Efficient Distributed Tag-Based Encryption and its Application to Group Signatures with Efficient Distributed Traceability

Essam Ghadafi

Computer Science Department, University of Bristol, UK

Abstract. In this work, we first formalize the notion of dynamic group signatures with distributed traceability, where the capability to trace signatures is distributed among n managers without requiring any interaction. This ensures that only the participation of all tracing managers permits tracing a signature, which reduces the trust placed in a single tracing manager. The threshold variant follows easily from our definitions and constructions. Our model offers strong security requirements. Our second contribution is a generic construction for the notion which has a concurrent join protocol, meets strong security requirements, and offers efficient traceability, i.e. without requiring tracing managers to produce expensive zero-knowledge proofs for tracing correctness. To dispense with the expensive zero-knowledge proofs required in the tracing, we deploy a distributed tag-based encryption with public verifiability. Finally, we provide some concrete instantiations, which, to the best of our knowledge, are the first efficient provably secure realizations in the standard model simultaneously offering all the aforementioned properties. To realize our constructions efficiently, we construct an efficient distributed (and threshold) tag-based encryption scheme that works in the efficient Type-III asymmetric bilinear groups. Our distributed tag-based encryption scheme yields short ciphertexts (only 1280 bits at 128-bit security), and is secure under an existing variant of the standard decisional linear assumption. Our tag-based encryption scheme is of independent interest and is useful for many applications beyond the scope of this paper. As a special case of our distributed tag-based encryption scheme, we get an efficient tag-based encryption scheme in Type-III asymmetric bilinear groups that is secure in the standard model.

Keywords. Group signatures, distributed traceability, distributed public-key encryption, standard model.

1 Introduction

Group signatures, introduced by Chaum and van Heyst [25], are a fundamental cryptographic primitive allowing a member of a group (administered by a designated manager) to anonymously sign messages on behalf of the group. In the case of a dispute, a designated tracing manager can revoke anonymity by revealing the signer. The downside of granting a single entity the capability to trace signatures is the high trust placed in such an entity. As a result, anonymity in group signatures relying on a single tracing authority only holds if the tracing authority is fully honest. More precisely, a misbehaving tracing authority could abuse the power granted to it and open signatures need not be opened. Therefore, reducing the trust placed in the tracing manager by distributing the tracing capability among different parties is desirable. While some of the existing schemes can be translated into the distributed traceability setting by utilizing standard secret-sharing techniques, e.g. [10, 50], unfortunately, most of those secure in the strong Bellare et al. model [13], would become impractical due to the expensive zero-knowledge proofs required in the tracing.

Related Work. After their introduction, a long line of research on group signatures has emerged. Bellare, Micciancio and Warinschi [11] formalized the security definitions for group signatures supporting static groups. In such a notion, the group population is fixed at the setup phase. Moreover, the group manager (which also provides the traceability feature) needs to be fully trusted. Later, Bellare, Shi and Zhang [13] provided formal security definitions for the more practical dynamic case where members can enroll at any time. Also, [13] separated the tracing role from the group management.

Besides correctness, the model of [13] defines three other requirements: anonymity, traceability and non-frameability. Informally, anonymity requires that signatures do not reveal the identity of the signer; traceability requires that the tracing manager is always able to identify the signer and prove such a claim; non-frameability ensures that even if the group and tracing managers collude with the rest of the

group, they cannot frame an honest member. More recently, Sakai et al. [49] strengthened the security definitions of group signatures by adding the *opening soundness* requirement. The stronger variant of opening soundness ensures that even if all entities are corrupt, it is infeasible to produce a signature that traces to two different members.

Constructions of group signatures in the random oracle model [12] include [24, 23, 8, 18, 22, 21, 46, 29, 41, 26, 15]. Constructions not relying on random oracles include [7, 35, 19, 36, 20, 5, 43, 44]. Other measures in which the above mentioned constructions differ are: the security they offer, the size of the signatures they yield and the round complexity of the join protocol. For instance, some constructions, e.g. [18–20, 5], only offer a weaker variant of the anonymity requirement (i.e. CPA-Anonymity [18]) where the adversary is not granted access to a tracing oracle. On the other hand, other constructions, e.g. [36, 26], offer full anonymity where in the game the adversary is allowed to ask for any signature except the challenge signature to be traced.

Different approaches have been proposed to minimize the trust placed in the tracing manager. Sakai et al. [48] recently proposed the notion of group signatures with message-dependent opening. In such a notion, an admitter specifies what messages signatures upon which can be traced. This prevents the tracing manager from opening signatures on messages not admitted by the admitter. In [38], the authors informally highlighted how to extend their linkable group signature scheme (secure in the random oracle model) to provide distributed traceability. In [30], the authors presented a scheme where the roles of the managers can be distributed. Their scheme is only non-frameable against honest tracing managers, and requires both random oracles and the generic group model.

Benjumea et al. [14] introduced the notion of fair traceable multi-group signatures which combines the features of group signatures and traceable signatures [40] and in which traceability requires the co-operation of a judge with designated parties known as fairness authorities. The authors also provided a construction of their primitive in the random oracle model.

Zheng et al. [51] extended Manulis's notion of democratic group signatures [45] to add threshold traceability where group members must collude to trace signatures. Democratic group signatures differ from group signatures in many aspects. In the former, the roles of the group and tracing managers are eliminated and the group is managed by the members themselves. In addition, signatures are only anonymous to non-members.

