY model does not provide adversaries with an opening oracle in the definition of anonymity. On the other hand, since tracing is a distributed operation, the model considers (via the Q_{reveal}

oracle) the threat of corrupted tracing agents. In the following, we will stick to that model. In applications where anonymity should be preserved when opening queries are allowed, it is not hard to modify our scheme (using the technique of [33]) to obtain anonymity in the CCA2 sense.

2.3 Groth-Sahai Commitments

In the following, for equal-dimension vectors or matrices A and B containing group elements, $A \odot B$ stands for their component-wise product.

When based on the DLIN assumption, the Groth-Sahai proof systems [36] make use of a common reference string comprising vectors $\vec{g_1}, \vec{g_2}, \vec{g_3} \in \mathbb{G}^3$ where, for some group elements $g_1, g_2 \in \mathbb{G}$, $\vec{g_1} = (g_1, 1, g)$, $\vec{g_2} = (1, g_2, g)$. To commit to a group element $X \in \mathbb{G}$, one picks $r, s, t \xleftarrow{\$} \mathbb{Z}_p^*$ and computes $\vec{C} = (1, 1, X) \odot \vec{g_1}^r \odot \vec{g_2}^s \odot \vec{g_3}^t$. When the proof system is chosen to provide perfectly sound proofs, $\vec{g_3}$ is chosen as $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$ with $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p^*$. Commitments are then Boneh-Boyen-Shacham (BBS) encryptions since $\vec{C} = (g_1^{r+\xi_1 t}, g_2^{s+\xi_2 t}, X \cdot g^{r+s+t(\xi_1+\xi_2)})$ and decryption is possible using $\alpha_1 = \log_g(g_1)$, $\alpha_2 = \log_g(g_2)$. In the WI setting, $\vec{g_1}, \vec{g_2}, \vec{g_3}$ are linearly independent and $\vec{C}$ is a perfectly hiding commitment. Under the DLIN assumption, the two reference strings are indistinguishable.

To commit to exponents $x \in \mathbb{Z}_p$, one uses vectors $\vec{\varphi}, \vec{g_1}, \vec{g_2}$ and computes $\vec{C} = \vec{\varphi}^x \odot \vec{g_1}^r \odot \vec{g_2}^s$. In the soundness setting $\vec{\varphi}, \vec{g_1}, \vec{g_2}$ are linearly independent vectors whereas, in the WI setting, choosing $\vec{\varphi} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$ always gives a perfectly hiding commitment given that $\vec{C}$ is a BBS encryption of $1_{\mathbb{G}}$ regardless of the value x.

To provide evidence that committed variables satisfy a set of relations, the proof systems of [36] start from the relations themselves and replace variables by commitments. The prover then generates a proof (consisting of a set of group elements) for each relation. The whole proof consists of one commitment per variable and one proof for each relation. Such efficient non-interactive proofs are available for pairing-product relations, which are of the type

$$\prod_{i=1}^{n} e(\mathcal{A}_i, \mathcal{X}_i) \cdot \prod_{i=1}^{n} \cdot \prod_{j=1}^{n} e(\mathcal{X}_i, \mathcal{X}_j)^{a_{ij}} = t_T,$$

for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$ and constants $t_T \in \mathbb{G}_T$, $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathbb{G}$, $a_{ij} \in \mathbb{G}$, for $i, j \in \{1, \ldots, n\}$. Efficient proofs also exist for multi-exponentiation equations

$$\prod_{i=1}^{m} \mathcal{A}_i^{y_i} \cdot \prod_{j=1}^{n} \mathcal{X}_j^{b_j} \cdot \prod_{i=1}^{m} \cdot \prod_{j=1}^{n} \mathcal{X}_j^{y_i \gamma_{ij}} = T,$$

for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$, $y_1, \ldots, y_m \in \mathbb{Z}_p$ and constants $T, \mathcal{A}_1, \ldots, \mathcal{A}_m \in \mathbb{G}$, $b_1, \ldots, b_n \in \mathbb{Z}_p$ and $\gamma_{ij} \in \mathbb{G}$, for $i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}$.

Multi-exponentiation equations admit zero-knowledge proofs at no additional cost. On a simulated CRS (prepared for the WI setting), a trapdoor makes it is possible to simulate proofs without knowing witnesses and simulated proofs are perfectly indistinguishable from real proofs. As for pairing-product equations, zero-knowledge proofs are often possible but usually come at the expense of some overhead in comparison with NIWI proofs for the same equations: typically, the size of proofs may depend on the number of variables. In the paper, we only utilize NIZK proofs for multi-exponentiation equations.

In both cases, proofs for quadratic equations cost 9 group elements. Linear pairing-product equations (when $a_{ij} = 0$ for all i, j) take 3 group elements each. Linear multi-exponentiation equations of the type $\prod_{j=1}^{n} \mathcal{X}_j^{b_j} = T$ (resp. $\prod_{i=1}^{m} \mathcal{A}_i^{y_i} = T$) demand 3 (resp. 2) group elements.

3 Construction

Intuition. The group manager has a public key comprising elements $(\Omega = g^{\omega}, h_0, h_1, h_2)$ and uses $\omega \in \mathbb{Z}_p^*$ to generate membership certificates. These consist of 5 elements (K_1, K_2, K_3, K_4, y) and are reminiscent of users' private keys in the Boyen-Waters group signature [20]. Namely, K_1 is derived as $K_1 = (h_0 \cdot h_1^x \cdot h_2^y)^{1/(\omega + s_{\mathsf{ID}})}$, where s_{ID} is chosen by GM and identifies the user $\mathcal{U}$ while x is only known to $\mathcal{U}$ as his membership secret. The last element y is chosen by GM as part the tracing trapdoor for $\mathcal{U}$. The certificate also contains $K_3 = g^{s_{\mathsf{ID}}}$ and $K_4 = u_0^{s_{\mathsf{ID}}}$ as in [20]. Security proofs also require to include $K_2 = g^{1/(\omega + s_{\mathsf{ID}})}$ (so that, as in [18], ω and s_{ID} simultaneously appear more than once as denominators in the exponent).

To enable traceability when the appropriate tracing trapdoor is revealed (which is sometimes called "implicit tracing", as opposed to the "explicit tracing" that appeals to the signature opening algorithm), each signature must contain certain "traceability values" that make it possible to link the signature to its issuer. One of the technical points to address is to get these traceability values to interact with Groth-Sahai proof systems in a simple way. Indeed, at some step of the proof of anonymity, knowledge of the underlying values will have to be simulated in a zero-knowledge manner (*i.e.*, without knowing the actual witnesses). Previously used approaches to achieve implicit tracing using pairings (e.g., [23]) would require the traceability components to satisfy some pairing-product equation [36], for which zero-knowledge proofs usually come at some additional cost when they are at all possible. For this reason, as such traceability values, we rather let the signer include pieces of a linear tuple $(T_1, T_2, T_3) = (g^{x\delta_1}, g^{y\delta_2}, g^{\delta_1 + \delta_2})$ – which is a set of multi-exponentiation equations in the Groth-Sahai terminology – in each signature in such a way that the tracing trapdoor $(X = g^x, y)$ allows testing whether a signature stems from user $\mathcal{U}$ by checking if $e(T_1, g) = e(X, T_3/T_2^{1/y})$. Thanks to the use of multi-exponentiation equations, knowledge of the underlying scalars δ_1, δ_2 will be simulatable (in the WI setting) in a simple way in the proof of anonymity, which eventually relies on the sole DLIN assumption.

In traceability concerns, attention must be paid to the fact that users may be tempted to alter their membership certificate and modify the corresponding values x, y so as to defeat (implicit or explicit) tracing attempts. In the random oracle model, the problem is usually much easier in non-frameable pairing-based group signatures [43, 29, 26, 23], where membership certificates also have one component of the form $K_1 = (h_0 \cdot h_1^x \cdot h_2^y)^{1/(\omega + s_{\mathsf{ID}})}$. In those schemes, signatures prove knowledge of values $(K_1, x, y, s_{\mathsf{ID}})$ that satisfy the latter relation and, in the security proof, the forking lemma [44] allows extracting them in order to break some number theoretic assumption. In the present context, the problem is that committed exponents $x, y, s_{\mathsf{ID}} \in \mathbb{Z}_p$ are not fully extractable from Groth-Sahai commitments (typically, only $g^x, g^y, g^{s_{\mathsf{ID}}}$ are extractable) and we must settle for extracting a non-trivial information on them when it comes to prove traceability. To this end, we require each signature to contain redundancies in the form of (commitments to) quantities $h_1^x \cdot h_2^y$ and $h_3^x \cdot h_4^y$, for some group elements h_3 and h_4, which render certificate randomizations infeasible (as established by the proof against Type III forgeries in the security analysis against misidentification attacks). We remark that, in [33], Groth used a different method to build a non-frameable group signature using a certified signature scheme [12]. However, we cannot use the same technique since

the underlying certified signature only allows signing single group elements whereas we need to bind both $X \in \mathbb{G}$ and $y \in \mathbb{Z}_p$ to the membership certificate.

In [20], group members sign messages $\mathsf{m} \in \{0,1\}^n$ by randomly choosing $r \xleftarrow{\$} \mathbb{Z}_p$ and computing pairs $(\theta_1, \theta_2) = (u_0^{s_{\mathsf{ID}}} \cdot G_v(\mathsf{m})^r, g^r)$ using Waters' technique [45] and a suitable number theoretic hash function $G_v : \{0,1\}^n \to \mathbb{G}$ (termed "programmable" by Hofheinz and Kiltz [37]). In non-frameability concerns, we force signers to also use their membership secret x and generate pairs (θ_1, θ_2) somewhat in the fashion of the Waters-based multi-signature of Lu *et al.* [41]. Instead of signing a message m as $(\theta_1, \theta_2) = (u_0^{s_{\mathsf{ID}}} \cdot u_1^x \cdot G_v(m)^r, g^r)$, we need to generate such pairs as $(\theta_1, \theta_2) = (u_0^{s_{\mathsf{ID}}} \cdot u_1^{x\delta_1} \cdot G_v(m)^r, g^r)$ for the proof of non-frameability to work. Of course, u_1 and the set of group elements that implement the number theoretic hash function $G_v(.)$ are assumed to come from a trusted key generation procedure. In particular, the discrete logarithm $\log_g(u_1)$ must be held back from the group manager as, otherwise, a dishonest GM could frame honest users.

Signers are able to claim their signatures by proving knowledge of exponents $x, y \in \mathbb{Z}_p$ such that $T_3 = T_1^{1/x} \cdot T_2^{1/y}$. These proofs are also non-interactive and make use of a second Waters-like number theoretic hash function $G_f : \{0,1\}^n \to \mathbb{G}$, the parameters $(f_0, f_1, \ldots, f_n) \in_R \mathbb{G}^{n+1}$ of which must be generated by a trusted party (and not by the group manager as the latter could claim honest users' signatures if it were allowed to generate this reference string itself) when the scheme is set up. One difficulty is to prevent possibly dishonest group members from copying each other's claims. To this end, non-interactive claims are non-malleably bound to the long-term public key of the group member (as will be discussed hereafter, non-repudiation is enforced by having users register a public key upk in a PKI and use the private key usk to sign a piece of their membership certificate). In order to claim a signature containing traceability values $(T_1, T_2, T_3) = (g^{x\delta_1}, g^{y\delta_2}, g^{\delta_1+\delta_2})$ using their secret information $x, y \in \mathbb{Z}_p$, signers first compute a n-bit string $\mathsf{m}_c = \mathcal{H}(M||T_1||T_2||T_3||\mathsf{upk})$ and generate two pairs $(D_{x,1}, D_{x,2}) = (f^{1/x} \cdot G_f(\mathsf{m}_c)^{r_x}, T_1^{r_x})$ and $(D_{y,1}, D_{y,2}) = (f^{1/y} \cdot G_f(\mathsf{m}_c)^{r_y}, T_1^{r_y})$ using random $r_x, r_y \xleftarrow{\$} \mathbb{Z}_p$. Elements $(D_{x,1}, D_{x,2}, D_{y,1}, D_{y,2})$ are in turn signed using the long-term private key usk associated with upk.

In order to ensure non-repudiation, users have to register a long term public key upk in some PKI. In non-repudiation concerns, the underlying private key usk is used to sign (using an ordinary signature scheme) parts (X, K_1, K_2, K_3, y) of their membership certificate during the join protocol.

DESCRIPTION. In notations hereafter, it will be convenient to define the coordinate-wise pairing $E : \mathbb{G} \times \mathbb{G}^3 \to \mathbb{G}_T^3$ such that, for any element $h \in \mathbb{G}$ and any vector $\vec{g} = (g_1, g_2, g_3)$, we have $E\big(h, \vec{g}\big) = \big(e(h, g_1), e(h, g_2), e(h, g_3)\big)$. In addition, we also make use of a symmetric bilinear mapping $F : \mathbb{G}^3 \times \mathbb{G}^3 \to \mathbb{G}_T^9$ such that, for any two vectors $\vec{X} = (X_1, X_2, X_3) \in \mathbb{G}^3$ and $\vec{Y} = (Y_1, Y_2, Y_3) \in \mathbb{G}^3$, $F(\vec{X}, \vec{Y}) = \tilde{F}(\vec{X}, \vec{Y})^{1/2} \cdot \tilde{F}(\vec{Y}, \vec{X})^{1/2}$, where the non-commutative mapping $\tilde{F} : \mathbb{G}^3 \times \mathbb{G}^3 \to \mathbb{G}_T^9$ sends $(\vec{X}, \vec{Y})$ onto the matrix $\tilde{F}(\vec{X}, \vec{Y})$ of entry-wise pairings (*i.e.*, containing $e(X_i, Y_j)$ in its entry (i, j)).

Also, for any $z \in \mathbb{G}_T$, $\iota_T(z)$ denotes the 3×3 matrix containing z in position $(3,3)$ and $1_{\mathbb{G}_T}$ everywhere else. For $X \in \mathbb{G}$, the notation $\iota(X)$ will sometimes denote the vector $(1, 1, X) \in \mathbb{G}^3$.

Setup(λ, n)**:** for security parameters λ and $n \in \mathsf{poly}(\lambda)$, choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of order $p > 2^\lambda$, with $g, h_0, h_2, h_3, h_4, u_0, u_1 \xleftarrow{\$} \mathbb{G}$. Select $\gamma_1, \omega \xleftarrow{\$} \mathbb{Z}_p^*$ and set $h_1 = g^{\gamma_1}$, $\Omega = g^\omega$. Select $\overline{v} = (v_0, v_1, \ldots, v_n) \xleftarrow{\$} \mathbb{G}^{n+1}$. Choose vectors $\mathbf{g} = (\vec{g_1}, \vec{g_2}, \vec{g_3})$ such that $\vec{g_1} = (g_1, 1, g) \in \mathbb{G}^3$, $\vec{g_2} = (1, g_2, g) \in \mathbb{G}^3$, and $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$, with $g_1 = g^{\alpha_1}, g_2 = g^{\alpha_2}$ and $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_p^*$, $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$. It also chooses $\overline{f} = (f_0, f_1, \ldots, f_n) \xleftarrow{\$} \mathbb{G}^{n+1}$ and $f \xleftarrow{\$} \mathbb{G}$. The algorithm also specifies a hash function $\mathcal{H} : \{0,1\}^* \to \{0,1\}^n$ from a collision-resistant family. The group public key is defined

to be

$$\mathcal{Y} := \Big(g,\ h_0,\ h_1 = g^{\gamma_1},\ h_2,\ h_3,\ h_4,\ \Omega = g^{\omega},\ u_0,\ u_1,\ \overline{v},\ \overline{f},\ \mathbf{g},\ \mathcal{H}\Big)$$

while the private key $\mathcal{S} := \big(\gamma_1, \omega, \alpha_1, \alpha_2\big)$ is given to the group manager.

Join$^{(\mathrm{GM},\mathcal{U}_i)}$ **:** the prospective group member $\mathcal{U}_i$ and the group manager GM run an interactive protocol whereby the user obtains a membership certificate cert_i and a membership secret sec_i. The protocol is the following:

1. User $\mathcal{U}_i$ and the GM execute an interactive protocol (such as Groth's protocol [33, Section 4.1] recalled in appendix B) allowing them to jointly generate $X = g^x$ so that $x \in \mathbb{Z}_p$ is randomly distributed and known only to the user while GM learns the corresponding public value X.
2. GM first computes $h_1^x = X^{\gamma_1}$ and then uses it to compute $K_1 = (h_0 \cdot h_1^x \cdot h_2^y)^{1/(\omega+s_{\mathsf{ID}})}$, $K_2 = g^{1/(\omega+s_{\mathsf{ID}})}$, $K_3 = g^{s_{\mathsf{ID}}}$ and $K_4 = u_0^{s_{\mathsf{ID}}}$, for newly chosen random values $s_{\mathsf{ID}}, y \xleftarrow{\$} \mathbb{Z}_p^*$. Elements K_1, K_2, K_3 and y are sent to the user.
3. $\mathcal{U}_i$ checks that received elements (K_1, K_2, K_3, y) satisfy

$$e(K_1, \Omega \cdot K_3) = e(h_0, g) \cdot e(h_1, X) \cdot e(h_2, g)^y,$$
$$e(K_2, \Omega \cdot K_3) = e(g, g).$$

   If so, he generates a signature $sig_i = \mathrm{Sign}_{\mathsf{usk}[i]}\big(X||K_1||K_2||K_3||g^y\big)$ and sends it back to GM.
4. If $\mathrm{Verify}_{\mathsf{upk}[i]}\big(X||K_1||K_2||K_3||g^y, sig_i\big) = 1$, GM sends $K_4 = u_0^{s_{\mathsf{ID}}}$ to $\mathcal{U}_i$ and stores the record $\mathsf{transcript}_i := (X, K_1, K_2, K_3, K_4, y, sig_i)$ in its database $\mathsf{transcripts}$. User $\mathcal{U}_i$ checks that $e(K_3, u_0) = e(g, K_4)$. If so, he sets his membership certificate as $\mathsf{cert}_i := (K_1, K_2, K_3, K_4, y)$ and his membership secret as $\mathsf{sec}_i := x$.

Sign$(M, \mathcal{Y}, \mathsf{cert}_i, \mathsf{sec}_i)$**:** to sign M, user $\mathcal{U}_i$ parses cert_i as (K_1, K_2, K_3, K_4, y) and sec_i as $x \in \mathbb{Z}_p^*$ and conducts the following steps.

1. Choose $\delta_1, \delta_2 \xleftarrow{\$} \mathbb{Z}_p^*$ and compute the traceability values

$$T_1 = g^{x\delta_1} \qquad T_2 = g^{y\delta_2} \qquad T_3 = g^{\delta_1+\delta_2}$$

2. Set $G_v(\mathsf{m}) = v_0 \cdot \prod_{j=1}^{n} v_j^{m_j}$ with $\mathsf{m} = m_1 \ldots m_n = \mathcal{H}(M||T_1||T_2||T_3)$.
3. Pick $r_s \xleftarrow{\$} \mathbb{Z}_p^*$ and compute

$$\begin{aligned}
\theta_1 &= K_1 = (h_0 \cdot h_1^x \cdot h_2^y)^{1/(\omega+s_{\mathsf{ID}})} & \theta_5 &= g^{r_s}\\
\theta_2 &= K_2 = g^{1/(\omega+s_{\mathsf{ID}})} & \theta_6 &= h_1^x \cdot h_2^y\\
\theta_3 &= K_3 = g^{s_{\mathsf{ID}}} & \theta_7 &= h_3^x \cdot h_4^y\\
\theta_4 &= K_4 \cdot u_1^{x\delta_1} \cdot G_v(\mathsf{m})^{r_s} & \theta_8 &= g^x\\
&= u_0^{s_{\mathsf{ID}}} \cdot u_1^{x\delta_1} \cdot G_v(\mathsf{m})^{r_s} & \theta_9 &= g^y
\end{aligned}$$

   so that

$$e(\theta_1, \Omega \cdot \theta_3) = e(h_0, g) \cdot e(\theta_6, g) \quad (1)$$
$$e(\theta_2, \Omega \cdot \theta_3) = e(g, g) \quad (2)$$
$$e(\theta_4, g) = e(u_0, \theta_3) \cdot e(u_1, T_1) \cdot e(G_v(\mathsf{m}), \theta_5). \quad (3)$$
$$e(\theta_6, g) = e(h_1, \theta_8) \cdot e(h_2, \theta_9) \quad (4)$$
$$e(\theta_7, g) = e(h_3, \theta_8) \cdot e(h_4, \theta_9) \quad (5)$$

4. Commit to variables θ_i, for $i = 1, \ldots, 9$. That is, for $i = 1, \ldots, 9$, randomly choose $r_i, s_i, t_i \xleftarrow{\$} \mathbb{Z}_p$ and set $\vec{\sigma_i} = (1, 1, \theta_i) \cdot \vec{g_1}^{r_i} \cdot \vec{g_2}^{s_i} \cdot \vec{g_3}^{t_i}$. Then, commit to exponents $\delta_1, \delta_2 \in \mathbb{Z}_p$ by choosing $r_{10}, s_{10}, r_{11}, s_{11} \xleftarrow{\$} \mathbb{Z}_p^*$ and setting $\vec{\sigma_{10}} = \vec{\varphi}^{\delta_1} \cdot \vec{g_1}^{r_{10}} \cdot \vec{g_2}^{s_{10}}$, $\vec{\sigma_{11}} = \vec{\varphi}^{\delta_2} \cdot \vec{g_1}^{r_{11}} \cdot \vec{g_2}^{s_{11}}$, where $\vec{\varphi} = \vec{g_3} \odot (1, 1, g)$.
5. Give proofs that committed variables $\theta_1, \ldots, \theta_9$ satisfy (1)-(5) and that $\vec{\sigma_{10}}$, $\vec{\sigma_{11}}$ are commitment to values δ_1, δ_2 satisfying

$$T_1 = \theta_8^{\delta_1} \qquad T_2 = \theta_9^{\delta_2} \qquad T_3 = g^{\delta_1 + \delta_2} \tag{6}$$

   a. Relations (1)-(2) are quadratic pairing-product equations (in the Groth-Sahai terminology [36]) over variables $\theta_1, \theta_2, \theta_3, \theta_6$. Each relation requires a proof consisting of 9 group elements. Let us call these proofs $\pi_1 = (\vec{\pi}_{1,1}, \vec{\pi}_{1,2}, \vec{\pi}_{1,3})$, $\pi_2 = (\vec{\pi}_{2,1}, \vec{\pi}_{2,2}, \vec{\pi}_{2,3})$. Relations (6) are multi-exponentiation equations. The first two ones are quadratic and the corresponding proofs $\pi_6 = (\vec{\pi}_{6,1}, \vec{\pi}_{6,2}, \vec{\pi}_{6,3})$ and $\pi_7 = (\vec{\pi}_{7,1}, \vec{\pi}_{7,2}, \vec{\pi}_{7,3})$ both consist of 3 vectors of $\mathbb{G}^3$. The third relation of (6) is a linear multi-exponentiation equation and the proof $\pi_8 = (\pi_{8,1}, \pi_{8,2})$ is just 2 group elements.
   b. Relations (3)-(5) are linear pairing-product equations over variables $\theta_3, \ldots, \theta_9$. Corresponding proofs cost 3 group elements each and π_3, π_4, π_5 are all vectors of $\mathbb{G}^3$.

   For clarity, we abstract away the construction of these proofs from the present description and refer to appendix C for details on how proof elements are calculated.

   The signature finally consists of $\sigma = (T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$.

Verify$(M, \sigma, \mathcal{Y})$**:** parse the signature σ as a tuple $(T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$. Then, compute $\mathsf{m} = m_1 \ldots m_n = \mathcal{H}(M||T_1||T_2||T_3)$ and $G_v(\mathsf{m}) = v_0 \cdot \prod_{j=1}^{n} v_j^{m_j}$. Verifying $\pi_1, \ldots, \pi_8$ entails to check whether the following equations (some of which bear resemblance with relations (1)-(5)), where $\vec{\varphi} = \vec{g_3} \odot (1, 1, g)$, are all satisfied. The verifier returns 1 if they are and 0 otherwise.

$$\begin{aligned}
&1)\ F\big(\vec{\sigma_1}, \iota(\Omega) \odot \vec{\sigma_3}\big) = \iota_T\big(e(h_0, g)\big) \odot F\big(\vec{\sigma_6}, \iota(g)\big) \\
&\qquad\qquad \odot F\big(\vec{g_1}, \vec{\pi}_{1,1}\big) \odot F\big(\vec{g_2}, \vec{\pi}_{1,2}\big) \odot F\big(\vec{g_3}, \vec{\pi}_{1,3}\big) \\
&2)\ F\big(\vec{\sigma_2}, \iota(\Omega) \cdot \vec{\sigma_3}\big) = \iota_T\big(e(g, g)\big) \odot F\big(\vec{g_1}, \vec{\pi}_{2,1}\big) \odot F\big(\vec{g_2}, \vec{\pi}_{2,2}\big) \odot F\big(\vec{g_3}, \vec{\pi}_{2,3}\big) \\
&3)\ E\big(g, \vec{\sigma_4}\big) = E\big(u_0, \vec{\sigma_3}\big) \odot E\big(u_1, \iota(T_1)\big) \odot E\big(G_v(\mathsf{m}), \vec{\sigma_5}\big) \\
&\qquad\qquad \odot E\big(\pi_{3,1}, \vec{g_1}\big) \odot E\big(\pi_{3,2}, \vec{g_2}\big) \odot E\big(\pi_{3,3}, \vec{g_3}\big) \\
&4)\ E\big(g, \vec{\sigma_6}\big) = E\big(h_1, \vec{\sigma_8}\big) \odot E\big(h_2, \vec{\sigma_9}\big) \odot E\big(\pi_{4,1}, \vec{g_1}\big) \odot E\big(\pi_{4,2}, \vec{g_2}\big) \odot E\big(\pi_{4,3}, \vec{g_3}\big) \\
&5)\ E\big(g, \vec{\sigma_7}\big) = E\big(h_3, \vec{\sigma_8}\big) \odot E\big(h_4, \vec{\sigma_9}\big) \odot E\big(\pi_{5,1}, \vec{g_1}\big) \odot E\big(\pi_{5,2}, \vec{g_2}\big) \odot E\big(\pi_{5,3}, \vec{g_3}\big) \\
&6)\ F\big(\vec{\sigma_8}, \vec{\sigma_{10}}\big) = F\big(\iota(T_1), \vec{\varphi}\big) \odot F\big(\vec{g_1}, \vec{\pi}_{6,1}\big) \odot F\big(\vec{g_2}, \vec{\pi}_{6,2}\big) \odot F\big(\vec{g_3}, \vec{\pi}_{6,3}\big) \\
&7)\ F\big(\vec{\sigma_9}, \vec{\sigma_{11}}\big) = F\big(\iota(T_2), \vec{\varphi}\big) \odot F\big(\vec{g_1}, \vec{\pi}_{7,1}\big) \odot F\big(\vec{g_2}, \vec{\pi}_{7,2}\big) \odot F\big(\vec{g_3}, \vec{\pi}_{7,3}\big) \\
&8)\ E\big(g, \vec{\sigma_{10}} \odot \vec{\sigma_{11}}\big) = E(T_3, \vec{\varphi}) \odot E\big(\pi_{8,1}, \vec{g_1}\big) \odot E\big(\pi_{8,2}, \vec{g_2}\big)
\end{aligned}$$

Open$(\sigma, \mathcal{Y}, \mathcal{S})$**:** parse the signature σ as $(T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$ and the private key $\mathcal{S}$ as $\{\gamma_1, \omega, \alpha_1, \alpha_2\}$. For each $i \in \{3, 8, 9\}$, parse element $\vec{\sigma_i}$ as a BBS ciphertext $(\sigma_{i,1}, \sigma_{i,2}, \sigma_{i,3}) \in \mathbb{G}^3$ and compute $\theta_i = \sigma_{i,3} \cdot {\sigma_{i,1}}^{-1/\alpha_1} \cdot {\sigma_{i,2}}^{-1/\alpha_2}$. Check whether the database $\mathsf{transcripts}$ contains a record $\mathsf{transcript}_i = (X, K_1, K_2, K_3, K_4, y, sig_i)$ such that $\theta_3 = K_3$, $\theta_8 = X$ and $\theta_9 = g^y$. If yes, return i as the signer's index. Otherwise, return $\perp$.

Reveal$(i, \mathsf{transcripts})$**:** to reveal the tracing trapdoor for user $\mathcal{U}_i$, scan the database $\mathsf{transcripts}$ to find $\mathsf{transcript}_i = (X, K_1, K_2, K_3, K_4, y, sig_i)$ and output $\mathsf{trace}_i := (X, y)$.

Trace$(\sigma, \mathsf{trace}_i, \mathcal{Y})$**:** parse σ as $(T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$ and the tracing trapdoor trace_i as $(X, y) \in \mathbb{G} \times \mathbb{Z}_p^*$. Return 1 if $e(T_3/T_2^{1/y}, X) = e(g, T_1)$ and 0 otherwise.

Claim$(M, \sigma, \mathsf{sec}_i, \mathcal{Y})$**:** given $\sigma = (T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$, $\mathsf{sec}_i = x$ and part y of his membership certificate cert_i, the signer i parses his long term key pair as $(\mathsf{usk}[i], \mathsf{upk}[i])$ and computes $\mathsf{m}_c = \mathcal{H}(M||T_1||T_2||T_3||\mathsf{upk}[i]) = m_{c,1} \ldots m_{c,n} \in \{0,1\}^n$ as well as $G_f(\mathsf{m}_c) = f_0 \cdot \prod_{j=1}^n f_j^{m_{c,j}}$. Then, he picks $r_x, r_y \xleftarrow{\$} \mathbb{Z}_p$ and computes

$$(D_{x,1}, D_{x,2}) = \big(f^{1/x} \cdot G_f(\mathsf{m}_c)^{r_x},\ T_1^{r_x}\big), \qquad (D_{y,1}, D_{y,2}) = \big(f^{1/y} \cdot G_f(\mathsf{m}_c)^{r_y},\ T_2^{r_y}\big)$$

The non-interactive claim consists of the tuple $\tau := (D_{x,1}, D_{x,2}, D_{y,1}, D_{y,2}, csig_i)$, the last part of which is a digital signature $csig_i = \mathrm{Sign}_{\mathsf{usk}[i]}(D_{x,1}||D_{x,2}||D_{y,1}||D_{y,2})$.

Claim-Verify$(M, \sigma, \tau, \mathsf{upk}, \mathcal{Y})$**:** given a traceable signature $\sigma = (T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$, to verify the claim $\tau = (D_{x,1}, D_{x,2}, D_{y,1}, D_{y,2}, csig_i)$, compute $G_f(\mathsf{m}_c) = f_0 \cdot \prod_{j=1}^n f_j^{m_{c,j}}$, where $\mathsf{m}_c = \mathcal{H}(M||T_1||T_2||T_3||\mathsf{upk}[i]) = m_{c,1} \ldots m_{c,n} \in \{0,1\}^n$, and return 1 iff

$$e(f, T_3) = \frac{e(D_{x,1}, T_1) \cdot e(D_{y,1}, T_2)}{e(G_f(\mathsf{m}_c), D_{x,2} \cdot D_{y,2})} \tag{7}$$

and $\mathrm{Verify}_{\mathsf{upk}[i]}\big((D_{x,1}||D_{x,2}||D_{y,1}||D_{y,2}), csig_i\big) = 1$.

Correctness of the Claiming Algorithm. We easily verify that honestly generated claims $(D_{x,1}, D_{x,2}, D_{y,1}, D_{y,2})$ are always accepted since they satisfy the relations

$$e(D_{x,1}, T_1) = e(f, g^{\delta_1}) \cdot e(G_f(\mathsf{m}_c), D_{x,2})$$
$$e(D_{y,1}, T_2) = e(f, g^{\delta_2}) \cdot e(G_f(\mathsf{m}_c), D_{y,2}),$$

so that

$$\frac{e(D_{x,1}, T_1)}{e(G_f(\mathsf{m}_c), D_{x,2})} \cdot \frac{e(D_{y,1}, T_2)}{e(G_f(\mathsf{m}_c), D_{y,2})} = e\big(f, g^{\delta_1+\delta_2}\big).$$

Including the signer's long-term public key $\mathsf{upk}[i]$ among the inputs of the hash function $\mathcal{H}$ in the computation of m_c prevents other dishonest groups members from tampering with user i's claim $\tau = (D_{x,1}, D_{x,2}, D_{y,1}, D_{y,2}, csig_i)$ by replacing $csig_i$ with their own signature on the claiming information $(D_{x,1}||D_{x,2}||D_{y,1}||D_{y,2})$. As we shall see in the proof of security against framing attacks, dishonest group members will be unable to successfully claim an honest signer's signature as long as the Diffie-Hellman assumption holds.

Discussion. The opening algorithm performs BBS decryptions on ciphertexts $\vec{\sigma_3}$, $\vec{\sigma_8}$ and $\vec{\sigma_9}$. Theoretically, decrypting only $\vec{\sigma_3}$ suffices (since s_{ID} must be unique in the database $\mathsf{transcripts}$). However, also decrypting $\vec{\sigma_8}$ and $\vec{\sigma_9}$ simplifies the proofs of security against misidentification attacks and framing attacks. In the former for instance, a failure of the implicit tracing mechanism implies a failure of the opening algorithm and reduces the number of cases to consider.

We note that the claiming algorithm does not make use of pairing-based non-interactive witness indistinguishable proofs. However, such techniques can be adapted to work in this context as well. Indeed, we can alternatively build on Groth's techniques for constructing simulation-sound NIZK proofs [32][Section 6] and have the claimer generate a simulation-sound extractable (see [32] for

definitions) proof that he knows x, y such that $T_3 = T_1^{1/x} \cdot T_2^{1/y}$ *and* the private key associated with his long-term public key. Since simulation-sound extractable proofs are also non-malleable, the adversary cannot break the AND link between the two statements and re-use the proof of knowledge of x, y such that $T_3 = T_1^{1/x} \cdot T_2^{1/y}$.

In comparison with the latter technique, the advantage of our approach is to provide a better efficiency as claims only consist of four group elements and an ordinary signature. In addition, the signer's key pair $(\mathsf{usk}[i], \mathsf{upk}[i])$ can be a public key for any (not necessarily pairing-based) digital signature scheme.

Efficiency. From an efficiency standpoint, each signature consists of 83 group elements. Using a symmetric pairing configuration with 256-bit prime order groups, we obtain signatures of 2.593 kB.

Signing requires a few tens of exponentiations. While a number of pairing evaluations seem necessary to verify at first glance, probabilistic batch verification techniques (as exemplified in [11]) allow for dramatic improvements (at the expense of a small probability of wrongly accepting an invalid signature) w.r.t. naive implementations where each pairing is calculated individually. When suitably processed altogether, verification equations 3-5 and 8 require to compute a product of no more than 9 pairings and a few multi-exponentiations. Verification equations 1-2 and 6-7 can be handled by first translating them into a randomized product of several bilinear maps of the type $F(\cdot, \cdot)$. The structure of matrices $F(\cdot, \cdot)$ then makes it possible the decrease the overall verification cost of conditions 1-2 and 6-7 to the equivalent of a product of 15 pairings and some multi-exponentiations.

4 Security

We establish the security of the scheme in the standard model under the assumptions of section 2.1 and the assumption that the digital signature scheme in use is existentially unforgeable under chosen-message attacks (as defined in [31] and recalled in appendix A).