Our Contribution. We offer the following contributions:

1. A formal security model for group signatures with distributed traceability without requiring any interaction. Only the participation of all n tracing managers makes it possible to trace a signature. The more general k out of n threshold case follows easily from our definitions. Our model offers strong security including the notion of tracing soundness [49].
2. A generic framework for constructing group signatures with efficient (i.e. without requiring expensive zero-knowledge proofs in the tracing) distributed traceability that supports dynamic groups with a concurrent join protocol and which is provably secure w.r.t. our strong security model.
3. Instantiations of the generic framework in the standard model. To the best of our knowledge, they are the first provably secure realizations not relying on idealized assumptions offering all the aforementioned properties.
4. An efficient distributed/threshold selective-tag weakly IND-CCA tag-based encryption scheme that is based on an existing variant of the standard decisional linear assumption. Our scheme is non-interactive (i.e. requires no interaction between the decryption servers) and is robust, i.e. the validity of the decryption shares as well as the ciphertext is publicly verifiable. The scheme works in the efficient Type-III bilinear groups setting and yields short ciphertexts which are much shorther than those of the original Kiltz's tag-based encryption scheme [42] and its threshold variant of [6]. By combining our scheme with a strongly unforgeable one-time signature scheme as per the transformation in [42], we obtain an efficient fully secure IND-CCA distributed/threshold encryption scheme, which is useful for many applications beyond the scope of this paper.

Paper Organization. In Section 2, we give some preliminary definitions. We present our model for group signatures with distributed traceability in Section 3. We present the building blocks we use in Section 4. In Section 5, we present our generic construction and provide a proof of its security. In Section 6, we present instantiations in the standard model.

Notation. A function $\nu(.) : \mathbb{N} \rightarrow \mathbb{R}^+$ is negligible in c if for every polynomial $p(.)$ and all sufficiently large values of c, it holds that $\nu(c) < \frac{1}{p(c)}$. Given a probability distribution Y, we denote by $x \leftarrow Y$ the operation of selecting an element according to Y. If M is a probabilistic machine, we denote by $M(x_1, \ldots, x_n)$ the output distribution of M on inputs $(x_1, \ldots, x_n)$. By $[n]$ we denote the set $\{1, \ldots, n\}$. By PPT we mean running in probabilistic polynomial time in the relevant security parameter.

2 Preliminaries

In this section we provide some preliminary definitions.

Bilinear Groups. A bilinear group is a tuple $\mathcal{P} := (\mathbb{G}, \tilde{\mathbb{G}}, \mathbb{T}, p, G, \tilde{G}, e)$ where $\mathbb{G}$, $\tilde{\mathbb{G}}$ and $\mathbb{T}$ are groups of a prime order p, and G and $\tilde{G}$ generate $\mathbb{G}$ and $\tilde{\mathbb{G}}$, respectively. The function e is a non-degenerate bilinear map $e : \mathbb{G} \times \tilde{\mathbb{G}} \longrightarrow \mathbb{T}$. We use multiplicative notation for all the groups. We let $\mathbb{G}^\times := \mathbb{G} \setminus \{1_\mathbb{G}\}$ and $\tilde{\mathbb{G}}^\times := \tilde{\mathbb{G}} \setminus \{1_{\tilde{\mathbb{G}}}\}$. In this paper, we focus on the efficient Type-III setting [32], where $\mathbb{G} \neq \tilde{\mathbb{G}}$ and there is no isomorphism between the groups in either direction. We assume there is an algorithm $\mathsf{BGrpSetup}$ taking as input a security parameter λ and outputting a description of bilinear groups.

Complexity Assumptions. We use the following existing assumptions:

Symmetric External Decisional Diffie-Hellman (SXDH). The Decisional Diffie-Hellman (DDH) assumption holds in both groups $\mathbb{G}$ and $\tilde{\mathbb{G}}$.

Decisional Linear in $\mathbb{G}$ (DLIN$_\mathbb{G}$) Assumption [33, 1]. Given $\mathcal{P}$ and a tuple $(G^h, G^v, G^u, G^{rh}, G^{sv}, G^{ut}) \in \mathbb{G}^6$ for unknown $h, r, s, t, u, v \in \mathbb{Z}_p$, it is hard to determine whether or not $t = r + s$.

External Decisional Linear in $\mathbb{G}$ (XDLIN$_\mathbb{G}$) Assumption [1] [1]. Given $\mathcal{P}$ and a tuple $(G^h, G^v, G^u, G^{rh}, G^{sv}, G^{ut}, \tilde{G}^h, \tilde{G}^v, \tilde{G}^u, \tilde{G}^{rh}, \tilde{G}^{sv}) \in \mathbb{G}^6 \times \tilde{\mathbb{G}}^5$ for unknown $h, r, s, t, u, v \in \mathbb{Z}_p$, it is hard to determine whether or not $t = r + s$.

q-Strong Diffie-Hellman (q-SDH) Assumption in $\mathbb{G}$ [17]. Given the tuple $(G, G^x, \ldots, G^{x^q}) \in \mathbb{G}^{q+1}$ for $x \leftarrow \mathbb{Z}_p$, it is hard to output a pair $(c, G^{\frac{1}{x+c}}) \in \mathbb{Z}_p \times \mathbb{G}$ for an arbitrary $c \in \mathbb{Z}_p \backslash \{-x\}$.

q-AGHO [3]. Given a random tuple $(G, \tilde{G}, \tilde{W}, \tilde{X}, \tilde{Y}) \in \mathbb{G} \times \tilde{\mathbb{G}}^4$, and q uniformly random tuples $(A_i, B_i, R_i, \tilde{D}_i) \in \mathbb{G}^3 \times \tilde{\mathbb{G}}$, each satisfying:

$$e(A_i, \tilde{D}_i) = e(G, \tilde{G})$$
$$e(G, \tilde{X}) = e(A_i, \tilde{W}) e(B_i, \tilde{G}) e(R_i, \tilde{Y}),$$

it is hard to output a new tuple $(A^*, B^*, R^*, \tilde{D}^*)$ satisfying the above equations.