4.1 Security against Misidentification Attacks

Theorem 1 (Misidentification). *The scheme is secure against misidentification attacks assuming that the ℓ-HSDH problem, where ℓ is the total number of $Q_{\mathsf{a\text{-}join}}$ and $Q_{\mathsf{p\text{-}join}}$-queries, and the 1-mTDH problem are both hard in $\mathbb{G}$.*

Proof. To win the misidentification game, the adversary must output a non-trivial signature for which the opening algorithm or the implicit tracing algorithm fail to point to an adversarially-controlled group member.

Let $\sigma^\star = (T_1^\star, T_2^\star, T_3^\star, \vec{\sigma_1}^\star, \ldots, \vec{\sigma_{11}}^\star, \pi_1^\star, \ldots, \pi_8^\star)$ denote the adversary's forgery and let us first assume that $\mathsf{Open}(\sigma^\star, \mathcal{Y}, \mathcal{S}) \notin U^a$. We distinguish three cases:

- **Type I forgeries** are those for which the BBS decryption $\theta_3^\star = g^{s_{\mathsf{ID}}}$ of $\vec{\sigma_3}^\star$ does not appear anywhere in $\mathsf{transcripts}$. We distinguish Type I-A forgeries, where the underlying $\theta_3^\star = g^{s_{\mathsf{ID}}}$ never appears at any time during the game, from Type I-B forgeries for which $\theta_3^\star$ does not correspond to any record of $\mathsf{transcripts}$ but did appear (implicitly, as part of K_3) in a join protocol (triggered by a $Q_{\mathsf{a\text{-}join}}$ query) that aborted before reaching its last step.

- **Type II forgeries** are such that $\vec{\sigma_3}^\star$ decrypts to a value $\theta_3^\star = g^{s_{\mathsf{ID}}}$ that was assigned to some honest user $i \in U^p$ (initialized via a $Q_{\mathsf{p\text{-}join}}$-query). Such forgeries thus include those for which the opening algorithm points to some user $i \in U^p$ who did not sign the corresponding message.
- **Type III forgeries** open in such a way that $\vec{\sigma_3}^\star$ decrypts to the $\theta_3^\star$-value of an adversarially-controlled user in $\mathsf{transcripts}$ but $(\vec{\sigma_8}^\star, \vec{\sigma_9}^\star)$ does not. These forgeries include those that would defeat the implicit tracing algorithm.

Lemmas 1, 2 and 3 show that, if the adversary could produce either of such forgeries, it would be possible to break the HSDH or the 1-mTDH assumption.

Finally, one can readily check that an adversary cannot come up with a fake signature defeating the implicit tracing algorithm without being one of the above kinds of forgeries. Indeed, let $\sigma^\star$ be such a forgery and let us consider the decryption $\theta_3^\star$ of $\vec{\sigma_3}^\star$. If it differs from any K_3 appearing in $\mathsf{transcripts}$, $\sigma^\star$ is actually a Type I forgery. If $\theta_3^\star$ matches K_3 in $\mathsf{transcript}_i$ for some $i \in U^p$, we have a Type II forgery. We are left with the case where $\theta_3^\star$ matches K_3 in $\mathsf{transcript}_i$ for some $i \in U^a$. Here, a failure of the implicit tracing necessarily means that $\mathcal{A}$, acting as a cheating group member, was able to twist his membership certificate so as to keep the same s_{ID} and alter the membership secret x or the "traceability component" y. We thus have a Type III forgery. □

The security against Type I and Type II attacks can be established almost in the same way as traceability attacks are handled in the security proof of the Boyen-Waters group signature [20] and the proofs of lemmas 1 and 2 are available in appendix D.

Lemma 1. *The advantage of any Type I forger $\mathcal{A}$ is bounded by*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{mis-id-I}}(\lambda) \leq 2 \cdot \ell_a \cdot \mathbf{Adv}^{(\ell_a+\ell_p)\text{-HSDH}}(\lambda)$$

where ℓ_a and ℓ_p denote the number of $Q_{\mathsf{a\text{-}join}}$ and $Q_{\mathsf{p\text{-}join}}$-queries respectively.

Proof. Given in appendix D.1. □

Lemma 2. *The scheme is secure against Type II forgeries under the HSDH assumption. The advantage of any Type II adversary $\mathcal{A}$ is at most*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{mis-id-II}}(\lambda, n) \leq 4 \cdot n \cdot \ell_s \cdot \big(1 - \frac{\ell_a}{p}\big)^{-1} \cdot \mathbf{Adv}^{\ell_a\text{-HSDH}}(\lambda)$$

where ℓ_a and ℓ_s stand for the number of $Q_{\mathsf{a\text{-}join}}$ and Q_{sig}-queries.

Proof. Detailed in appendix D.2. □

Type III forgeries are somewhat trickier to deal with. Indeed, the proof of lemma 3 is the most difficult part of the proof of security against misidentification attacks and it appeals to the non-standard 1-mTDH assumption. We leave it as an interesting problem to hedge against misidentification attacks using only the HSDH assumption.

Lemma 3. *The advantage of any Type III adversary $\mathcal{A}$ is bounded by*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{mis-id-III}}(\lambda, n) \leq \ell_a \cdot \big(1 - \frac{1}{p}\big)^{-1} \cdot \mathbf{Adv}^{\text{1-mTDH}}(\lambda)$$

where ℓ_a is the number of $Q_{\mathsf{a\text{-}join}}$-queries.

Proof. In a successful Type III forgery $\sigma^\star$, by assumption, the opening algorithm decrypts $\vec{\sigma_3}^\star$ to a value $\theta_3^\star = \sigma_{3,3}^\star \cdot {\sigma_{3,1}^\star}^{-1/\alpha_1} \cdot {\sigma_{3,2}^\star}^{-1/\alpha_2}$ that equals some K_3 appearing in the transcript of a user in U^a whereas the BBS decryption of $(\vec{\sigma_8}^\star, \vec{\sigma_9}^\star)$ does not match the values (X, y) that were assigned to that user.

The simulator $\mathcal{B}$ receives as input a modified 1-Triple Diffie-Hellman instance consisting of $(g, A = g^a, B = g^b) \in \mathbb{G}^3$ and a single pair $(C = g^{1/(a+c)}, c) \in \mathbb{G} \times \mathbb{Z}_p^*$. To prepare the public key $\mathcal{Y}$, it picks $\omega, \rho_{u,0}, \rho_{u,1}, \beta_0, \ldots, \beta_n \xleftarrow{\$} \mathbb{Z}_p^*$. It sets $\Omega = g^\omega$, $v_i = g^{\beta_i}$, for $i = 0, \ldots, n$, and $u_i = g^{\rho_{u,i}}$ for $i = 0, 1$. Then, it draws new random values $\rho, \gamma_0, \gamma_1, \gamma_2, \gamma_3, \gamma_4, x^\star, y^\star \xleftarrow{\$} \mathbb{Z}_p^*$ and defines $h_1 = g^\rho \cdot B^{\gamma_1}$, $h_2 = g^\rho \cdot B^{\gamma_2}$, $h_3 = g^{\gamma_3} \cdot A^\rho$, $h_4 = g^{\gamma_4} \cdot A^\rho$ and $h_0 = g^{\gamma_0} \cdot h_1^{-x^\star} \cdot h_2^{-y^\star}$. It finally chooses vector sets $\mathbf{g}$, $\mathbf{f}$ to have perfectly sound proof systems.
The group public key is

$$\mathcal{Y} := \Big(g,\ \{h_i\}_{i=0,\ldots,4},\ \Omega,\ u_0,\ u_1,\ \{v_i\}_{i=0,\ldots,n},\ \mathbf{g},\ \mathbf{f}\Big).$$

At the outset of the simulation, $\mathcal{B}$ draws an index $i^\star \xleftarrow{\$} \{1, \ldots, \ell_a\}$ and initializes variables $ctr_a, ctr'_a, ctr_p \leftarrow 0$.

- $Q_{\mathsf{a\text{-}join}}$-queries: $\mathcal{B}$ increments ctr'_a and considers the following two cases.
  - If $ctr'_a \neq i^\star$, $\mathcal{B}$ acts as the group manager as specified by the protocol (recall that it knows ω and can always properly generate certificates).
  - If $ctr'_a = i^\star$, $\mathcal{B}$ simulates $\mathcal{A}$'s view in the first step of the join protocol to force $\mathcal{A}$'s membership secret to be $x^\star$ (so that the public value is $X = g^{x^\star}$). The simulation implicity defines $s_{\mathsf{ID}_{i^\star}} = \frac{1}{a+c} - \omega$ (and thus $1/(s_{\mathsf{ID}_{i^\star}} + \omega) = a + c$) by setting

$$\begin{aligned} K_1 &= (h_0 \cdot h_1^{x^\star} \cdot h_2^{y^\star})^{\frac{1}{\omega + s_{i^\star}}} = (A \cdot g^c)^{\gamma_0} \\ K_2 &= g^{\frac{1}{\omega + s_{i^\star}}} = A \cdot g^c \\ K_3 &= g^{s_{i^\star}} = g^{1/(a+c)} \cdot g^{-\omega} = C \cdot g^{-\omega} \\ K_4 &= u_0^{s_{i^\star}} = (C \cdot g^{-\omega})^{\rho_{u,0}} \end{aligned}$$

    In step 2, $\mathcal{B}$ first sends $K_1, K_2, K_3, y^\star$ to $\mathcal{A}$ and aborts if he fails to send back a valid signature on $X||K_1||K_2||K_3||g^{y^\star}$. If $\mathcal{A}$ correctly answers, $\mathcal{B}$ hands him K_4, increments ctr_a and stores a record $(\mathsf{N}, \mathsf{transcripts}_{\mathsf{N}})$, with $\mathsf{N} = ctr_a + ctr_p$ in $\mathsf{transcripts}$.
- $Q_{\mathsf{p\text{-}join}}$-queries and Q_{sig}-queries: to answer $Q_{\mathsf{p\text{-}join}}$-queries, $\mathcal{B}$ follows the join protocol using the group secret key $\mathcal{S} := (\gamma_1, \omega, p)$ and increments ctr_p. It can also perfectly answer signing queries on behalf of honest user since it knows their membership certificates and secrets.
- $Q_{\mathcal{Y}}$ and $Q_{\mathsf{reveal}}(i)$-queries: can be handled according to the specification of the scheme since $\mathcal{B}$ always knows the values requested by $\mathcal{A}$.
- Q_{sig}-queries: always involve users in U^p and $\mathcal{B}$ thus always knows private elements that it needs to answer the query.

Eventually, the adversary $\mathcal{A}$ comes up with a message $M^\star$ along with a valid traceable signature $\sigma^\star = (T_1^\star, T_2^\star, T_3^\star, \vec{\sigma_1}^\star, \ldots, \vec{\sigma_{11}}^\star, \pi_1^\star, \ldots, \pi_8^\star)$ that must be a type III forgery. At this stage, $\mathcal{B}$ fails if the decryption of $\vec{\sigma_3}^\star$ differs from the element $K_3 = C \cdot g^{-\omega}$ that $\mathcal{B}$ calculated at the $i^{\star\text{th}}$ $Q_{\mathsf{a\text{-}join}}$-query (as it guessed the wrong $i^\star$).

Otherwise, for all $i \in \{1,\ldots,9\}\backslash\{3\}$, it decrypts other $\vec{\sigma_i}^\star$ into $\theta_i^\star$. as in the proofs of lemmas 1 and 2. Since the proof system is configured for the perfect soundness setting, it comes that

$$\begin{aligned}
\theta_1^\star &= (h_0 \cdot h_1^{x'} \cdot h_2^{y'})^{\frac{1}{\omega + s_{\mathsf{ID}_{i^\star}}}} & \theta_6^\star &= h_1^{x'} \cdot h_2^{y'} \\
&= \left(g^{\gamma_0} \cdot h_1^{(x'-x^\star)} \cdot h_2^{y'-y^\star}\right)^{a+c} & \theta_7^\star &= h_3^{x'} \cdot h_4^{y'} \\
\theta_8^\star &= g^{x'} & \theta_9^\star &= g^{y'}
\end{aligned}$$

for some $x', y' \in \mathbb{Z}_p^*$ that $\mathcal{B}$ does not know. However, if we set $\Delta x = x' - x^\star$ and $\Delta y = y' - y^\star$, $\mathcal{B}$ can compute

$$\begin{aligned}
Z_1 &= \theta_1^\star / K_1 = \left(h_1^{\Delta x} \cdot h_2^{\Delta y}\right)^{a+c} = \left(g^{\rho(\Delta x + \Delta y)} \cdot B^{\gamma_1 \Delta x + \gamma_2 \Delta y}\right)^{a+c} \\
Z_2 &= \theta_7^\star / (h_3^{x^\star} \cdot h_4^{y^\star}) = h_3^{\Delta x} \cdot h_4^{\Delta y} = g^{\gamma_3 \Delta x + \gamma_4 \Delta y} \cdot A^{\rho(\Delta x + \Delta y)} \\
Z_3 &= \theta_8^\star / g^{x^\star} = g^{\Delta x} \\
Z_4 &= \theta_9^\star / g^{y^\star} = g^{\Delta y} \\
Z_5 &= \theta_6^\star / (h_1^{x^\star} \cdot h_2^{y^\star}) = h_1^{\Delta x} \cdot h_2^{\Delta y} = g^{\rho(\Delta x + \Delta y)} \cdot B^{\gamma_1 \Delta x + \gamma_2 \Delta y}
\end{aligned}$$

which in turn reveal

$$\begin{aligned}
Z_6 &= (A \cdot g^c)^{\rho(\Delta x + \Delta y)} = \left(Z_2 / (Z_3^{\gamma_3} \cdot Z_4^{\gamma_4})\right) \cdot (Z_3 \cdot Z_4)^{\rho c} \\
Z_7 &= B^{\gamma_1 \Delta x + \gamma_2 \Delta y} = Z_5 \cdot (Z_3 \cdot Z_4)^{\rho}
\end{aligned}$$

and finally

$$Z_8 = g^{ab(\gamma_1 \Delta x + \gamma_2 \Delta y)} = B^{a(\gamma_1 \Delta x + \gamma_2 \Delta y)} = Z_1 / (Z_6 \cdot Z_7^c),$$

so that, if we implicitly define $\mu = \gamma_1 \Delta x + \gamma_2 \Delta y$, $\mathcal{B}$ has eventually found a triple

$$(g^\mu, g^{b\mu}, g^{ab\mu}) = (Z_3^{\gamma_1} \cdot Z_4^{\gamma_2}, Z_7, Z_8).$$

Since γ_1 and γ_2 are both random and perfectly hidden from $\mathcal{A}$'s view, we have $g^\mu \neq 1_{\mathbb{G}}$ (and thus $\gamma_1 \Delta x + \gamma_2 \Delta y \neq 0 \bmod p$) with overwhelming probability (greater than $1 - 1/p$) and the triple $(g^\mu, g^{b\mu}, g^{ab\mu})$ is non-trivial. We easily check that, if $\mathcal{A}$ is successful, so is $\mathcal{B}$ as long as it correctly guesses $i^\star \in \{1,\ldots,\ell_a\}$. □

4.2 Security against Framing Attacks

Establishing the security of the scheme against framing attacks also entails to separately consider several kinds of forgeries.

Theorem 2 (Non-frameability). *The scheme is secure against framing attacks assuming that: (i) the 1-mTDH assumption holds in $\mathbb{G}$; (ii) the underlying digital signature is existentially unforgeable under chosen-message attacks; (iii) $\mathcal{H}$ is a collision-resistant hash function.*

Proof. As required by the model, we consider two kinds of frameability attacks.

- **Type I attacks**: the adversary outputs a pair $(M^\star, \sigma^\star)$ that opens or traces to signer $i \in U^b$ whereas i did not produce a pair (M, σ) such that $(M, \sigma) \equiv_s (M^\star, \sigma^\star)$. We further distinguish Type I-A attacks, where no signer $i \in U^b$ produced (M, σ) with $(M, \sigma) \equiv_s (M^\star, \sigma^\star)$, from Type I-B attacks, where at least one such pair was produced by another signer $j \neq i$.

- **Type II attacks**: the adversary breaks the security of the claiming procedure. In other words, he outputs a tuple $(M^\star, \sigma^\star, \tau^\star, \mathsf{upk}^\star)$ that causes the experiment of definition 5 to return 1 in either step 5 (*i.e.*, the adversary forges an honest member's claim) or step 6 (*i.e.*, an honest member's signature is claimed in the name of somebody else).

Lemmas 4 and 5 show that no PPT adversary can produce either kind of forgery as long as the 1-mTDH assumption holds. □

The proof of lemma 4 is based on standard techniques (notably in the simulation of signing queries as in [14, 45]) and separately considers two kinds of Type I attacks. An alternative proof allows considering a single kind of Type I attack. However, this alternative reduction is looser by another multiplicative factor of $O(\ell_b)$ (since it has to guess upfront which honest user will be framed) and eventually ends up with a degradation factor of $O(\ell_s \cdot \ell_b)$. By separately analyzing Type I-A and Type I-B attacks as in lemma 4, the security bound only declines with $\max(\ell_s, \ell_b)$.

Lemma 4. *The scheme is secure against framing attacks of Type I if the 1-mTDH problem is hard and if $\mathcal{H}$ is a collision-resistant hash function. More precisely, the advantage of any adversary after ℓ_s signing queries, ℓ_b $Q_{\mathsf{b\text{-}join}}$-queries and ℓ_c Q_{claim}-queries is at most*

$$\mathbf{Adv}^{\text{fra-I}}(\lambda, n) \leq 2 \cdot \mathbf{Adv}^{\text{CR}}(n) + 4n \cdot \max(\ell_s, \ell_b) \cdot \big(1 - \frac{\ell_c}{p}\big)^{-1} \cdot \mathbf{Adv}^{\text{1-mTDH}}(\lambda),$$

where the first term accounts for the probability of finding collisions on $\mathcal{H}$.

Proof. As discussed earlier, we distinguish two cases. The bound of the lemma's statement stems from the fact that $\mathcal{B}$ has to guess upfront, by flipping a fair binary coin independently of $\mathcal{A}$'s view, whether $\mathcal{A}$ will mount a Type I-A or Type I-B attack and set up the scheme accordingly. With probability 1/2, $\mathcal{B}$ guesses the correct type of attack and the result follows.

Type I-A attacks. We first assume that a Type I-A adversary $\mathcal{A}$ comes up with a forgery $(M^\star, \sigma^\star)$ that traces to some honest user $i \in U^b$ and that no message-signature pair (M, σ) such that $(M, \sigma) \equiv_s (M^\star, \sigma^\star)$ was produced by any honest group member. We show that such an adversary allows solving a problem that is not easier than 1-mTDH since it consists in finding a triple $(C_1, C_2, C_3) = (g^\mu, g^{b\mu}, g^{ab\mu})$ given only $(g, A = g^a, B = g^b)$. We note that a triple (C_1, C_2, C_3) is an admissible solution if and only if it satisfies

$$e(C_3, g) = e(C_2, A) \;\text{ and }\; e(C_1, B) = e(C_2, g). \tag{8}$$

Given a problem instance $(g, A = g^a, B = g^b)$, the simulator $\mathcal{B}$ chooses the GM private key $\omega, \alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_p^*$. It sets $u_1 = A \in \mathbb{G}$ and also picks $k \xleftarrow{\$} \{0, \ldots, n\}$, integers $\rho_f, \rho_{u,0}, \rho_0, \ldots, \rho_n \xleftarrow{\$} \mathbb{Z}_p^*$ and $\beta_0, \ldots, \beta_n \xleftarrow{\$} \{0, \ldots, 2\ell_s - 1\}$. It then defines $\Omega = g^\omega$, $v_0 = u_1^{\beta_0 - 2k\ell_s} \cdot g^{\rho_0}$, $v_i = u_1^{\beta_i} \cdot g^{\rho_i}$ for $i = 1, \ldots, n$ as well as $u_0 = g^{\rho_{u,0}}$ and $f = B^{\rho_f}$. It also sets $h_i = g^{\gamma_i} \in \mathbb{G}$ for $i = 0, \ldots, 4$ with $\gamma_0, \ldots, \gamma_4 \xleftarrow{\$} \mathbb{Z}_p^*$ and chooses $\mathbf{g}$ for the perfect soundness setting as specified by the setup algorithm, with $\vec{g_1} = (g_1 = g^{\alpha_1}, 1, g)$, $\vec{g_2} = (1, g_2 = g^{\alpha_2}, g)$ and $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$, for some random exponents $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p^*$. It finally chooses $\overline{f} = (f_0, f_1, \ldots, f_n) \xleftarrow{\$} \mathbb{G}^{n+1}$ at random.

Throughout the game, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows.

- $Q_{\mathcal{S}}$-queries: if $\mathcal{A}$ decides to corrupt the group manager, $\mathcal{B}$ hands him the group manager's private key $\mathcal{S} = \{\gamma_1, \omega, \alpha_1, \alpha_2\}$.

- $Q_{\mathsf{b\text{-}join}}$-queries: when $\mathcal{A}$, acting as a corrupted group manager, wants to introduce a new honest user i in the group, $\mathcal{B}$ starts interacting with $\mathcal{A}$ in an execution of the join protocol and assumes the role of the new user. Namely, $\mathcal{B}$ chooses $x_i \xleftarrow{\$} \mathbb{Z}_p^*$, sets $X_i = B^{x_i} \in \mathbb{G}$ and simulates $\mathcal{A}$'s view in the first step to force user i's membership secret to implicitly become $\mathsf{sec}_i = bx_i$ (and the associated public value to be X_i). At step 2 of the join protocol, $\mathcal{A}$ outputs K_1, K_2, K_3, y and $\mathcal{B}$ generates a signature on $X_i||K_1||K_2||K_3||g^y$. Then, $\mathcal{A}$ replies with the final part K_4 of user i's certificate.
- $Q_{\mathcal{Y}}$-queries: are treated as in the proof of lemma 1 and the simulator always knows the values requested by $\mathcal{A}$.
- Q_{sig}-queries: when $\mathcal{A}$ asks user $i \in U^b$ to sign a message M, algorithm $\mathcal{B}$ first retrieves the membership certificate (K_1, K_2, K_3, K_4, y) and the previously stored $X_i = B^{x_i}$. It randomly picks $\delta_1, \delta_2 \xleftarrow{\$} \mathbb{Z}_p^*$ and computes a tuple of traceability values $(T_1 = X_i^{\delta_1}, T_2 = g^{y\delta_2}, T_3 = g^{\delta_1+\delta_2})$, which is hashed as $\mathsf{m} = m_1 \ldots m_n = \mathcal{H}(M||T_1||T_2||T_3)$. Then, the number theoretic hash value $G_v(\mathsf{m}) = v_0 \cdot \prod_{j=1}^n v_j^{m_j}$ can be expressed as $G_v(\mathsf{m}) = u_1^{J_v(\mathsf{m})} \cdot g^{K_v(\mathsf{m})}$ where

$$J_v(\mathsf{m}) = \beta_0 - 2k\ell_s + \sum_{j=1}^{n} \beta_j m_j, \qquad K_v(\mathsf{m}) = \rho_0 + \sum_{j=1}^{n} \rho_j m_j.$$

  Here, $\mathcal{B}$ aborts if $J_v(\mathsf{m}) = 0$. Otherwise, it picks $r_s \xleftarrow{\$} \mathbb{Z}_p^*$ and computes

$$\Big(\theta_4 = K_4 \cdot G_v(\mathsf{m})^{r_s} \cdot T_1^{-\frac{K_v(\mathsf{m})}{J_v(\mathsf{m})}},\ \theta_5 = g^{r_s} \cdot T_1^{-\frac{1}{J_v(\mathsf{m})}}\Big),$$

  which has the required distribution since, if we implicitly define $\tilde{r_s} = r_s - (bx_i\delta_1)/J_v(\mathsf{m})$, it can be written $(\theta_4 = K_4 \cdot u_1^{x_i\delta_1} \cdot G_v(\mathsf{m})^{\tilde{r_s}}, \theta_5 = g^{\tilde{r_s}})$. Together with certificate components (K_1, K_2, K_3), the newly generated pair (θ_4, θ_5) allows computing a traceable signature since elements (θ_6, θ_7) can be obtained as $(\theta_6, \theta_7) = (X_i^{\gamma_1} \cdot h_2^y, X_i^{\gamma_3} \cdot h_4^y)$.
- Q_{claim}-queries: at any time, $\mathcal{A}$ is also allowed to request claims for signatures that were previously generated by honest group members. For each claiming query (i, M, σ), where the signature σ parses as $(T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$, $\mathcal{B}$ returns $\perp$ if $i \notin U^b$ or if Sigs does not contain an entry (i, M, σ') such that σ' contains (T_1, T_2, T_3). Otherwise, $\mathcal{B}$ is always able to compute a valid claim τ. Indeed, the pair $(D_{1,x}, D_{2,x}) = (f^{1/(bx_i)} \cdot G(\mathsf{m}_c)^{r_x}, T_1^{r_x})$ is computable from $f^{1/(bx_i)} = g^{\rho_f/x_i}$ and $(D_{1,y}, D_{2,y})$ does not involve any secret.

Finally, $\mathcal{A}$ outputs a traceable signature $\sigma^\star = (T_1^\star, T_2^\star, T_3^\star, \vec{\sigma_1}^\star, \ldots, \vec{\sigma_{11}}^\star, \pi_1^\star, \ldots, \pi_8^\star)$, for some message $M^\star$, that traces to some user $i^\star \in U^b$ and for which no honest user produced a signature involving the same $(M^\star, T_1^\star, T_2^\star, T_3^\star)$. In the event that Sigs contains a tuple (M', T_1', T_2', T_3') for which $\mathsf{m}^\star = \mathcal{H}(M^\star||T_1^\star||T_2^\star||T_3^\star) = \mathcal{H}(M'||T_1'||T_2'||T_3')$ but $(M', T_1', T_2', T_3') \neq (M^\star, T_1^\star, T_2^\star, T_3^\star)$, $\mathcal{A}$ was necessarily able to break the collision-resistance of $\mathcal{H}$. Otherwise, $\mathcal{B}$ uses α_1, α_2 to BBS-decrypt $\theta_i^\star = \sigma_{i,3} \cdot {\sigma_{i,1}^\star}^{-1/\alpha_1} \cdot {\sigma_{i,2}^\star}^{-1/\alpha_2}$ for $i = 1, \ldots, 9$. The perfect soundness of the proof system ensures that

$$\theta_3^\star = g^{s_{\mathsf{ID}^\star}} \qquad \theta_4^\star = u_0^{s_{\mathsf{ID}^\star}} \cdot u_1^{x'\delta_1} \cdot G_v(\mathsf{m}^\star)^{r_s} \qquad \theta_5^\star = g^{r_s} \qquad T_1^\star = g^{x'\delta_1},$$

for some exponents $s_{\mathsf{ID}^\star}, r_s, x', \delta_1 \in \mathbb{Z}_p^*$, and where

$$G_v(\mathsf{m}^\star) = v_0 \cdot \prod_{j=1}^{n} v_j^{m_j} = u_1^{J_v(\mathsf{m}^\star)} \cdot g^{K_v(\mathsf{m}^\star)}.$$

Then, the simulator $\mathcal{B}$ halts and declares failure if $J_v(\mathsf{m}^\star) \neq 0$. Otherwise, $\mathcal{B}$ is able to compute $\Theta = u_1^{x'\delta_1} = \theta_4^\star/(\theta_5^{\star K_v(\mathsf{m}^\star)} \cdot \theta_3^{\star \rho_{u,0}})$, which must satisfy the equality $e(\Theta, g) = e(u_1, T_1^\star)$. Since $\sigma^\star$ traces to user $i^\star$ by assumption, we must have $e(X_{i^\star}, T_3^\star/T_2^{\star 1/y^\star}) = e(T_1^\star, g)$, where $y^\star$ is part of the tracing trapdoor that $\mathcal{A}$ assigned to $i^\star$ and $X_{i^\star} = B^{x_{i^\star}}$ for some $x_{i^\star}$ which is known to $\mathcal{B}$. Since $u_1 = A$, we find that the assignment

$$C_1^\star = T_3^\star/T_2^{\star 1/y^\star} \qquad C_2^\star = T_1^{\star 1/x_{i^\star}} \qquad C_3^\star = \Theta^{1/x_{i^\star}}$$

must satisfy relation (8) and thus solve the problem instance.

$\mathcal{B}$'s probability not to abort can be shown to be at least $1/(2n\ell_s)$ as in the proof of lemma 2. The situation where $\sigma^\star$ accuses user $i^\star$ via the opening algorithm (instead of the implicit tracing) can be handled in the same way. Indeed, the perfect soundness of the proof system guarantees that, if a signature opens to user $i^\star$, the tracing algorithm necessarily points to the same user.

Type I-B attacks. We still have to consider Type I-B forgeries $\sigma^\star$ that open or trace to some user $i^\star$ but involve a tuple $(M^\star, T_1^\star, T_2^\star, T_3^\star)$ also appearing in a signature produced by $j^\star \neq i^\star \in U^b$. We show that such forgeries imply an algorithm solving an equivalent formulation of the Diffie-Hellman problem (that is not easier than 1-mTDH) which is to compute $g^{b/a}$ given $(A = g^a, B = g^b)$. The simulator $\mathcal{B}$ performs the setup as in the expectation of a Type I-A attack but chooses $u_1 \in \mathbb{G}$ so as to know $\rho_{u,1} = \log_g(u_1)$. Another difference is that, while f is still set as $f = B^{\rho_f}$, for some random $\rho_f \xleftarrow{\$} \mathbb{Z}_p$, the vector $\overline{f} = (f_0, f_1, \ldots, f_n)$ is chosen by setting $f_i = f^{\alpha_{f,i}} \cdot A^{\beta_{f,i}}$, for randomly drawn $\alpha_{f,i}, \beta_{f,i} \xleftarrow{\$} \mathbb{Z}_p$, for $i = 0$ to n. As in the case of Type I-A attacks, $h_1, h_3 \in \mathbb{G}$ are chosen in such a way that $\mathcal{B}$ knows $\gamma_1 = \log_g(h_1)$ and $\gamma_3 = \log_g(h_3)$. Additionally, $\mathcal{B}$ chooses an index $k \xleftarrow{\$} \{1, \ldots, \ell_b\}$ and interacts with the adversary $\mathcal{A}$ as follows.

- $Q_{\mathcal{S}}$ and $Q_{\mathcal{Y}}$-queries: are handled as when dealing with Type I-A attacks.
- $Q_{\mathsf{b\text{-}join}}$-queries: when $\mathcal{A}$ decides to introduce a new user i, $\mathcal{B}$'s behavior depends on the index $i \in \{1, \ldots, \ell_b\}$ of the $Q_{\mathsf{b\text{-}join}}$-query.
  - If $i \neq k$, $\mathcal{B}$ runs the join protocol on behalf of the prospective honest group member. It chooses $x_i \xleftarrow{\$} \mathbb{Z}_p^*$, sets $X_i = A^{x_i} \in \mathbb{G}$ and simulates $\mathcal{A}$'s view to end up with the public value X_i (so that $\mathsf{sec}_i = a x_i$ is the implicitly defined membership secret). The next steps of the join protocol are executed as in the case of Type I-A forgeries.
  - If $i = k$, $\mathcal{B}$ proceeds as in the first case but rather simulates $\mathcal{A}$'s view to force the public value X_k to become $B = g^b$ (which implicitly sets $\mathsf{sec}_k = b$).
- Q_{sig}-queries: since $\mathcal{B}$ knows $\rho_{u,1} = \log_g(u_1)$, $\gamma_1 = \log_g(h_1)$ and $\gamma_3 = \log_g(h_3)$, it can perfectly simulate Q_{sig}-queries by computing θ_4 using $u_1^{\mathsf{sec}_i} = X_i^{\rho_{u,1}}$ and (θ_6, θ_7) as $(X_i^{\gamma_1} \cdot h_2^y, X_i^{\gamma_3} \cdot h_4^y)$. The simulator also retains the random values $\delta_1, \delta_2 \in \mathbb{Z}_p$ that are used to calculate T_1, T_2, T_3 at each query.
- Q_{claim}-queries: for each claiming query (i, M, σ), the simulator $\mathcal{B}$ first performs the same checks as in the case of Type I-A attacks. Then, $\mathcal{B}$ is able to generate claims on behalf of user k since $X_k = B$ and $f^{1/b} = g^{\rho_f}$ is computable. For each user $i \in U^b$ such that $i \neq k$, $\mathcal{B}$ first computes the n-bit string $\mathsf{m}_c = \mathcal{H}(M||T_1||T_2||T_3||\mathsf{upk}) = m_{c,1} \ldots m_{c,n} \in \{0,1\}^n$ and evaluates $J = \alpha_{f,0} + \sum_{j=1}^n \alpha_{f,j} m_{c,j}$ and $K = \beta_{f,0} + \sum_{j=1}^n \beta_{f,j} m_{c,j}$ such that $G_f(\mathsf{m}_c) = f^J \cdot A^K$. It aborts in the unlikely event that $J = 0$ (since the random values $\alpha_{f,0}, \ldots, \alpha_{f,n}$ are completely independent of $\mathcal{A}$'s view, this occurs with negligible probability $1/p$ at each query). Otherwise,

$\mathcal{B}$ recalls the value δ_1 such that $T_1 = X_i^{\delta_1} = A^{x_i\delta_1}$. It then picks $r_x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and computes $(D_{1,x}, D_{2,x}) = \big(G_f(\mathsf{m}_c)^{r_x} \cdot g^{-K/(x_i J)}, T_1^{r_x} \cdot g^{-\delta_1/J}\big)$, which has the proper distribution since it can be expressed as $(f^{1/(ax_i)} \cdot G_f(\mathsf{m}_c)^{\tilde{r}_x}, T_1^{\tilde{r}_x})$, where $\tilde{r}_x = r_x - 1/(ax_iJ)$.

After polynomially-many queries, $\mathcal{A}$ outputs a Type I-B forgery $\sigma^\star$ that opens or traces to some user $i^\star \in U^b$. By assumption, another user $j^\star \in U^b$ must have issued a signature with the same $(M^\star, T_1^\star, T_2^\star, T_3^\star)$. At this stage, $\mathcal{B}$ fails if $j^\star$ is not the user that was introduced at the k^{th} $Q_{\mathsf{b\text{-}join}}$-query. With probability $1/\ell_b$ however, $\mathcal{B}$ correctly guessed k. It also still knows the exponent δ_1 that was used by user $j^\star$ to calculate $T_1^\star = X_k^{\delta_1} = B^{\delta_1}$. Moreover, since $\sigma^\star$ traces to user $i^\star$, there must exist $\delta_1', \delta_2' \in \mathbb{Z}_p$ such that $(T_1^\star, T_2^\star, T_3^\star) = (A^{x_{i^\star}\delta_1'}, g^{y_{i^\star}\delta_2'}, g^{\delta_1'+\delta_2'})$ and, given that $\mathcal{B}$ also knows the tracing trapdoor $y_{i^\star}$ (that was supplied by $\mathcal{A}$ during the k^{th} $Q_{\mathsf{b\text{-}join}}$-query), it can compute $g^{\delta_1'} = T_3^\star / T_2^{\star 1/y_{i^\star}}$ and thus $g^{b/a} = (g^{\delta_1'})^{(x_{i^\star}/\delta_1)}$. □

The security against Type II attacks relies on the standard Computational Diffie-Hellman assumption and the security of the ordinary signature scheme in the sense of [31]. It makes (now classical) use of the technique of [13, 45] and the "programmability" [37] of the number theoretic hash function $G_f : \{0,1\}^n \to \mathbb{G}$. The proof is deferred to appendix E.