Group Signatures. Here we briefly review the model of Bellare et al. [13] for dynamic group signatures with a single tracing authority. A dynamic group signature scheme consists of the following algorithms:

- $\mathsf{GKg}(1^\lambda)$ outputs a group public key gpk, a group manager's secret key msk and a tracing key tsk.
- $\mathsf{UKg}(1^\lambda)$ outputs a secret/public key pair $(\mathbf{usk}[\mathsf{uid}], \mathbf{upk}[\mathsf{uid}])$ for user uid.
- $\langle \mathsf{Join}(\mathsf{gpk}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}]), \mathsf{Issue}(\mathsf{msk}, \mathsf{uid}, \mathbf{upk}[\mathsf{uid}]) \rangle$ is an interactive protocol between a user uid and the group manager GM via which the user joins the group. If successful, the final state of the Issue algorithm is stored in the registration table at index uid (i.e. $\mathbf{reg}[\mathsf{uid}]$), whereas that of the Join algorithm is stored in $\mathbf{gsk}[\mathsf{uid}]$ and is used as the user's group signing key.
- $\mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], m)$ outputs a group signature Σ on the message m by member uid.

[1] XDLIN$_{\tilde{\mathbb{G}}}$ can be defined analogously by giving $\tilde{G}^{ut} \in \tilde{\mathbb{G}}$ instead of $G^{ut} \in \mathbb{G}$.

- $\mathsf{Verify}(\mathsf{gpk}, m, \Sigma)$ verifies whether or not Σ is a valid group signature on m outputting a bit.
- $\mathsf{Trace}(\mathsf{gpk}, \mathsf{tsk}, m, \Sigma, \mathbf{reg})$ is the tracing algorithm in which the tracing manager uses its tracing key tsk to identify the group member uid who produced the signature Σ plus a proof π_{Trace} for such a claim.
- $\mathsf{TraceVerify}(\mathsf{gpk}, \mathsf{uid}, \pi_{\text{Trace}}, \mathbf{upk}[\mathsf{uid}], m, \Sigma)$ verifies the tracing proof π_{Trace} outputting a bit accordingly.

Besides correctness, the security requirements defined by [13] are:

- *Anonymity:* A group signature does not reveal the identity of the member who produced it even when the keys of the group manager and all group members are all revealed. This requirement relies on the tracing manager being fully honest.
- *Non-Frameability:* Even if the group and tracing managers collude with the rest of the group, they cannot frame an honest group member.
- *Traceability:* Even if the tracing manager and all group members are corrupt, they cannot produce a signature that does not trace to a member of the group.

3 Syntax and Security of Dynamic Group Signatures with Distributed Traceability

The parties involved in a Dynamic Group Signature with Distributed Traceability ($\mathcal{DGSDT}$) are: a group manager GM who authorizes who can join the group; κ tracing managers $\mathsf{TM}_1, \ldots, \mathsf{TM}_\kappa$ which only the participation of all of which makes it possible to identify who produced a signature; a set of users who can join group at any time by contacting the group manager. A $\mathcal{DGSDT}$ scheme consists of the following polynomial-time algorithms:

- $\mathsf{GKg}(1^\lambda, \kappa)$ is run by a trusted third party. On input a security parameter λ and the number of tracing managers κ, it outputs a group public key gpk, a group manager's secret key msk and secret tracing keys $\{\mathsf{tsk}_i\}_{i=1}^{\kappa}$.
- $\mathsf{UKg}(1^\lambda)$ outputs a secret/public key pair $(\mathbf{usk}[\mathsf{uid}], \mathbf{upk}[\mathsf{uid}])$ for user uid. We assume that the public key table $\mathbf{upk}$ is publicly available (possibly via some PKI) so that anyone can obtain authentic copies of the public keys.
- $\langle \mathsf{Join}(\mathsf{gpk}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}]), \mathsf{Issue}(\mathsf{msk}, \mathsf{uid}, \mathbf{upk}[\mathsf{uid}]) \rangle$ is an interactive protocol between a user uid and the group manager GM. Upon successful completion, uid becomes a member of the group. The final state of the Issue algorithm is stored in the registration table at index uid (i.e. $\mathbf{reg}[\mathsf{uid}]$), whereas that of the Join algorithm is stored in $\mathbf{gsk}[\mathsf{uid}]$. We assume that the communication in this interactive protocol takes place over a secure (i.e. private and authentic) channel. The protocol is initiated by a call to Join.
- $\mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], m)$ on input the group public key gpk, a user's group signing key $\mathbf{gsk}[\mathsf{uid}]$ and a message m, outputs a group signature Σ on m by the group member uid.
- $\mathsf{Verify}(\mathsf{gpk}, m, \Sigma)$ is a deterministic algorithm which checks whether or not Σ is a valid group signature on m outputting a bit.
- $\mathsf{TraceShare}(\mathsf{gpk}, \mathsf{tsk}_i, m, \Sigma)$ on input the group public key gpk, a tracing key tsk_i belonging to tracing manager TM_i, a message m and a signature Σ, it outputs $(\nu, \pi_{\text{Trace}})$ where ν is the tracing share of TM_i of Σ and π_{Trace} is a proof for the correctness of the tracing share. If TM_i is unable to compute her share, she outputs $(\perp, \perp)$. If the validity of the shares are publicly verifiable, we just omit π_{Trace}.
- $\mathsf{ShareVerify}(\mathsf{gpk}, \mathsf{tid}, \nu, \pi_{\text{Trace}}, m, \Sigma)$ verifies whether the share ν is a valid tracing share of Σ by tracing manager $\mathsf{TM}_{\mathsf{tid}}$ outputting a bit accordingly.
- $\mathsf{TraceCombine}(\mathsf{gpk}, \{(\nu, \pi_{\text{Trace}})_i\}_{i=1}^{\kappa}, m, \Sigma, \mathbf{reg})$ on input the group public key gpk, κ tracing shares and their proofs, a message m, a signature Σ, and the users' registeration table, it outputs an identity $\mathsf{uid} > 0$ of the user who produced Σ plus a proof θ_{Trace} attesting to this claim. If the algorithm is