Lemma 5. *The scheme is secure against Type II framing attacks if the Computational Diffie-Hellman problem is hard in $\mathbb{G}$, if $\mathcal{H}$ is a collision-resistant hash function and if the ordinary digital signature is existentially unforgeable under chosen-message attacks. More precisely, the advantage of any adversary after ℓ_s signing queries, ℓ_b $Q_{\mathsf{b\text{-}join}}$-queries and ℓ_c claiming queries is at most $\mathbf{Adv}^{\text{fra-II}}(\lambda, n) \leq \mathbf{Adv}^{\text{CR}}(n) + 2\cdot\mathbf{Adv}^{\text{euf-sig}}(\lambda) + 4\cdot n\cdot\ell_c\cdot\mathbf{Adv}^{\text{CDH}}(\lambda)$, where the first and second terms account for the probability of breaking the collision-resistance of $\mathcal{H}$ and the existential unforgeability of the signature, respectively.*

Proof. Given in appendix E. □

4.3 Anonymity

Since we introduced a claiming oracle in the modeling of anonymity, we are faced with the additional difficulty of simulating the Q_{claim} oracle without always knowing the appropriate secret elements in the reduction. Fortunately, we can prove the result using only the DLIN assumption.

Theorem 3 (Anonymity). *The scheme is anonymous assuming that the Decision Linear Problem is hard in $\mathbb{G}$. More precisely, we have*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{anon}}(\lambda) \leq \ell_p \cdot \Big(\frac{\ell_c}{p} + 3 \cdot \mathbf{Adv}_{\mathbb{G}}^{\text{DLIN}}(\lambda)\Big)$$

where ℓ_p and ℓ_c denote the number of $Q_{\mathsf{p\text{-}join}}$-queries and the number of Q_{claim}-queries, respectively.

Proof. The proof proceeds with a sequence of games organized in such a way that even a computationally unbounded adversary has no advantage in the final game while the first one is the real attack game. Throughout the sequence, we always call S_i the event that the adversary wins and his advantage is measured by $Adv_i = |\Pr[S_i] - 1/2|$. Also, when we speak of user i, for some $i \in \{1, \ldots, \ell_p\}$, we mean the i^{th} user that joins the group after a $Q_{\mathsf{p\text{-}join}}$ query.

Game 1: operates as the real game does. The challenger $\mathcal{B}$ performs the setup according to the specification of the scheme. It chooses random exponents $\omega, \gamma_1, \ldots, \gamma_4, \rho_{u,1}, \xi_1, \xi_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and sets g^ω, $h_i = g^{\gamma_i}$ for $i = 1, \ldots, 4$ and $u_1 = g^{\rho_{u,1}}$. It also picks $h_0, u_0, f, g_1, g_2 \stackrel{\$}{\leftarrow} \mathbb{G}$ and vectors $\overline{v} \in \mathbb{G}^{n+1}$, $\overline{f} = (f_0, f_1, \ldots, f_n) \stackrel{\$}{\leftarrow} \mathbb{G}^{n+1}$ and defines $\vec{g_1} = (g_1, 1, g)$, $\vec{g_2} = (1, g_2, g)$, $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$. Using ω, γ_1, it answers $Q_{\mathsf{a\text{-}join}}$ and $Q_{\mathsf{p\text{-}join}}$-queries and updates $\mathsf{transcripts}$ each time. When the adversary $\mathcal{A}$ makes a Q_{reveal}-query (resp. Q_{sig}-query), $\mathcal{B}$ uses the database $\mathsf{transcripts}$ to return the queried tracing trapdoor (resp. recover the appropriate cert_i and sec_i and generate a signature on behalf of user $i \in U^p$). At the challenge phase, the adversary chooses two users $i_0^\star, i_1^\star \in U^p$ such that $i_0^\star, i_1^\star \notin \mathsf{revs}$ and is returned a traceable signature $\sigma^\star$ on behalf of signer $i_{d^\star}^\star$. Unlike what occurs in the real game, the challenger retains the values $\gamma_i = \log_g(h_i)$, for $i = 1, \ldots, 4$, and $\rho_{u,1} = \log_g(u_1)$ but this does not impact $\mathcal{A}$'s behavior. Eventually, he outputs a guess $d' \in \{0, 1\}$ and his advantage is $Adv_1 = |\Pr[S_1] - 1/2|$, where S_1 denotes the event that $d' = d^\star$.

Game 2: we modify the simulation. At the beginning, $\mathcal{B}$ picks an index $i^\star \stackrel{\$}{\leftarrow} \{1, \ldots, \ell_p\}$. In the challenge phase, $\mathcal{B}$ aborts if $\mathcal{A}$'s chosen pair $(i_0^\star, i_1^\star)$ does not contain $i^\star$. It also fails if $i^\star$ is ever queried to Q_{reveal} before the challenge step. Assuming that $\mathcal{B}$ is lucky when choosing $i^\star$ at the outset of the game (which is the case with probability $2/\ell_p$ since $i^\star$ is independent of $\mathcal{A}$'s view), the introduced abortion rule does not apply. We can write $Adv_2 = 2 \cdot Adv_1/\ell_p$.

Game 3: we add yet another abortion rule. At the challenge step, we must have $i^\star \in \{i_0^\star, i_1^\star\}$ unless the abortion rule of Game 2 applies. The new rule is the following: when $\mathcal{B}$ flips its secret coin $d^\star \stackrel{\$}{\leftarrow} \{0, 1\}$, it aborts if $i_{d^\star}^\star \neq i^\star$. With probability $1/2$, this new rule does not apply and we have $Adv_3 = 1/2 \cdot Adv_2$.

Game 4: we bring two modifications to the setup phase. Namely, $\mathcal{B}$ considers random group elements $Z_1 = g^{z_1}$, $Z_2 = g^{z_2}$ in $\mathbb{G}$. It first defines $f = Z_2^{\rho_f}$ for a random $\alpha_f \stackrel{\$}{\leftarrow} \mathbb{Z}_p$. The vector $\vec{f} = (f_0, f_1, \ldots, f_n) \in \mathbb{G}^{n+1}$ is then generated as follows. For $i = 0$ to n, it picks $\alpha_{f,i}, \beta_{f,i} \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and sets $f_i = f^{\alpha_{f,i}} \cdot Z_1^{\beta_{f,i}}$. This change is purely conceptual since the distribution of f and $(f_0, f_1, \ldots, f_n)$ remains unchanged. We thus have $\Pr[S_4] = \Pr[S_3]$ and $Adv_4 = Adv_3$.

In the upcoming games, it will be convenient to define functions $J, K : \{0, 1\}^n \to \mathbb{Z}_p$ that map n-bit strings $\mathsf{m} = m_1 \ldots m_n$ onto $J(\mathsf{m}) = \alpha_{f,0} + \sum_{j=}^{n} \alpha_{f,j} m_j$ and $K(\mathsf{m}) = \beta_{f,0} + \sum_{j=}^{n} \beta_{f,j} m_j$.

Game 5: we introduce a failure event F_5. Namely, when the simulator $\mathcal{B}$ has to claim a previously generated signature $\sigma = (T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$, it first computes the hash value $\mathsf{m}_c = m_{c,1} \ldots m_{c,n} = \mathcal{H}(M||T_1||T_2||T_3||\mathsf{upk}[i]) \in \{0, 1\}^n$, evaluates $J(\mathsf{m}_c) = \alpha_{f,0} + \sum_{j=1}^{n} \alpha_{f,j} m_{c,j}$ and aborts in the event[5] that $J(\mathsf{m}_c) = 0$. Since the values $\{\alpha_{f,i}\}_{i=0}^n$ are completely random and independent of $\mathcal{A}$'s view, this can only happen by pure chance. At each query, the probability to have $J(\mathsf{m}_c) = 0$ is thus at most $1/p$. In addition, Game 5 and Game 4 proceed identically until event F_5 causes $\mathcal{B}$ to abort. It comes that $|\Pr[S_5] - \Pr[S_4]| \leq \Pr[F_5] \leq \ell_c/p$.

Game 6: we modify the treatment of signing queries involving user $i^\star$. Now, $\mathcal{B}$ re-uses the values $z_1 = \log_g(Z_1)$ and $z_2 = \log_g(Z_2)$ (introduced in Game 4) and implicitly defines user $i^\star$'s membership secret to be $\mathsf{sec}_{i^\star} = z_1$ while the value y of his membership certificate is z_2. More precisely, its

[5] A difference with the security proof of Waters' identity-based encryption scheme [45] is that we do not need $J(.)$ to cancel in the challenge phase, which is why $\{\alpha_{f,i}\}_{i=0}^n$ can be chosen uniformly in $\mathbb{Z}_p$ (rather than in a much smaller interval as in [45]). For the same reason, Game 5 and Game 4 can be linked by a transition based on a failure event of small probability and no artificial abort step is needed.

membership certificate is calculated as per

$$K_1 = \left(h_0 \cdot Z_1^{\gamma_1} \cdot Z_2^{\gamma_2}\right)^{1/(\omega+s_{\mathsf{ID}_{i^\star}})} \quad K_2 = g^{1/(\omega+s_{\mathsf{ID}_{i^\star}})} \quad K_3 = g^{s_{\mathsf{ID}_{i^\star}}} \quad K_4 = u^{s_{\mathsf{ID}_{i^\star}}},$$

for a random $s_{\mathsf{ID}_{i^\star}} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. Unless the failure event of Game 2 occurs, no Q_{reveal} query is made for $i^\star$ and $\mathcal{B}$ can answer signing queries without using z_1, z_2 and knowing only Z_1, Z_2. Indeed, signing queries related to $i^\star$ can be answered by calculating $\theta_6 = Z_1^{\gamma_1} \cdot Z_2^{\gamma_2}$, $\theta_7 = Z_1^{\gamma_3} \cdot Z_2^{\gamma_4}$, $\theta_8 = Z_1$, $\theta_9 = Z_2$, $T_1 = Z_1^{\delta_1}$, $T_2 = Z_2^{\delta_2}$ and $T_3 = g^{\delta_1+\delta_2}$, using random $\delta_1, \delta_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. It then sets $u_1^{z_1\delta_1} = T_1^{\rho_{u,1}}$ in the computation of θ_4. The challenge signature, which $\mathcal{B}$ generates on behalf of $i^\star$ unless one of the previous failure events occurs, is produced in the same way. These changes do not affect $\mathcal{A}$'s view, so that $\Pr[S_6] = \Pr[S_5]$ and $Adv_6 = Adv_5$.

Game 7: we modify the treatment of Q_{claim}-queries involving user $i^\star$. Recall that, since Game 6, the simulator $\mathcal{B}$ answers Q_{sig}-queries involving $i^\star$ by setting $T_1 = Z_1^{\delta_1}$, $T_2 = Z_2^{\delta_2}$ and $T_3 = g^{\delta_1+\delta_2}$ for random $\delta_1, \delta_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. In this game, when it comes to claim a previously generated signature $(T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$, $\mathcal{B}$ computes $\mathsf{m}_c = m_{c,1} \ldots m_{c,n} = \mathcal{H}(M||T_1||T_2||T_3||\mathsf{upk}[i^\star])$ and then evaluates functions $J(\mathsf{m}_c) = \alpha_{f,0} + \sum_{j=}^{n} \alpha_{f,j} m_{c,j}$ and $K(\mathsf{m}_c) = \beta_{f,0} + \sum_{j=}^{n} \beta_{f,j} m_{c,j}$ such that $G_f(\mathsf{m}_c) = f_0 \cdot \prod_{j=1}^{n} f_j^{m_{c,j}} = f^{J(\mathsf{m}_c)} \cdot Z_1^{K(\mathsf{m}_c)}$. We must have $J(\mathsf{m}_c) \neq 0$ unless the failure event introduced in Game 5 applies. As long as $J(\mathsf{m}_c) \neq 0$, $\mathcal{B}$ can pick $r_x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$, implicitly define $\tilde{r}_x = r_x - 1/(z_1 J(\mathsf{m}_c))$ and compute the first part of the claim as

$$(D_{x,1}, D_{x,2}) = (G_f(\mathsf{m}_c)^{r_x} \cdot g^{-K(\mathsf{m}_c)/J(\mathsf{m}_c)},\ T_1^{r_x} \cdot g^{-\delta_1/J(\mathsf{m}_c)}) = (f^{1/z_1} \cdot G_f(\mathsf{m}_c)^{\tilde{r}_x},\ T_1^{\tilde{r}_x}). \qquad (9)$$

As for the second piece $(D_{y,1}, D_{y,2}) = (f^{1/z_2} \cdot G_f(\mathsf{m}_c)^{r_y}, T_2^{r_y})$ of the claim, $\mathcal{B}$ can generate it from $f^{1/z_2} = g^{\rho_f}$, which is computable thanks to the way that f is chosen since Game 4. In Game 7, we note that $\mathcal{B}$ does not use the values $z_1, z_2 \in \mathbb{Z}_p$ at any time. Still, the introduced modifications are only conceptual and do not alter $\mathcal{A}$'s view. We can thus write $\Pr[S_7] = \Pr[S_6]$ and $Adv_7 = Adv_6$

Game 8: we modify the setup phase and choose $\vec{g_3}$ as $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2} \odot (1, 1, g)^{-1}$ instead of $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$. We note that vectors $\vec{g_1}, \vec{g_3}, \vec{g_3}$ are now linearly independent. Any noticeable change in the adversary's behavior is easily seen[6] to imply a statistical test for the Decision Linear problem so that we can write $|\Pr[S_8] - \Pr[S_7]| = 2 \cdot \mathbf{Adv}^{\mathrm{DLIN}}(\mathcal{B})$.

Game 9: in this game, we modify the generation of the challenge signature $\sigma^\star$ on behalf of $i^\star$. The generation of $\vec{\sigma_1}^\star, \ldots, \vec{\sigma_9}^\star$ and $\pi_1^\star, \ldots, \pi_5^\star$ is still made using the actual witnesses $\mathsf{sec}_{i\star}, \mathsf{cert}_{i^\star}$. In particular, $\mathcal{B}$ sets $\vec{\sigma_8}^\star = \iota(Z_1) \odot \vec{g_1}^{r_8} \odot \vec{g_2}^{s_8} \odot \vec{g_3}^{t_8}$ and $\vec{\sigma_9}^\star = \iota(Z_2) \odot \vec{g_1}^{r_9} \odot \vec{g_2}^{s_9} \odot \vec{g_3}^{t_9}$ as in Game 5. However, instead of generating $\vec{\sigma_{10}}^\star$, $\vec{\sigma_{11}}^\star$ as well as $\pi_6^\star$, $\pi_7^\star$ and $\pi_8^\star$ using δ_1, δ_2 such that $T_1^\star = Z_1^{\delta_1}$ and $T_2^\star = Z_2^{\delta_2}$, $\mathcal{B}$ uses the simulated reference string and its trapdoor information (ξ_1, ξ_2) to generate simulated proofs. Namely, $\vec{\sigma_{10}}^\star = \vec{g_1}^{r_{10}} \odot \vec{g_2}^{s_{10}}$ and $\vec{\sigma_{11}}^\star = \vec{g_1}^{r_{11}} \odot \vec{g_2}^{s_{11}}$ are both

[6] Indeed, $\Pr[\mathcal{B}(g_1, g_2, g_1^{\xi_1}, g_2^{\xi_2}, g^{\xi_1+\xi_2}) = 1]$ and $\Pr[\mathcal{B}(g_1, g_2, g_1^{\xi_1}, g_2^{\xi_2}, g^{\xi_1+\xi_2-1}) = 1]$ are both within distance $\mathbf{Adv}^{\mathrm{DLIN}}(\mathcal{B})$ from $\Pr[\mathcal{B}(g_1, g_2, g_1^{\xi_1}, g_2^{\xi_2}, g^z) = 1]$, where z is random.

calculated as commitments to 0. As for proof elements $\pi_6^\star, \pi_7^\star, \pi_8^\star$, $\mathcal{B}$ obtains them as

$$\begin{aligned}
\pi_6^\star &= (\vec{\pi_{61}}^\star, \vec{\pi_{62}}^\star, \vec{\pi_{63}}^\star) \\
&= \Big(\vec{\sigma_8}^{\star r_{10}} \odot \iota(g)^{r_{10}t_8} \odot \iota(T_1^\star)^{-\xi_1} \odot \vec{\sigma_{10}}^{\star -\xi_1 t_8}, \\
&\quad \vec{\sigma_8}^{\star s_{10}} \odot \iota(g)^{s_{10}t_8} \odot \iota(T_1^\star)^{-\xi_2} \odot \vec{\sigma_{10}}^{\star -\xi_2 t_8},\ \vec{\sigma_{10}}^{\star t_8}\Big), \\
\pi_7^\star &= (\vec{\pi_{71}}^\star, \vec{\pi_{72}}^\star, \vec{\pi_{73}}^\star) \\
&= \Big(\vec{\sigma_9}^{\star r_{11}} \odot \iota(g)^{r_{11}t_9} \odot \iota(T_2^\star)^{-\xi_1} \odot \vec{\sigma_{11}}^{\star -\xi_1 t_9}, \\
&\quad \vec{\sigma_9}^{\star s_{11}} \odot \iota(g)^{s_{11}t_9} \odot \iota(T_1^\star)^{-\xi_2} \odot \vec{\sigma_{11}}^{\star -\xi_2 t_9},\ \vec{\sigma_{11}}^{\star t_9}\Big), \\
\pi_8^\star &= \Big(g^{r_{10}+r_{11}} \cdot T_3^{\star -\xi_1},\ g^{s_{10}+s_{11}} \cdot T_3^{\star -\xi_2}\Big).
\end{aligned}$$

It can be checked that verification equations 6, 7 and 8 are still satisfied by the above assignment. To achieve perfect witness indistinguishability, $\pi_6^\star$ and $\pi_7^\star$ must be re-randomized (as explained in [36]) to be uniform in the space of valid proofs for quadratic equations. On a simulated CRS $(\vec{g_1}, \vec{g_2}, \vec{g_3})$, simulated proofs are known to be perfectly indistinguishable from real proofs. Hence, this change is only conceptual and we have $\Pr[S_9] = \Pr[S_8]$.