unable to trace the signature to a user, it returns $(0, \theta_{\text{Trace}})$. This algorithm does not require any secret information and hence could be run by any party.

- $\mathsf{TraceVerify}(\mathsf{gpk}, \mathsf{uid}, \theta_{\text{Trace}}, \mathbf{upk}[\mathsf{uid}], m, \Sigma)$ on input the group public key gpk, a user identity uid, a tracing proof θ_{Trace}, the user's public key $\mathbf{upk}[\mathsf{uid}]$, a message m, and a signature Σ, outputs 1 if θ_{Trace} is a valid proof that uid has produced Σ or 0 otherwise.

3.1 Security of Dynamic Group Signatures with Distributed Traceability

Our model extends Bellare's et al. model [13] to provide distributed traceability and additionally captures tracing soundness as recently defined by [49] in the context of group signatures with a single tracing manager, which is vital for many applications as we explain later. Moreover, our non-frameability definition is slightly stronger than that of [13].

The security requirements of a dynamic group signature with distributed traceability are: *correctness*, *anonymity*, *non-frameability*, *traceability* and *tracing soundness*. To define those requirements, we use a set of games in which the adversary has access to a set of oracles. The following global lists are maintained: HUL is a list of honest users; CUL is a list of corrupt users whose personal secret keys have been chosen by the adversary; BUL is a list of bad users whose personal and group signing keys have been revealed to the adversary; SL is a list of signatures obtained from the Sign oracle; CL is a list of challenge signatures obtained from the challenge oracle.

The details of the following oracles are given in Fig. 1.

$\mathsf{AddU}(\mathsf{uid})$ adds an honest user uid to the group.
$\mathsf{CrptU}(\mathsf{uid}, \mathsf{pk})$ adds a new corrupt user whose public key $\mathbf{upk}[\mathsf{uid}]$ is chosen by the adversary. This is called in preparation for calling the SndM oracle.
$\mathsf{SndM}(\mathsf{uid}, M_{\mathsf{in}})$ used to engage in the Join-Issue protocol with the honest, Issue-executing group manager.
$\mathsf{SndU}(\mathsf{uid}, M_{\mathsf{in}})$ used to engage in the Join-Issue protocol with an honest, Join-executing user uid on behalf of the corrupt group manager.
$\mathsf{RReg}(\mathsf{uid})$ returns the registration information $\mathbf{reg}[\mathsf{uid}]$ of user uid.
$\mathsf{WReg}(\mathsf{uid}, val)$ modifies the entry $\mathbf{reg}[\mathsf{uid}]$ by setting $\mathbf{reg}[\mathsf{uid}] := val$.
$\mathsf{RevealU}(\mathsf{uid})$ returns the personal secret key $\mathbf{usk}[\mathsf{uid}]$ and the group signing key $\mathbf{gsk}[\mathsf{uid}]$ of group member uid.
$\mathsf{Sign}(\mathsf{uid}, m)$ returns a signature on the message m by the group member uid.
$\mathsf{CH}_b(\mathsf{uid}_0, \mathsf{uid}_1, m)$ is a left-right oracle for defining anonymity. The adversary sends a couple of identities $(\mathsf{uid}_0, \mathsf{uid}_1)$ and a message m and receives a group signature by member uid_b for $b \leftarrow \{0, 1\}$.
$\mathsf{TraceShare}(\mathsf{tid}, m, \Sigma)$ returns the tracing share of signature Σ of tracing manager $\mathsf{TM}_{\mathsf{tid}}$.
$\mathsf{Trace}(m, \Sigma)$ returns the identity of the signer of the signature Σ, i.e. first obtains the different tracing shares and then combines them.

The following security requirements are defined by the games in Fig. 2.

Correctness. This guarantees that: signatures produced by honest users are accepted by the Verify algorithm, the tracing shares are accepted by the ShareVerify algorithm, and the final tracing outcome of TraceCombine is accepted by the TraceVerify algorithm and points out to the user who produced the signature.