Game 10: we bring a new change to the generation of the challenge $\sigma^\star$. In Game 9, we had $T_1^\star = Z_1^{\delta_1}$, $T_2^\star = Z_2^{\delta_2}$ and $T_3^\star = g^{\delta_1+\delta_2}$, where $\delta_1, \delta_2 \xleftarrow{\$} \mathbb{Z}_p^*$. Now, $\mathcal{B}$ rather chooses $T_3^\star \xleftarrow{\$} \mathbb{G}$ at random (and still computes $\pi_6^\star, \pi_7^\star, \pi_8^\star$ from $T_1^\star, T_2^\star, T_3^\star$ as in Game 9). Under the Decision Linear assumption[7] in $\mathbb{G}$, this change is not noticeable by $\mathcal{A}$ and we have $|\Pr[S_{10}] - \Pr[S_9]| \leq \mathbf{Adv}^{\mathrm{DLIN}}(\mathcal{B})$.

In Game 10, we claim that $\Pr[S_{10}] = 1/2$ (and Adv_{10} is thus zero). To see this, let us consider what $\mathcal{A}$ knows in the information theoretic sense. By observing two signatures from $i^\star = i_{d^\star}^\star$ (and more precisely the traceability values T_1, T_2, T_3 in each signature), $\mathcal{A}$ can figure out the values of his membership secret $\mathsf{sec}_{i^\star} = z_1$ and part $y = z_2$ of his membership certificate. She can also infer their counterpart for user $i_{1-d^\star}^\star$ by requesting two signatures from the latter.

When $\mathcal{A}$ sees the challenge signature $\sigma^\star$, however, $T_1^\star$, $T_2^\star$ and $T_3^\star$ reveal no information on $d^\star \in \{0,1\}$. Indeed, $T_3^\star$ is completely random and $T_1^\star$, $T_2^\star$ are compatible with either candidate $i_0^\star, i_1^\star$. Moreover, $\vec{\sigma_1}^\star, \ldots, \vec{\sigma_9}^\star$ are all perfectly hiding commitments and, in the WI setting, proofs $\pi_1^\star, \ldots, \pi_5^\star$ reveal no information on underlying witnesses $\mathsf{sec}_{i^\star}, \mathsf{cert}_{i^\star}$. Finally $\vec{\sigma_{10}}^\star$, $\vec{\sigma_{11}}^\star$ are independent of $d^\star \in \{0,1\}$ and so are simulated proofs $\pi_6^\star, \pi_7^\star, \pi_8^\star$. □

5 Conclusion

This paper described the first efficient construction of traceable signature in the standard model. We additionally extended the original model in order to offer support for abuse-free non-interactive claiming mechanisms.

Securely implementing all the functionalities of the primitive without appealing to the random oracle idealization raised its deal of technical issues and we had to use several (non-standard) intractability assumptions to solve all problems encountered on the road. It would be interesting to see if TS systems can be even more efficiently obtained without sacrificing security guarantees

[7] We note that the DLIN distinguisher that "bridges" between Game 10 and Game 9 cannot directly decide whether $(Z_1, Z_2, T_1^\star, T_2^\star, T_3^\star)$ forms a linear tuple by generating a claim for the challenge signature $\sigma^\star$. The reason is that it does not know the exponent $\delta_1 = \log_{Z_1}(T_1^\star)$, which prevents it from computing a claim as per (9).

in the standard model. Another open problem would be to extend the results of [10] and design multi-group extensions of traceable signatures outside the random oracle heuristic.

References

1. M. Abe and S. Fehr. Perfect NIZK with adaptive soundness. In *Theory of Cryptography Conference – TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 118–136. Springer, 2007.
2. J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Eurocrypt'02*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.
3. G. Ateniese, J. Camenisch, S. Hohenberger, and B. de Medeiros. Practical group signatures without random oracles. Cryptology ePrint Archive: Report 2005/385, 2005.
4. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Crypto 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 255–270. Springer, 2000.
5. M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable Proofs and Delegatable Anonymous Credentials. In *Crypto 2009*, *Lecture Notes in Computer Science* series. Springer, 2009.
6. M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *Theory of Cryptography Conference – TCC 2008*, volume 4948 of *Lecture Notes in Computer Science*, pages 356–374. Springer, 2008.
7. M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Eurocrypt 2003*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629. Springer, 2003.
8. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *1st ACM Conference on Computer and Communications Security (ACM CCS'93)*, pages 62–73. ACM Press, 1993.
9. M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, volume 3376 of *Lecture Notes in Computer Science*, pages 136–153. Springer, 2005.
10. V. Benjumea, S. G. Choi, J. Lopez, M. Yung. Fair traceable multi-group signatures. In *Financial Cryptography 2008*, volume 5143 of *Lecture Notes in Computer Science*, pages 231–246. Springer, 2008.
11. O. Blazy, G. Fuchsbauer, M. Izabachène, A. Jambert, H. Sibert, D. Vergnaud. Batch Groth-Sahai. In *Applied Cryptography and Network Security (ACNS'10)*, volume 6123 of *Lecture Notes in Computer Science*, pages 218–235, 2010.
12. A. Boldyreva, M. Fischlin, A. Palacio, and B. Warinschi. A closer look at PKI: Security and efficiency. In *PKC'07*, LNCS 4450, pages 458–475, 2007.
13. D. Boneh and X. Boyen. Short signatures without random oracles. In *Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer, 2004.
14. D. Boneh and X. Boyen. Efficient selective-ID Secure identity-based encryption without random oracles. In *Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2004.
15. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Crypto 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer, 2004.
16. D. Boneh, E.-J. Goh, and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography Conference – TCC 2005*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.
17. D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *ACM Conference on Computer and Communications Security (ACM CCS'04)*, pages 168–177. ACM Press, 2004.
18. X. Boyen and C. Delerablée. Expressive subgroup signatures. In *Security and Cryptography in Networks (SCN 2008)*, volume 5229 of *Lecture Notes in Computer Science*, pages 185–200. Springer, 2008.
19. X. Boyen and B. Waters. Compact group signatures without random oracles. In *Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 427–444. Springer, 2006.
20. X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography 2007 (PKC'07)*, volume 4450 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2007.
21. R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, 2004.
22. D. Chaum and E. van Heyst. Group signatures. In *Eurocrypt 1991*, volume 547 of *Lecture Notes in Computer Science*, pages 257–265. Springer, 1991.

23. S. G. Choi, K. Park, and M. Yung. Short traceable signatures based on bilinear pairings. In *International Workshop on Security 2006 (IWSEC'06)*, volume 4266 of *Lecture Notes in Computer Science*, pages 88–103. Springer, 2006.
24. S. Chow. Real Traceable Signatures. In *Selected Areas in Cryptography 2009 (SAC'09)*, volume 5867 of *Lecture Notes in Computer Science*, pages 92–107. Springer, 2009.
25. I. Damgård. Towards practical public key systems secure against chosen ciphertext attacks. In *Crypto 1991*, volume 576 of *Lecture Notes in Computer Science*, pages 445–456. Springer, 1991.
26. C. Delerablée and D. Pointcheval. Dynamic fully anonymous short group signatures. In *Viecrypt 2006*, volume 4341 of *Lecture Notes in Computer Science*, pages 193–210. Springer, 2006.
27. A. Dent. The hardness of the DHK problem in the generic group model. Cryptology ePrint Archive: Report 2006/156, 2006.
28. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto 1986*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.
29. J. Furukawa and H. Imai. An efficient group signature scheme from bilinear maps. In *Australasian Conference Information Security and Privacy (ACISP 2005)*, volume 3574 of *Lecture Notes in Computer Science*, pages 455–467. Springer, 2005.
30. H. Ge, S.-R. Tate. Traceable signature: better efficiency and beyond. In *ICCSA (3) 2006*, volume 3982 of *Lecture Notes in Computer Science*, pages 327–337. Springer, 2006.
31. S. Goldwasser, S. Micali, R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Comput.* 17(2), pp. 281–308, 1988.
32. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Asiacrypt 2006*, volume 4284 of *Lecture Notes in Computer Science*, pages 444–459. Springer, 2006.
33. J. Groth. Fully anonymous group signatures without random oracles. In *Asiacrypt 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 164–180. Springer, 2007.
34. J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for NIZK. In *Crypto 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 97–111. Springer, 2006.
35. J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for NP. In *Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2006.
36. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432. Springer, 2008.
37. D. Hofheinz, E. Kiltz. Programmable Hash Functions and Their Applications. In *Crypto'08*, volume 5157 of *Lecture Notes in Computer Science*, pages 21–38. Springer, 2008.
38. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Eurocrypt 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer, 2004.
39. A. Kiayias and M. Yung. Efficient secure group signatures with dynamic joins and keeping anonymity against group managers. In *Mycrypt 2005*, volume 3715 of *Lecture Notes in Computer Science*, pages 151–170. Springer, 2005.
40. B. Libert and M. Yung. Efficient Traceable Signatures in the Standard Model. In *Pairing-Based Cryptography 2009 (Pairing'09)*, volume 5671 of *Lecture Notes in Computer Science*, pages 187–205. Springer, 2009.
41. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In *Eurocrypt 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, 2006.
42. M. Naor. On cryptographic assumptions and challenges. In *Crypto'03*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.
43. L. Nguyen and R. Safavi-Naini. Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings. In *Asiacrypt 2004*, volume 3329 of *Lecture Notes in Computer Science*, pages 372–386. Springer, 2004.
44. D. Pointcheval, J. Stern. Security Arguments for Digital Signatures and Blind Signatures. In *J. of Cryptology 13(3)*, pp. 361–396, 2000.
45. B. Waters. Efficient identity-based encryption without random oracles. In *Eurocrypt 2005*, volume 2567 of *Lecture Notes in Computer Science*. Springer, 2005.

A Security Definition for Digital Signatures

A digital signature is a triple of algorithms $\Sigma = (\text{Keygen}, \text{Sign}, \text{Verify})$ such that:

(i) On input of a security parameter $\lambda \in \mathbb{N}$, $\text{Keygen}(\lambda)$ outputs a key pair $(\mathsf{usk}, \mathsf{upk})$.

(ii) Sign is a possibly randomized algorithm that takes in a message M and the private key usk. It outputs a signature σ.

(iii) Verify is a deterministic algorithm that takes as input a message M, a public key upk and a signature σ. It outputs 1 or 0 depending on whether σ is deemed valid or not.

The standard notion of security for digital signatures is the one given by Goldwasser, Micali and Rivest [31].

Definition 7 ([31]). *A digital signature Σ is existentially unforgeable under chosen-message attacks if, for any PPT adversary $\mathcal{A}$ playing the game hereafter, it holds that $\mathbf{Adv}^{\text{euf-sig}}(\mathcal{A}) \in \mathsf{negl}(\lambda)$.*

1. *The game begins with the challenger running $(\mathsf{usk}, \mathsf{upk}) \leftarrow \text{Keygen}(\lambda)$ and giving upk to the adversary $\mathcal{A}$.*
2. *$\mathcal{A}$ adaptively interacts with a signing oracle. Namely, at any time, $\mathcal{A}$ can ask for a signature on an arbitrary message M and the challenger responds by computing $\sigma \leftarrow \text{Sign}_{\mathsf{usk}}(M)$ and returning σ to $\mathcal{A}$.*
3. *$\mathcal{A}$ outputs a message $M^\star$ and a signature $\sigma^\star$. He wins if $M^\star$ was never queried for signature and $\text{Verify}_{\mathsf{upk}}(M^\star, \sigma^\star) = 1$.*

$\mathcal{A}$'s advantage $\mathbf{Adv}^{\text{euf-sig}}(\mathcal{A})$ is simply his probability of victory, taken over all coin tosses.

B Groth's Key Generation Protocol

In [33], Groth described the following 5-move protocol that allows a prospective group member $\mathcal{U}$ and a group manager GM to jointly generate $X = g^x \in \mathbb{G}$ in such a way that only the user gets to know the membership secret $\mathsf{sec}_i = x \in \mathbb{Z}_p^*$ and the latter is further guaranteed to be uniformly distributed. The user $\mathcal{U}$ first generates g^a. Both parties run a coin-flipping protocol to generate a random value $b + c$, that also serves as a challenge when $\mathcal{U}$ proves knowledge of a, and the common output finally consists of $X = g^{a+b+c}$, whereas only $\mathcal{U}$ happens to know $x = a + b + c$.

1. $\mathcal{U}$ picks $a, r \xleftarrow{\$} \mathbb{Z}_p$, $\eta \xleftarrow{\$} \mathbb{Z}_p^*$ and sends $A = g^a$, $R = g^r$, $h = g^\eta$ to GM.
2. GM picks $b, s \xleftarrow{\$} \mathbb{Z}_p$ and sends a commitment $B = g^b \cdot h^s$ to $\mathcal{U}$.
3. $\mathcal{U}$ sends $c \xleftarrow{\$} \mathbb{Z}_p$ to GM.
4. GM opens the commitment B and sends the values b, s back to $\mathcal{U}$.
5. $\mathcal{U}$ checks that $B = g^b \cdot h^s$. If so, $\mathcal{U}$ sends $z = (b + c)a + r \bmod p$ and η to GM and outputs $x = a + b + c$.
6. GM finally checks that $\eta \in \mathbb{Z}_p^*$, $h = g^\eta$ and $A^{b+c} \cdot R = g^z$. If so, GM outputs $X = A \cdot g^{b+c}$.

Under the discrete logarithm assumption in $\mathbb{G}$, this protocol has black-box simulators that can emulate the view of a malicious user or a malicious group manager. In the former case, the simulator has rewind access to the malicious user and can force his private output to be a given value $x \in \mathbb{Z}_p$. In the latter case, the view of the malicious issuer can be simulated to get his output to be a given $X \in \mathbb{G}$. Moreover, the simulator does not need to know $x = \log_g(X)$.

C Details on the Construction of Proof Elements

To construct proof elements, the signer starts from an assignment of $\pi_1, \ldots, \pi_8$ that satisfies the verification equations, which can be obtained as follows.

$$\begin{aligned}
\pi_1 &= (\vec{\pi}_{1,1}, \vec{\pi}_{1,2}, \vec{\pi}_{1,3}) \\
&= \Big(\iota(\theta_1)^{r_3} \odot \vec{g_1}^{2r_1r_3} \odot \vec{g_2}^{s_1r_3+r_1s_3} \odot \vec{g_3}^{t_1r_3+r_1t_3} \odot \iota(\Omega \cdot \theta_3)^{r_1} \cdot \iota(g)^{-r_6}, \\
&\quad \iota(\theta_1)^{s_3} \odot \vec{g_1}^{r_1s_3+r_3s_1} \odot \vec{g_2}^{2s_1s_3} \odot \vec{g_3}^{t_1s_3+s_1t_3} \cdot \iota(\Omega \cdot \theta_3)^{s_1} \cdot \iota(g)^{-s_6}, \\
&\quad \iota(\theta_1)^{t_3} \cdot \vec{g_1}^{r_1t_3+r_3t_1} \odot \vec{g_2}^{s_1t_3+s_3t_1} \odot \vec{g_3}^{2t_1t_3} \odot \iota(\Omega \cdot \theta_3)^{t_1} \odot \iota(g)^{-t_6}\Big) \\
\pi_2 &= (\vec{\pi}_{2,1}, \vec{\pi}_{2,2}, \vec{\pi}_{2,3}) \\
&= \Big(\iota(\Omega \cdot \theta_3)^{r_2} \odot \vec{g_1}^{2r_2r_3} \odot \vec{g_2}^{r_2s_3+s_2r_3} \odot \vec{g_3}^{r_2t_3+t_2r_3} \odot \iota(\theta_2)^{r_3}, \\
&\quad \iota(\Omega \cdot \theta_3)^{s_2} \odot \vec{g_1}^{r_3s_2+r_2s_3} \odot \vec{g_2}^{2s_2s_3} \odot \vec{g_3}^{t_3s_2+t_2s_3} \odot \iota(\theta_2)^{s_3}, \\
&\quad \iota(\Omega \cdot \theta_3)^{t_2} \odot \vec{g_1}^{r_3t_2+r_2t_3} \odot \vec{g_2}^{s_3t_2+s_2t_3} \odot \vec{g_3}^{2t_2t_3} \odot \iota(\theta_2)^{t_3}\Big)
\end{aligned}$$

$$\begin{aligned}
\pi_3 &= (\pi_{3,1}, \pi_{3,2}, \pi_{3,3}) \\
&= (g^{r_4} \cdot u_0^{-r_3} \cdot G_v(\mathsf{m})^{-r_5},\ g^{s_4} \cdot u_0^{-s_3} \cdot G_v(\mathsf{m})^{-s_5},\ g^{t_4} \cdot u_0^{-t_3} \cdot G_v(\mathsf{m})^{-t_5}) \\
\pi_4 &= (\pi_{4,1}, \pi_{4,2}, \pi_{4,3}) \\
&= (g^{r_6} \cdot h_1^{-r_8} \cdot h_2^{-r_9},\ g^{s_6} \cdot h_1^{-s_8} \cdot h_2^{-s_9},\ g^{t_6} \cdot h_1^{-t_8} \cdot h_2^{-t_9}) \\
\pi_5 &= (\pi_{5,1}, \pi_{5,2}, \pi_{5,3}) \\
&= (g^{r_7} \cdot h_3^{-r_8} \cdot h_4^{-r_9},\ g^{s_7} \cdot h_3^{-s_8} \cdot h_4^{-s_9},\ g^{t_7} \cdot h_3^{-t_8} \cdot h_4^{-t_9})
\end{aligned}$$

$$\begin{aligned}
\pi_6 &= \big(\vec{\pi}_{6,1}, \vec{\pi}_{6,2}, \vec{\pi}_{6,3}\big) \\
&= \Big(\vec{\varphi}^{r_8\delta_1} \odot \iota(X)^{r_{10}} \odot \vec{g_1}^{r_8r_{10}} \odot \vec{g_2}^{s_8r_{10}} \odot \vec{g_3}^{t_8r_{10}}, \\
&\quad \vec{\varphi}^{s_8\delta_1} \odot \iota(X)^{s_{10}} \odot \vec{g_1}^{r_8s_{10}} \odot \vec{g_2}^{s_8s_{10}} \odot \vec{g_3}^{t_8s_{10}},\ \vec{\varphi}^{t_8\delta_1}\Big) \\
\pi_7 &= \big(\vec{\pi}_{7,1}, \vec{\pi}_{7,2}, \vec{\pi}_{7,3}\big) \\
&= \Big(\vec{\varphi}^{r_9\delta_2} \odot \iota(X)^{r_{11}} \odot \vec{g_1}^{r_9r_{11}} \odot \vec{g_2}^{s_9r_{11}} \odot \vec{g_3}^{t_9r_{11}}, \\
&\quad \vec{\varphi}^{s_9\delta_2} \odot \iota(X)^{s_{11}} \odot \vec{g_1}^{r_9s_{11}} \odot \vec{g_2}^{s_9s_{11}} \odot \vec{g_3}^{t_9s_{11}},\ \vec{\varphi}^{t_9\delta_2}\Big) \\
\pi_8 &= (\pi_{8,1}, \pi_{8,2}) = (g^{r_{10}+r_{11}}, g^{s_{10}+s_{11}})
\end{aligned}$$

To obtain perfectly witness-indistinguishable proofs with a simulated common reference string, the signer needs to randomize $\pi_1, \pi_2, \pi_6, \pi_7$ (*i.e.*, that relate to quadratic pairing product equations) and make them uniform in the space of proofs satisfying verification equations 1-2 and 6-7.

D Proofs of Lemmas 1 and 2

D.1 Proof of Lemma 1

The proof is similar to the one of lemma A.1 in [20]. The simulator $\mathcal{B}$ is given a HSDH instance consisting of $(g, \Omega = g^{\omega}, u) \in \mathbb{G}^3$ and triples $\{A_i = g^{1/(\omega+s_{\mathsf{ID}_i})}, B_i = g^{s_{\mathsf{ID}_i}}, C_i = u^{s_{\mathsf{ID}_i}}\}_{i=1,\ldots,\ell}$, with

$\ell = \ell_a + \ell_p$, in $\mathbb{G}$. Before preparing the simulation, $\mathcal{B}$ makes a guess $d_{mode} \xleftarrow{\$} \{0,1\}$ as to whether $\mathcal{A}$ will produce a Type I-A or Type I-B forgery.

Type I-A forgeries. In the expectation of a Type I-A forgery ($d_{mode} = 0$), algorithm $\mathcal{B}$ picks $\rho_{u,1}, \rho_0, \ldots, \rho_n \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $v_i = g^{\rho_i}$, for $i = 0, \ldots, n$, and $u_1 = g^{\rho_{u,1}}$. It also defines $u_0 = u \in \mathbb{G}$ and $h_i = g^{\gamma_i} \in \mathbb{G}$ for $i = 0, \ldots, 4$ using randomly drawn $\gamma_0, \ldots, \gamma_4 \xleftarrow{\$} \mathbb{Z}_p^*$. It finally chooses $\alpha_1, \alpha_2 \xleftarrow{\$} \mathbb{Z}_p^*$, $\xi_1, \xi_2 \xleftarrow{\$} \mathbb{Z}_p$, vector sets $\mathbf{g} = (\vec{g_1}, \vec{g_2}, \vec{g_3})$ and $\mathbf{f} = (\vec{f}, \vec{f_1}, \vec{f_2})$ so that $\vec{g_1} = (g^{\alpha_1}, 1, g)$, $\vec{g_2} = (1, g^{\alpha_2}, g)$ and $\vec{g_3} = \vec{g_1}^{\xi_1} \cdot \vec{g_2}^{\xi_2}$ while $\mathbf{f}$ spans $\mathbb{G}^3$. It defines the group public key as

$$\mathcal{Y} := \Big(g,\ \{h_i\}_{i=0,\ldots,4},\ \Omega,\ u_0,\ u_1,\ \{v_i\}_{i=0,\ldots,n},\ \mathbf{g},\ \mathbf{f}\Big).$$

Then, $\mathcal{B}$ starts interacting with $\mathcal{A}$ and initializes ctr_a, ctr'_a and ctr_p to 0.

- $Q_{\mathsf{a\text{-}join}}$-queries: when $\mathcal{A}$ wants to introduce a malicious user in the group, he triggers an execution of the join protocol with $\mathcal{B}$ acting as the issuer. Then, $\mathcal{B}$ increments ctr'_a, chooses $x, y \xleftarrow{\$} \mathbb{Z}_p^*$ and simulates $\mathcal{A}$'s view (using the black-box simulation technique of theorem 3 in [33]) in step 1 of the protocol to make it end up with the public value $X = g^x$ (so that the new user's membership secret becomes x). Next, $\mathcal{B}$ uses $(A_{ctr}, B_{ctr}, C_{ctr})$, with $ctr = ctr'_a + ctr_p$, and generates a certificate $(K_1, K_2, K_3, K_4) = (A_{ctr}^{(\gamma_0 + x\gamma_1 + y\gamma_2)}, A_{ctr}, B_{ctr}, C_{ctr})$. Parts K_1, K_2, K_3 are sent to $\mathcal{A}$ along with $y \in \mathbb{Z}_p^*$. When $\mathcal{A}$ responds with a signature on $X||K_1||K_2||K_3||g^y$, $\mathcal{B}$ checks that the latter is valid. If so, $\mathcal{B}$ provides $\mathcal{A}$ with the final part K_4 of the membership certificate, increments ctr_a, adds the current value of $\mathsf{N} = ctr_a + ctr_p$ in U^a and stores $(\mathsf{N}, \mathsf{transcript_N})$ in $\mathsf{transcripts}$.
- $Q_{\mathsf{p\text{-}join}}$-queries: when $\mathcal{A}$ asks $\mathcal{B}$ to introduce a new honest user, $\mathcal{B}$ increments ctr_p and proceeds in the same way as with $Q_{\mathsf{a\text{-}join}}$-queries. The only difference is that it executes the join protocol in private and does not have the simulate the view of a malicious user in the first step. As previously, it uses the ctr^{th} triple $(A_{ctr}, B_{ctr}, C_{ctr})$, with $ctr = ctr'_a + ctr_p$, of the HSDH input to compute $K_1, \ldots, K_4$. Also, $\mathcal{B}$ stores the index $\mathsf{N} = ctr_a + ctr_p$ of the new user in U^p and the entry $(\mathsf{N}, \mathsf{transcript_N})$ in $\mathsf{transcripts}$. The latter transcript is used to consistently answer subsequent signing queries that involve the new user.
- $Q_{\mathcal{Y}}$-queries: upon $\mathcal{A}$'s request, $\mathcal{B}$ sends him the public key $\mathcal{Y}$ and the current number of users $\mathsf{N} = ctr_a + ctr_p$ in the group.
- Q_{sig}-queries: to answer signing queries involving honest group members (in U^p), $\mathcal{B}$ simply runs the signing algorithm according the its specification using the (known) membership certificate and the membership secret.
- $Q_{\mathsf{reveal}}(i)$-queries: at any time, $\mathcal{A}$ may also ask for the tracing trapdoor of any user $i \in U^p$ (he already knows those of users in U^a). The simulator $\mathcal{B}$ can always answer such queries since it chose values $(X = g^x, y)$ itself when answering the matching $Q_{\mathsf{p\text{-}join}}$ query.

When $\mathcal{A}$ outputs a Type I forgery $\sigma^\star = (T_1^\star, T_2^\star, T_3^\star, \vec{\sigma_1}^\star, \ldots, \vec{\sigma_{11}}^\star, \pi_1^\star, \ldots, \pi_8^\star)$ for some message $M^\star$, $\mathcal{B}$ computes $\mathsf{m}^\star = m_1 \ldots m_n = \mathcal{H}(M^\star||T_1^\star||T_2^\star||T_3^\star)$. It uses α_1, α_2 to compute a BBS decryption $\theta_i^\star = \sigma_{i,3}{}^\star \cdot \sigma_{i,1}^\star{}^{-1/\alpha_1} \cdot \sigma_{i,2}^\star{}^{-1/\alpha_2}$, where $\vec{\sigma_i}^\star = (\sigma_{i,1}^\star, \sigma_{i,2}^\star, \sigma_{3,i}^\star)$, for $i = 1, \ldots, 9$. The perfect soundness of the proof system guarantees that

$$\theta_2^\star = g^{\frac{1}{\omega + s_{\mathsf{ID}^\star}}}, \qquad \theta_3^\star = g^{s_{\mathsf{ID}^\star}}, \qquad \theta_4^\star = u^{s_{\mathsf{ID}^\star}} \cdot u_1^{x\delta_1} \cdot \big(v_0 \cdot \prod_{j=1}^{n} v_j^{m_j}\big)^{r_s},$$

$$\theta_5^\star = g^{r_s}, \qquad T_1^\star = g^{x\delta_1}$$

for some $s_{\mathsf{ID}^\star} \in \mathbb{Z}_p^*$ (that differs from values $s_{\mathsf{ID}_1}, \ldots, s_{\mathsf{ID}_\ell}$ with overwhelming probability[8]) and $r_s, x, \delta_1 \in \mathbb{Z}_p^*$. Then, $\mathcal{B}$ can extract a HSDH solution as

$$(g^{1/(\omega+s_{\mathsf{ID}^\star})}, g^{s_{\mathsf{ID}^\star}}, u^{s_{\mathsf{ID}^\star}}) = \Big(\theta_2^\star,\ \theta_3^\star,\ \theta_4^\star/(T_1^{\star\rho_{u,1}} \cdot \theta_5^{\star\ \rho_0+\sum_{j=1}^n \rho_j m_j})\Big).$$

Type I-B forgeries. We now consider the case of a Type I-B forgery. When $d_{mode} = 1$, $\mathcal{B}$ initially picks a random $i^\star \xleftarrow{\$} \{1, \ldots, \ell_a\}$. The group public key $\mathcal{Y}$ is generated in the same way. The difference with the case $d_{mode} = 1$ lies in the treatment of $Q_{\mathsf{a\text{-}join}}$ queries: at the $i^{\star\text{th}}$ such query (when $ctr'_a \neq i^\star$, $\mathcal{B}$ behaves as when $d_{mode} = 0$), $\mathcal{B}$ conducts the first step as in the case $d_{mode} = 0$ but the second step is run in a different way. It constructs (K_1, K_2, K_3) by picking $t \xleftarrow{\$} \mathbb{Z}_p^*$, and defining

$$K_1 = g^{\frac{\gamma_0+x\gamma_1+y\gamma_2}{t}} \qquad K_2 = g^{1/t} \qquad K_3 = g^t \cdot \Omega^{-1}.$$

This implicitly defines $s_{\mathsf{ID}} \in \mathbb{Z}_p^*$ to be $t-\omega$. If $\mathcal{A}$, acting as the malicious user, does not fail to respond as specified by the final step of the protocol, $\mathcal{B}$ aborts. Otherwise, the simulation continues. If $\mathcal{A}$ eventually outputs a forgery for which the BBS decryption of $\vec{\sigma_3}^\star$ is $K_3 = g^t \cdot \Omega^{-1}$ (which occurs with probability $1/\ell_a$ in a Type I-B forgery), the BBS decryption $\theta_4^\star$ of $\vec{\sigma_4}^\star$ must be $u^{t-\omega} \cdot u_1^{x\delta_1} \cdot G_v(\mathsf{m})^r$. Along with $T_1^\star = g^{x\delta_1}$, $\rho_{u,1} = \log_g(u_1)$ and the decryption g^r of $\vec{\sigma_5}^\star$, $\theta_4^\star$ must provide $\mathcal{B}$ with a new ℓ-HSDH triple $(g^{1/t}, g^{t-\omega}, u^{t-\omega})$.

To conclude the proof, the actual kind of Type I forgery will match the value of d_{mode} with probability $1/2$ since d_{mode} is independent of $\mathcal{A}$'s view. □

D.2 Proof of Lemma 2

The proof is based on lemma A.2 in the security proof of the Boyen-Waters group signature [20]. As in the latter, the simulator $\mathcal{B}$ receives as input a ℓ_a-HSDH instance comprising elements $(g, \Omega = g^\omega, u) \in \mathbb{G}^3$ as well as a set of triples $\{A_i = g^{1/(\omega+s_{\mathsf{ID}_i})}, B_i = g^{s_{\mathsf{ID}_i}}, C_i = u^{s_{\mathsf{ID}_i}}\}_{i=1,\ldots,\ell_a}$.

To prepare the public key $\mathcal{Y}$, it picks a random index $k \xleftarrow{\$} \{0, \ldots, n\}$, exponents $\rho_{u,1}, \rho_0, \ldots, \rho_n \xleftarrow{\$} \mathbb{Z}_p^*$ and integers $\beta_0, \ldots, \beta_n \xleftarrow{\$} \{0, \ldots, 2\ell_s - 1\}$. It sets $v_0 = u^{\beta_0 - 2k\ell_s} \cdot g^{\rho_0}$, $v_i = u^{\beta_i} \cdot g^{\rho_i}$ for $i = 1, \ldots, n$ and $u_1 = g^{\rho_{u,1}}$. It also defines $u_0 = u \in \mathbb{G}$ and $h_i = g^{\gamma_i} \in \mathbb{G}$ for $i = 0, \ldots, 4$ using random $\gamma_0, \ldots, \gamma_4 \xleftarrow{\$} \mathbb{Z}_p^*$ and finally chooses vector sets $\mathbf{g}$, $\mathbf{f}$ as specified by the setup algorithm.

Before starting its interaction with the Type II forger $\mathcal{A}$, $\mathcal{B}$ initializes counters ctr_a, ctr'_a and ctr_p to 0. These will account for the number of users in U^a, the number of $Q_{\mathsf{a\text{-}join}}$-queries so far, and the number of users in U^p, respectively.

- $Q_{\mathsf{a\text{-}join}}$-queries: adversarial-join queries are handled as in the proof of lemma 1. Namely, $\mathcal{B}$ increments ctr'_a, chooses random values $x, y \xleftarrow{\$} \mathbb{Z}_p^*$ and simulates $\mathcal{A}$'s view in the first step of the protocol so as to force the new user's membership secret to become x. To generate $\mathcal{A}$'s membership certificate, $\mathcal{B}$ uses the next unused triple $(A_{ctr'_a}, B_{ctr'_a}, C_{ctr'_a})$ to produce K_1, K_2, K_3, K_4. If the join protocol successfully terminates, $\mathcal{B}$ increments ctr_a, stores a record $(\mathsf{N}, \mathsf{transcript}_\mathsf{N})$, with $\mathsf{N} = ctr_a + ctr_p$, containing the interaction transcript in $\mathsf{transcripts}$ and adds index N in the set U^a.

[8] With negligible probability smaller than ℓ/p, it may happen that $s_{\mathsf{ID}^\star}$ collides with some s_{ID_i} that $\mathcal{B}$ did not use.

- $Q_{\mathsf{p\text{-}join}}$-queries: when a new honest user is requested to privately join the system, $\mathcal{B}$ increments both ctr and ctr_p and sets $\mathsf{N} = ctr_a + ctr_p$. Then, $\mathcal{B}$ chooses $x, y \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$. It also randomly selects $t_{\mathsf{N}} \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and calculates

$$\begin{aligned} K_1 &= g^{(\gamma_0 + x\gamma_1 + y\gamma_2)/t_{\mathsf{N}}}, & K_2 &= g^{1/t_{\mathsf{N}}}, \\ K_3 &= g^{t_{\mathsf{N}}} \cdot \Omega^{-1}, & K_4 &= \star, \end{aligned}$$

  where $\star$ is a placeholder for an unknown group element. This implicitly defines $s_{\mathsf{ID}} = \log_g(K_3)$ to be $t_{\mathsf{N}} - \omega$. The value N is added to U^p while (K_1, K_2, K_3, K_4, y) is stored as part of $\mathsf{transcripts}_{\mathsf{N}}$. In the expectation of future signing queries involving the user, $\mathcal{B}$ retains $\mathsf{sec}_{\mathsf{N}} = x$ and t_{N}.
- $Q_{\mathcal{Y}}$ and $Q_{\mathsf{reveal}}(i)$-queries: are treated as in the proof of lemma 1 and the simulator always knows the values requested by $\mathcal{A}$.
- Q_{sig}-queries: when $\mathcal{A}$ requests user $i \in U^p$ (with $1 \leq i \leq \mathsf{N}$) to sign message M, $\mathcal{B}$ first retrieves the tuple (K_1, K_2, K_3, K_4, y) from $\mathsf{transcript}_i$ and the previously stored $\mathsf{sec}_i = x$ and $t_i \in \mathbb{Z}_p^*$. Since $i \in U^p$, $K_4 = \star$ is not available and $\mathcal{B}$ must simulate knowing it. To do so, it begins by generating a triple $(T_1 = g^{x\delta_1}, T_2 = g^{y\delta_2}, T_3 = g^{\delta_1+\delta_2})$ which is hashed along with M to obtain $\mathsf{m} = m_1 \ldots m_n = \mathcal{H}(M||T_1||T_2||T_3)$. At this stage, it is convenient to write $G_v(\mathsf{m}) = v_0 \cdot \prod_{j=1}^n v_j^{m_j}$ as $u^{J_v(\mathsf{m})} \cdot g^{K_v(\mathsf{m})}$ where $J_v(\mathsf{m}) = \beta_0 - 2k\ell_s + \sum_{j=1}^n \beta_j m_j$, $K_v(\mathsf{m}) = \rho_0 + \sum_{j=1}^n \rho_j m_j$. If $J_v(\cdot)$ is zero in $\mathbb{Z}_p^*$, $\mathcal{B}$ aborts. Otherwise, it can pick $r_s \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ and compute a pair

$$\Big(\theta_4 = u^{t_i} \cdot g^{x \cdot \delta_1 \cdot \rho_{u,1}} \cdot \Omega^{\frac{K_v(\mathsf{m})}{J_v(\mathsf{m})}} \cdot G_v(\mathsf{m})^{r_s},\ \theta_5 = g^{r_s} \cdot \Omega^{\frac{1}{J_v(\mathsf{m})}}\Big),$$

  which can be re-written as $(\theta_4 = u^{t_i - \omega} \cdot u_1^{x \cdot \delta_1} \cdot G_v(\mathsf{m})^{\tilde{r_s}}, \theta_5 = g^{\tilde{r_s}})$ if one implicitly defines $\tilde{r_s} = r_s + \omega/J_v(\mathsf{m})$. This pair has the required distribution and, together with (K_1, K_2, K_3), allows generating an appropriately anonymized signature.