Formally, a $\mathcal{DGSDT}$ scheme is *correct* if for all $\lambda, \kappa \in \mathbb{N}$, the advantage $\mathsf{Adv}^{\text{Corr}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) := \Pr[\mathsf{Exp}^{\text{Corr}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) = 1]$ is negligible for all PPT adversaries $\mathcal{A}$.

(Full) Anonymity. This requires that signatures do not reveal the identity of the group member who produced them. In the game, the adversary can corrupt any user and fully corrupt the group manager. It can also learn the secret tracing keys of up to $\kappa - 1$ tracing managers of its choice. The only restriction is that the adversary is not allowed to query the TraceShare and Trace oracles on the challenge signature.

AddU(uid)
- ⋆ If uid ∈ HUL ∪ CUL ∪ BUL Then Return ⊥.
- ⋆ $(\mathbf{usk}[\mathsf{uid}], \mathbf{upk}[\mathsf{uid}]) \leftarrow \mathsf{UKg}(1^\lambda)$.
- ⋆ $\mathsf{HUL} := \mathsf{HUL} \cup \{\mathsf{uid}\}$, $\mathsf{cert}_{\mathsf{uid}} := \perp$, $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Issue}} := \mathsf{cont}$.
- ⋆ $\mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}} := (\mathsf{gpk}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}])$.
- ⋆ $\mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}} := (\mathsf{msk}, \mathsf{uid}, \mathbf{upk}[\mathsf{uid}])$.
- ⋆ $(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}}, \perp)$.
- ⋆ While $(\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Issue}} = \mathsf{cont}$ and $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Join}} = \mathsf{cont})$ Do
  - ∘ $(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}}, M_{\mathsf{Join}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{Issue}}) \leftarrow \mathsf{Issue}(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}}, M_{\mathsf{Issue}})$.
  - ∘ $(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}}, M_{\mathsf{Join}})$.
- ⋆ If $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Issue}} = \mathsf{accept}$ Then $\mathbf{reg}[\mathsf{uid}] := \mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}}$.
- ⋆ If $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Join}} = \mathsf{accept}$ Then $\mathbf{gsk}[\mathsf{uid}] := \mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}}$.
- ⋆ Return $\mathbf{upk}[\mathsf{uid}]$.

SndU(uid, M_{in})
- ⋆ If uid ∈ CUL ∪ BUL Then Return ⊥.
- ⋆ If uid ∉ HUL Then
  - ∘ $\mathsf{HUL} := \mathsf{HUL} \cup \{\mathsf{uid}\}$.
  - ∘ $(\mathbf{usk}[\mathsf{uid}], \mathbf{upk}[\mathsf{uid}]) \leftarrow \mathsf{UKg}(1^\lambda)$.
  - ∘ $\mathbf{gsk}[\mathsf{uid}] := \perp$, $M_{\mathsf{in}} := \perp$.
- ⋆ If $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Join}} \neq \mathsf{cont}$ Then Return ⊥.
- ⋆ If $\mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}}$ is undefined
  - ∘ $\mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}} := (\mathsf{gpk}, \mathsf{uid}, \mathbf{usk}[\mathsf{uid}])$.
- ⋆ $(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}}, M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{Join}}) \leftarrow \mathsf{Join}(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}}, M_{\mathsf{in}})$
- ⋆ If $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Join}} = \mathsf{accept}$ Then $\mathbf{gsk}[\mathsf{uid}] := \mathsf{st}^{\mathsf{uid}}_{\mathsf{Join}}$.
- ⋆ Return $(M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{Join}})$.

Trace(m, Σ)
- ⋆ Return $(\perp, \perp)$ if $\mathsf{Verify}(\mathsf{gpk}, m, \Sigma) = 0$ or $(m, \Sigma) \in \mathsf{CL}$.
- ⋆ For $\mathsf{tid} = 1$ To κ Do
  - ∘ $(\nu, \pi_{\mathrm{Trace}})_{\mathsf{tid}} \leftarrow \mathsf{TraceShare}(\mathsf{gpk}, \mathsf{tsk}_{\mathsf{tid}}, m, \Sigma)$.
- ⋆ Return $\mathsf{TraceCombine}(\mathsf{gpk}, \{(\nu, \pi_{\mathrm{Trace}})_i\}_{i=1}^{\kappa}, m, \Sigma, \mathbf{reg})$.

TraceShare(tid, m, Σ)
- ⋆ Return $(\perp, \perp)$ if $\mathsf{Verify}(\mathsf{gpk}, m, \Sigma) = 0$ or $(m, \Sigma) \in \mathsf{CL}$.
- ⋆ Return $\mathsf{TraceShare}(\mathsf{gpk}, \mathsf{tsk}_{\mathsf{tid}}, m, \Sigma)$.

RevealU(uid)
- ⋆ Return ⊥ if uid ∉ HUL \ (CUL ∪ BUL).
- ⋆ $\mathsf{BUL} := \mathsf{BUL} \cup \{\mathsf{uid}\}$.
- ⋆ Return $(\mathbf{usk}[\mathsf{uid}], \mathbf{gsk}[\mathsf{uid}])$.

CrptU(uid, pk)
- ⋆ Return ⊥ if uid ∈ HUL ∪ CUL.
- ⋆ $\mathsf{CUL} := \mathsf{CUL} \cup \{\mathsf{uid}\}$.
- ⋆ $\mathbf{upk}[\mathsf{uid}] := \mathsf{pk}$, $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Issue}} := \mathsf{cont}$.
- ⋆ $\mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}} := (\mathsf{msk}, \mathsf{uid}, \mathbf{upk}[\mathsf{uid}])$.
- ⋆ Return accept.

RReg(uid)
- ⋆ Return $\mathbf{reg}[\mathsf{uid}]$.

WReg(uid, val)
- ⋆ $\mathbf{reg}[\mathsf{uid}] := val$.

SndM(uid, M_{in})
- ⋆ Return ⊥ if uid ∉ CUL ∪ BUL.
- ⋆ Return ⊥ if $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Issue}} \neq \mathsf{cont}$.
- ⋆ If $\mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}}$ is undefined
  - ∘ $\mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}} := (\mathsf{msk}, \mathsf{uid}, \mathbf{upk}[\mathsf{uid}])$.
- ⋆ $(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}}, M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{Issue}}) \leftarrow \mathsf{Issue}(\mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}}, M_{\mathsf{in}})$.
- ⋆ If $\mathsf{dec}^{\mathsf{uid}}_{\mathsf{Issue}} = \mathsf{accept}$ Then $\mathbf{reg}[\mathsf{uid}] := \mathsf{st}^{\mathsf{uid}}_{\mathsf{Issue}}$.
- ⋆ Return $(M_{\mathsf{out}}, \mathsf{dec}^{\mathsf{uid}}_{\mathsf{Issue}})$.

Sign(uid, m)
- ⋆ Return ⊥ if uid ∉ HUL or $\mathbf{gsk}[\mathsf{uid}] = \perp$.
- ⋆ $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], m)$.
- ⋆ $\mathsf{SL} := \mathsf{SL} \cup \{(\mathsf{uid}, m, \Sigma)\}$.
- ⋆ Return Σ.

CH$_b$(uid$_0$, uid$_1$, m)
- ⋆ Return ⊥ if $\mathsf{uid}_0 \notin \mathsf{HUL}$ or $\mathsf{uid}_1 \notin \mathsf{HUL}$.
- ⋆ Return ⊥ if $\exists b \in \{0,1\}$ s.t. $\mathbf{gsk}[\mathsf{uid}_b] = \perp$.
- ⋆ $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}_b], m)$.
- ⋆ $\mathsf{CL} := \mathsf{CL} \cup \{(m, \Sigma)\}$.
- ⋆ Return Σ.

Fig. 1. Details of the oracles used in the security games

Since the adversary can learn the personal secret and group signing keys of any user, including the challenge users, our definition captures full key exposure attacks. Also, since the adversary can corrupt up to $\kappa - 1$ tracing managers, it can obtain up to $\kappa - 1$ tracing shares of the challenge signature.

In the game, the adversary chooses a message and two group members and gets a signature by either member and wins if it correctly guesses the member. WLOG we allow the adversary a single call to the challenge oracle. A hybrid argument (similar to that used in [13]) can be used to prove that this is sufficient.

Formally, a $\mathcal{DGSDT}$ scheme is *(fully) anonymous* if for all $\lambda, \kappa \in \mathbb{N}$, the advantage $\mathsf{Adv}^{\mathrm{Anon}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda)$ is negligible for all PPT adversaries $\mathcal{A}$, where

$$\mathsf{Adv}^{\mathrm{Anon}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) := \left|\Pr[\mathsf{Exp}^{\text{Anon-0}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{Anon-1}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) = 1]\right|.$$

Non-Frameability. This ensures that even if the rest of the group as well as the group and all tracing managers are fully corrupt, they cannot produce a signature that traces to an honest group member who did not produce such a signature. Our definition is stronger than that used in other group signature models, e.g. [13], in the sense that the adversary in our game wins even if it produces a new signature on a message that was queried to the signing oracle, i.e. analogous to strong unforgeability in traditional signatures. The definition can in a straightforward manner be adapted to the weaker variant by requiring that the adversary's signature is on a new message that the framed user did not sign.

Experiment: $\mathsf{Exp}^{\text{Corr}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda)$

- $(\mathsf{gpk}, \mathsf{msk}, \{\mathsf{tsk}_i\}_{i=1}^{\kappa}) \leftarrow \mathsf{GKg}(1^\lambda, \kappa)$; $\mathsf{HUL} := \emptyset$.
- $(\mathsf{uid}, m) \leftarrow \mathcal{A}(\mathsf{gpk} : \mathsf{AddU}(\cdot), \mathsf{RReg}(\cdot))$.
- If $\mathsf{uid} \notin \mathsf{HUL}$ or $\mathbf{gsk}[\mathsf{uid}] = \perp$ Then Return 0.
- $\Sigma \leftarrow \mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], m)$.
- If $\mathsf{Verify}(\mathsf{gpk}, m, \Sigma) = 0$ Then Return 1.
- For $\mathsf{tid} = 1$ To κ Do
  - $(\nu, \pi_{\text{Trace}})_{\mathsf{tid}} \leftarrow \mathsf{TraceShare}(\mathsf{gpk}, \mathsf{tsk}_{\mathsf{tid}}, m, \Sigma)$.
  - If $\nu_{\mathsf{tid}} = \perp$ or $\pi_{\text{Trace}_{\mathsf{tid}}} = \perp$ Then Return 1.
  - If $\mathsf{ShareVerify}(\mathsf{gpk}, \mathsf{tid}, \nu_{\mathsf{tid}}, \pi_{\text{Trace}_{\mathsf{tid}}}, m, \Sigma) = 0$ Then Return 1.
- $(\mathsf{uid}^*, \theta_{\text{Trace}}) \leftarrow \mathsf{TraceCombine}(\mathsf{gpk}, \{(\nu, \pi_{\text{Trace}})_{\mathsf{tid}}\}_{\mathsf{tid}=1}^{\kappa}, m, \Sigma, \mathbf{reg})$.
- If $\mathsf{uid} \neq \mathsf{uid}^*$ Then Return 1.
- If $\mathsf{TraceVerify}(\mathsf{gpk}, \mathsf{uid}, \theta_{\text{Trace}}, \mathbf{upk}[\mathsf{uid}], m, \Sigma) \neq 1$ Then Return 1 Else Return 0.