At the end of the game, the adversary $\mathcal{A}$ outputs a message $M^\star$ together with a Type II forgery $(T_1^\star, T_2^\star, T_3^\star, \vec{\sigma_1}^\star, \ldots, \vec{\sigma_{11}}^\star, \pi_1^\star, \ldots, \pi_8^\star)$. By assumption, $\vec{\sigma_3}^\star$ must be a BBS encryption of a value K_3 appearing in the transcript of some user $i^\star \in U^p$. Then, $\mathcal{B}$ computes the bitstring $\mathsf{m}^\star = m_1 \ldots m_n$. As in the proof of lemma 1, it performs BBS decryptions $\theta_i^\star = \sigma_{i,3} \cdot {\sigma_{i,1}^\star}^{-1/\alpha_1} \cdot {\sigma_{i,2}^\star}^{-1/\alpha_2}$ for $i = 1, \ldots, 9$. By the perfect soundness of the proof system, these values must satisfy

$$\begin{aligned} \theta_2^\star &= g^{1/(\omega + s_{\mathsf{ID}_{i^\star}})} & \theta_3^\star &= g^{s_{\mathsf{ID}_{i^\star}}} & \theta_4^\star &= u^{s_{\mathsf{ID}_{i^\star}}} \cdot u_1^{x\delta_1} \cdot G_v(\mathsf{m}^\star)^{r_s} \\ \theta_5^\star &= g^{r_s} & T_1^\star &= g^{x\delta_1}, \end{aligned}$$

for some $r_s, x, \delta_1 \in \mathbb{Z}_p^*$, and where $G_v(\mathsf{m}^\star) = v_0 \cdot \prod_{j=1}^n v_j^{m_j} = u^{J_v(\mathsf{m}^\star)} \cdot g^{K_v(\mathsf{m}^\star)}$ and $s_{\mathsf{ID}_{i^\star}} = t_{i^\star} - \omega$. Then, the simulator $\mathcal{B}$ aborts if $J_v(\mathsf{m}^\star) = \beta_0 + \sum_{j=1}^n \beta_j m_j - 2k\ell_s \neq 0$. Otherwise, $\mathcal{B}$ can compute $u^{s_{\mathsf{ID}_{i^*}}} = \theta_4^\star / (\theta_5^{\star K_v(\mathsf{m}^\star)} \cdot T_1^{\star \rho_{u,1}})$, which yields a full tuple $\big(g^{1/(\omega + s_{\mathsf{ID}_{i^\star}})},\ g^{s_{\mathsf{ID}_{i^\star}}},\ u^{s_{\mathsf{ID}_{i^\star}}}\big)$ such that the underlying $s_{\mathsf{ID}_{i^\star}} = t_{i^\star} - \omega$ differs from $s_{\mathsf{ID}_1}, \ldots, s_{\mathsf{ID}_{\ell_a}}$ with probability at least $1 - \ell_a/p$ (since the value $t_{i^\star}$ was chosen at random when answering a $Q_{\mathsf{p\text{-}join}}$-query).

To assess $\mathcal{B}$'s probability not to abort throughout the simulation, we can proceed as in [45, 20]. Namely, one can show that $J_v(\mathsf{m}) \neq 0$ in all signing queries with probability $\geq 1/2$. Conditioned on the event that $\mathcal{B}$ does not abort before the forgery stage, the probability to have $J_v(\mathsf{m}^\star) = 0$ is then shown to be at least $1/(2n\ell_s)$ (see [45] for a detailed probabilistic analysis). □

E Proof of Lemma 5

For convenience, we use the same variant of the CDH problem as in the treatment of framing attacks of Type I-B (in the proof of lemma 4). Namely, if the adversary is able to claim a signature of the same equivalence class as one created by an honest signer, we show how to compute $g^{b/a}$ given $(A = g^a, B = g^b)$.

We also consider two kinds of Type II attacks: Type II-A attacks are those for which step 6 is executed (which means that the condition of step 5 is not fulfilled) in the experiment of definition 5 whereas, in Type II-B attacks, the experiment returns 1 and halts at step 5.

As in the proof of lemma 4, the simulator has to guess whether $\mathcal{A}$ will be a Type II-A or a Type II-B attacker beforehand. To this end, he flips a fair binary coin at the very beginning of the game and prepares the public key accordingly.

Type II-A attacks. Let us first consider Type II-A attacks and assume an adversary $\mathcal{A}$ for which step 6 is reached in the non-frameability experiment (the case of the experiment returning 1 at step 5 will be easier to explain then). We outline a simulator $\mathcal{B}$ that uses $\mathcal{A}$ to solve a CDH instance $(A = g^a, B = g^b)$. Namely, $\mathcal{B}$ prepares $\mathcal{Y}$ as in the proof of security against Type I-B attacks. In particular, it knows discrete logarithms $\omega = \log_g(\Omega)$, $\alpha_1 = \log_g(g_1)$, $\alpha_2 = \log_g(g_2)$, $\rho_{u,1} = \log_g(u_1)$ and $\gamma_i = \log_g(h_i)$ for $i = 0$ to 4.

It also defines $f = B$ and, as in previous lemmas, it randomly chooses a vector of group elements $\overline{f} = (f_0, f_1, \ldots, f_n) \in \mathbb{G}^n$ using the technique of [45] in such a way that, for any $\mathsf{m}_c \in \{0,1\}^n$, it holds that $G_f(\mathsf{m}_c) = f^{J(\mathsf{m}_c)} \cdot A^{K(\mathsf{m}_c)}$, for some integer-valued functions $J(.), K(.)$ such that $J(.)$ is small in absolute value and cancels with probability $O(1/(n+1) \cdot \ell_c)$. Throughout the game, $\mathcal{B}$ interacts with $\mathcal{A}$ as follows:

- $Q_{\mathcal{S}}$ and $Q_{\mathcal{Y}}$-queries: are handled as when dealing with Type I forgeries.
- $Q_{\mathsf{b\text{-}join}}$-queries: at the i^{th} such query, $\mathcal{B}$ runs the join protocol and plays the role of the prospective honest group member. It picks $x_i \xleftarrow{\$} \mathbb{Z}_p^*$, sets $X_i = A^{x_i} \in \mathbb{G}$ and simulates $\mathcal{A}$'s view in such a way that the public value becomes X_i (and the underlying membership secret happens to be $\mathsf{sec}_i = ax_i$). Other steps of the join protocol are conducted as previously.
- Q_{sig}-queries: at the j^{th} Q_{sig}-query involving user $i \in U^b$, $\mathcal{B}$ retrieves the previously stored element $X_i = A^{x_i}$ and generates signature components $(\theta_1, \theta_2, \theta_3)$ using the membership certificate $(K_1, K_2, K_3, K_4, y_i)$. It then chooses $\delta_{1,ij}, \mu_{ij} \xleftarrow{\$} \mathbb{Z}_p^*$ and calculates $\theta_4 = K_4 \cdot X_i^{\rho_{u,1}\delta_{1,ij}} \cdot G_v(\mathsf{m})^{r_s}$, $\theta_5 = g^{r_s}$ for a random $r_s \xleftarrow{\$} \mathbb{Z}_p^*$. It finally computes traceability values (T_1, T_2, T_3) as

$$T_1 = X_i^{\delta_{1,ij}} = A^{x_i\delta_{1,ij}}, \qquad T_2 = A^{y_i\mu_{ij}}, \qquad T_3 = g^{\delta_{1,ij}} \cdot A^{\mu_{ij}},$$

  which implicitly defines $\tilde{\delta}_{2,ij} = a\mu_{ij}$ in such a way that $T_2 = g^{y_i\tilde{\delta}_{2,ij}}$ and $T_3 = g^{\delta_{1,ij}+\tilde{\delta}_{2,ij}}$. Signature components (θ_6, θ_7) are computable as $(X_i^{\gamma_1} \cdot h_2^{y_i}, X_i^{\gamma_3} \cdot h_2^{y_i})$. Also, since $\mathcal{B}$ knows $\alpha_1, \alpha_2, \xi_1, \xi_2$ such that $\vec{\varphi} = (g^{\alpha_1\xi_1}, g^{\alpha_2\xi_2}, g^{\xi_1+\xi_2+1})$, it is able to compute the commitment to $\tilde{\delta}_{2,ij}$ (*i.e.*, $\vec{\sigma_{11}} = \vec{\varphi}^{\tilde{\delta_{2,ij}}} \odot \vec{g_1}^{r_{11}} \odot \vec{g_2}^{s_{11}}$, using random exponents $r_{11}, s_{11} \xleftarrow{\$} \mathbb{Z}_p^*$) and the proof element π_7 (as described in appendix C).
- Q_{claim}-queries: when user $i \in U^b$ is required to claim a message-signature pair (M, σ), where $\sigma = (T_1, T_2, T_3, \vec{\sigma_1}, \ldots, \vec{\sigma_{11}}, \pi_1, \ldots, \pi_8)$, $\mathcal{B}$ returns $\perp$ if $i \notin U^b$ or if user i did not previously create a message-signature pair involving (M, T_1, T_2, T_3). Otherwise, $\mathcal{B}$ recalls the value $x_i, y_i \in \mathbb{Z}_p$

such that $X_i = A^{x_i}$ and $\mathsf{cert}_i = (K_{1,i}, K_{2,i}, K_{3,i}, K_{4,i}, y_i)$ as well as the scalars $\delta_{1,ij}, \mu_{ij} \in_R \mathbb{Z}_p$ that were used to define $T_1 = A^{x_i\delta_{1,ij}}$, $T_2 = A^{y_i\mu_{ij}}$ and $T_3 = g^{\delta_{1,ij}} \cdot A^{\mu_{ij}}$. Then, $\mathcal{B}$ computes $\mathsf{m}_c = \mathcal{H}(M||T_1||T_2||T_3||\mathsf{upk}[i]) \in \{0,1\}^n$, where $\mathsf{upk}[i]$ is the previously defined long term key of user i, and evaluates the functions $J(\mathsf{m}_c)$ and $K(\mathsf{m}_c)$ such that $G_f(\mathsf{m}_c) = f^{J(\mathsf{m}_c)} \cdot A^{K(\mathsf{m}_c)}$. If $J(\mathsf{m}_c) = 0$, $\mathcal{B}$ halts and reports failure. Otherwise, the first part $(D_{x,1}, D_{x,2})$ of the claim can be obtained by choosing $r_x \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ and computing

$$(D_{x,1}, D_{x,2}) = \big(G_f(\mathsf{m}_c)^{r_x} \cdot g^{-x_i K(\mathsf{m}_c)/J(\mathsf{m}_c)},\ T_1^{r_x} \cdot g^{-\delta_{1,ij}/J(\mathsf{m}_c)}\big),$$

which is well-distributed since it can be written $(D_{x,1}, D_{x,2}) = (f^{1/(ax_i)} \cdot G_f(\mathsf{m}_c)^{\tilde{r}_x},\ T_1^{\tilde{r}_x})$ if we implicitly define $\tilde{r}_x = r_x - \frac{1}{ax_iJ(\mathsf{m}_c)}$. As for the second part $(D_{y,1}, D_{y,2})$ of the claim, $\mathcal{B}$ can compute it as specified by the claiming algorithm since it knows $y_i \in \mathbb{Z}_p$. The tuple $(D_{x,1}, D_{x,2}, D_{y,1}, D_{y,2})$ can then be signed using the private key $\mathsf{usk}[i]$ of user i.

The game ends with $\mathcal{A}$ outputting a message $M^\star$ and a triple $(\sigma^\star, \tau^\star, \mathsf{upk}^\star)$ such that $\sigma^\star$ opens or traces to $i^\star \in U^b$, $\mathsf{Claim\text{-}Verify}(M^\star, \sigma^\star, \tau^\star, \mathsf{upk}^\star, \mathcal{Y}) = 1$ and user $i^\star$ produced a message-signature pair $(M^\star, \sigma^\star)$ with the same traceability values $(T_1^\star, T_2^\star, T_3^\star)$. In addition, $(M^\star, \sigma^\star)$ may have been claimed by the legitimate signer (holding a long term key $\mathsf{upk}[i^\star]$) upon adversarial request. By assumption, however, the claim $\tau^\star = (D_{x,1}^\star, D_{x,2}^\star, D_{y,1}^\star, D_{y,2}^\star, csig^\star)$ must be valid and pertain to a public key $\mathsf{upk}^\star$ such that $\mathsf{upk}^\star \neq \mathsf{upk}[i^\star]$.

At this stage, $\mathcal{B}$ computes $\mathsf{m}_c^\star = \mathcal{H}(M^\star||T_1^\star||T_2^\star||T_3^\star||\mathsf{upk}^\star) \in \{0,1\}^n$ and evaluates the functions $J(\mathsf{m}_c^\star)$ and $K(\mathsf{m}_c^\star)$. It fails if $J(\mathsf{m}_c^\star) \neq 0$ or if it happens that $\mathsf{m}_c^\star = \mathcal{H}(M'||T_1'||T_2'||T_3'||\mathsf{upk}')$ for some $(M', T_1', T_2', T_3', \mathsf{upk}') \neq (M^\star, T_1^\star, T_2^\star, T_3^\star, \mathsf{upk}^\star)$ that was previously involved in Q_{claim}-query. Otherwise, since we have $G(\mathsf{m}_c^\star) = A^{K(\mathsf{m}_c^\star)}$ and since $\tau^\star$ satisfies the verification test of equation (7), it must hold that

$$\frac{e\big(D_{x,1}^\star, A^{x_i\delta_{1,ij}}\big) \cdot e\big(D_{y,1}^\star, A^{y_i\mu_{ij}}\big)}{e\big(A^{K(\mathsf{m}_c^\star)}, D_{x,2}^\star \cdot D_{y,2}^\star\big)} = e(B, g^{\delta_{1,ij}} \cdot A^{\mu_{ij}}). \tag{10}$$

If we now re-arrange terms in the above equation, we find

$$e\big(A, {D_{x,1}^\star}^{x_i} \cdot {D_{y,1}^\star}^{y_i\mu_{ij}/\delta_{1,ij}} \cdot (D_{x,2}^\star \cdot D_{y,2}^\star)^{-K(\mathsf{m}_c^\star)/\delta_{1,ij}} \cdot B^{-\mu_{ij}/\delta_{1,ij}}\big) = e(g, B),$$

which implies that

$${D_{x,1}^\star}^{x_i} \cdot {D_{y,1}^\star}^{y_i\mu_{ij}/\delta_{1,ij}} \cdot (D_{x,2}^\star \cdot D_{y,2}^\star)^{-K(\mathsf{m}_c^\star)/\delta_{1,ij}} \cdot B^{-\mu_{ij}/\delta_{1,ij}} = g^{b/a}$$

is computable by $\mathcal{B}$. If the hash function $\mathcal{H}$ is collision-resistant, a sufficient condition for $g^{b/a}$ to be computable is to have $J(\mathsf{m}_c) \neq 0$ in each Q_{claim}-query and $J(\mathsf{m}_c^\star) = 0$ in the adversary's output $(M^\star, \sigma^\star, \tau^\star)$. As in previous lemmas, known results [37, 45] on "programmable" hash functions tell us that this condition is satisfied with probability $O(1/4n\ell_c)$.

Type II-B attacks. We are left with the case of Type II-B attacks where the experiment of definition 5 returns 1 at step 5 and does not reach step 6. In such a situation, the adversary breaks either the security (*i.e.*, the existential unforgeability under chosen-message attack) of the ordinary signature scheme used by group members or the Diffie-Hellman assumption. Namely, when the adversary outputs $(M^\star, \sigma^\star, \tau^\star)$, where $\sigma^\star$ contains $(T_1^\star, T_2^\star, T_3^\star)$ and with $\tau^\star = (D_{x,1}^\star, D_{x,2}^\star, D_{y,1}^\star, D_{y,2}^\star)$, two situations can be distinguished.

- The simulator $\mathcal{B}$ did not sign the tuple $(D^{\star}_{x,1}, D^{\star}_{x,2}, D^{\star}_{y,1}, D^{\star}_{y,2})$ using member i's signing key $\mathsf{usk}[i]$ when answering Q_{claim}-queries. In this case, $\mathcal{A}$ is necessarily able to break the unforgeability of the ordinary signature scheme (we omit the proof which is straightforward).
- $\mathcal{B}$ signed the same tuple $(D^{\star}_{x,1}, D^{\star}_{x,2}, D^{\star}_{y,1}, D^{\star}_{y,2})$ when answering a Q_{claim}-query for another signature involving a tuple $(M, T_1, T_2, T_3) \neq (M^{\star}, T^{\star}_1, T^{\star}_2, T^{\star}_3)$. In this case, $\mathcal{B}$ can either find a collision for $\mathcal{H}$ or solve a Diffie-Hellman instance $(A = g^a, B = g^b)$ by proceeding exactly as in the case where the experiment reaches step 6. To this end, $\mathcal{B}$ can prepare the public key $\mathcal{Y}$ exactly in the same way and obtain $g^{b/a}$ with the same probability $O(1/4n\ell_c)$.

□