Experiment: $\mathsf{Exp}^{\text{Anon-}b}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda)$

- $(\mathsf{st}_{\mathsf{init}}, \mathsf{BTL} \subset [\kappa]) \leftarrow \mathcal{A}_{\mathsf{init}}(\lambda, \kappa)$.
- $(\mathsf{gpk}, \mathsf{msk}, \{\mathsf{tsk}_i\}_{i=1}^{\kappa}) \leftarrow \mathsf{GKg}(1^\lambda, \kappa)$.
- $\mathsf{HUL}, \mathsf{CUL}, \mathsf{BUL}, \mathsf{SL}, \mathsf{CL} := \emptyset$.
- $b^* \leftarrow \mathcal{A}_{\mathsf{guess}}\big(\mathsf{st}_{\mathsf{init}}, \mathsf{gpk}, \mathsf{msk}, \{\mathsf{tsk}_i\}_{i\in\mathsf{BTL}} : \mathsf{AddU}(\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{SndU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot,), \mathsf{RevealU}(\cdot), \mathsf{TraceShare}(\cdot,\cdot,\cdot), \mathsf{Trace}(\cdot,\cdot), \mathsf{CH}_b(\cdot,\cdot,\cdot)\big)$.
- Return b^*.

Experiment: $\mathsf{Exp}^{\text{Non-Frame}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda)$

- $(\mathsf{gpk}, \mathsf{msk}, \{\mathsf{tsk}_i\}_{i=1}^{\kappa}) \leftarrow \mathsf{GKg}(1^\lambda, \kappa)$.
- $\mathsf{HUL}, \mathsf{CUL}, \mathsf{BUL}, \mathsf{SL} := \emptyset$.
- $(m^*, \Sigma^*, \mathsf{uid}^*, \theta^*_{\text{Trace}}) \leftarrow \mathcal{A}\big(\mathsf{gpk}, \mathsf{msk}, \{\mathsf{tsk}_i\}_{i=1}^{\kappa} : \mathsf{CrptU}(\cdot,\cdot), \mathsf{SndU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot), \mathsf{RevealU}(\cdot), \mathsf{Sign}(\cdot,\cdot)\big)$.
- If $\mathsf{Verify}(\mathsf{gpk}, m^*, \Sigma^*) = 0$ Then Return 0.
- If $\mathsf{TraceVerify}(\mathsf{gpk}, \mathsf{uid}^*, \theta^*_{\text{Trace}}, \mathbf{upk}[\mathsf{uid}^*], m^*, \Sigma^*) \neq 1$ Then Return 0.
- If $\mathsf{uid}^* \notin \mathsf{HUL} \setminus \mathsf{BUL}$ or $(\mathsf{uid}^*, m^*, \Sigma^*) \in \mathsf{SL}$ Then Return 0 Else Return 1.

Experiment: $\mathsf{Exp}^{\text{Trace}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda)$

- $(\mathsf{gpk}, \mathsf{msk}, \{\mathsf{tsk}_i\}_{i=1}^{\kappa}) \leftarrow \mathsf{GKg}(1^\lambda, \kappa)$.
- $\mathsf{HUL}, \mathsf{CUL}, \mathsf{BUL}, \mathsf{SL} := \emptyset$.
- $(m^*, \Sigma^*) \leftarrow \mathcal{A}\big(\mathsf{gpk}, \{\mathsf{tsk}_i\}_{i=1}^{\kappa} : \mathsf{AddU}(\cdot), \mathsf{CrptU}(\cdot,\cdot), \mathsf{SndM}(\cdot,\cdot), \mathsf{RevealU}(\cdot), \mathsf{Sign}(\cdot,\cdot), \mathsf{RReg}(\cdot)\big)$.
- If $\mathsf{Verify}(\mathsf{gpk}, m^*, \Sigma^*) = 0$ Then Return 0.
- For $\mathsf{tid} = 1$ To κ Do
  - $(\nu, \pi_{\text{Trace}})_{\mathsf{tid}} \leftarrow \mathsf{TraceShare}(\mathsf{gpk}, \mathsf{tsk}_{\mathsf{tid}}, m^*, \Sigma^*)$.
  - If $\nu_{\mathsf{tid}} = \perp$ or $\pi_{\text{Trace}_{\mathsf{tid}}} = \perp$ Then Return 1.
  - If $\mathsf{ShareVerify}(\mathsf{gpk}, \mathsf{tid}, \nu_{\mathsf{tid}}, \pi_{\text{Trace}_{\mathsf{tid}}}, m^*, \Sigma^*) = 0$ Then Return 1.
- $(\mathsf{uid}^*, \theta^*_{\text{Trace}}) \leftarrow \mathsf{TraceCombine}(\mathsf{gpk}, \{(\nu, \pi_{\text{Trace}})_i\}_{i=1}^{\kappa}, m^*, \Sigma^*, \mathbf{reg})$.
- If $\mathsf{uid}^* = 0$ or $\mathsf{TraceVerify}(\mathsf{gpk}, \mathsf{uid}^*, \theta^*_{\text{Trace}}, \mathbf{upk}[\mathsf{uid}^*], m^*, \Sigma^*) = 0$ Then Return 1 Else Return 0.

Experiment: $\mathsf{Exp}^{\text{Trace-Sound}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda)$

- $(\mathsf{gpk}, \mathsf{msk}, \{\mathsf{tsk}_i\}_{i=1}^{\kappa}) \leftarrow \mathsf{GKg}(1^\lambda, \kappa)$.
- $\mathsf{HUL}, \mathsf{CUL}, \mathsf{BUL}, \mathsf{SL} := \emptyset$.
- $(m^*, \Sigma^*, \mathsf{uid}^*_1, \theta^*_{\text{Trace}_1}, \mathsf{uid}^*_2, \theta^*_{\text{Trace}_2}) \leftarrow \mathcal{A}\big(\mathsf{gpk}, \mathsf{msk}, \{\mathsf{tsk}_i\}_{i=1}^{\kappa} : \mathsf{CrptU}(\cdot,\cdot), \mathsf{WReg}(\cdot,\cdot)\big)$.
- If $\mathsf{Verify}(\mathsf{gpk}, m^*, \Sigma^*) = 0$ or $\mathsf{uid}^*_1 = \mathsf{uid}^*_2$ or $\mathsf{uid}^*_1 = \perp$ or $\mathsf{uid}^*_2 = \perp$ Then Return 0.
- If $\exists i \in \{1, 2\}$ s.t. $\mathsf{TraceVerify}(\mathsf{gpk}, \mathsf{uid}^*_i, \theta^*_{\text{Trace}_i}, \mathbf{upk}[\mathsf{uid}^*_i], m^*, \Sigma^*) = 0$ Then Return 0 Else Return 1.

Fig. 2. Security games for dynamic group signatures with distributed traceability

Formally, a $\mathcal{DGSDT}$ scheme is *non-frameable* if for all $\lambda, \kappa \in \mathbb{N}$, the advantage $\mathsf{Adv}^{\text{Non-Frame}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) := \Pr[\mathsf{Exp}^{\text{Non-Frame}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) = 1]$ is negligible for all PPT adversaries $\mathcal{A}$.

Traceability. This ensures that the adversary cannot produce a signature that cannot be traced to a member in the group. In the game, the adversary can corrupt any user and learn the tracing keys of all tracing managers. The only restriction is that the adversary is not given the group manager's secret key as this would allow it to create dummy users which are thus untraceable.

Formally, a $\mathcal{DGSDT}$ scheme is *traceable* if for all $\lambda, \kappa \in \mathbb{N}$, the advantage $\mathsf{Adv}^{\text{Trace}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) := \Pr[\mathsf{Exp}^{\text{Trace}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) = 1]$ is negligible for all PPT adversaries $\mathcal{A}$.

Tracing Soundness. Tracing soundness, as recently defined by [49] in the context of group signatures with a single tracing manager, requires that even if the group and all tracing managers as well as all members of the group collude, they cannot produce a valid signature that traces to two different members. As shown (in the single tracing authority setting) in [49], such a property is important for many applications. For example, applications where signers might get rewarded or where abusers might be prosecuted. In such applications, it is important that signatures can only trace to one user. Refer to [49] for more details.

Formally, a $\mathcal{DGSDT}$ scheme has *tracing soundness* if for all $\lambda, \kappa \in \mathbb{N}$, $\mathsf{Adv}^{\text{Trace-Sound}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) := \Pr[\mathsf{Exp}^{\text{Trace-Sound}}_{\mathcal{DGSDT},\mathcal{A},\kappa}(\lambda) = 1]$ is negligible for all PPT adversaries $\mathcal{A}$.

4 Building Blocks

In this section we present the building blocks that we use in our constructions.

4.1 Distributed Tag-Based Encryption with Public Verification

A Distributed Tag-Based Encryption scheme DTBE is a special case of threshold tag-based encryption [6] where n out of n decryption servers must compute their decryption shares honestly for the decryption to succeed. *Public verification* requires that checking the well-formedness of the ciphertext only require public information. We say the scheme is *non-interactive* if decrypting a ciphertext requires no interaction among the decryption servers. Also, the scheme is *robust* if invalid decryption shares can be identified by the combiner.

Formally, a DTBE scheme for a message space $\mathcal{M}_{\mathcal{DTBE}}$ and a tag space $\mathcal{T}_{\mathcal{DTBE}}$ is a tuple of polynomial-time algorithms $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{IsValid}, \mathsf{ShareDec}, \mathsf{ShareVerify}, \mathsf{Combine})$, where $\mathsf{Setup}(1^\lambda, n)$ outputs a public key pk and vectors $\mathbf{svk} = (\mathsf{svk}_1, \ldots, \mathsf{svk}_n)$ and $\mathbf{sk} = (\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$ of verification/secret keys for the decryption servers; $\mathsf{Enc}(\mathsf{pk}, t, m)$ outputs a ciphertext C_{dtbe} on the message m under the tag t; $\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{dtbe}})$ outputs 1 if the ciphertext is valid under the tag t w.r.t. the pubic key pk or 0 otherwise; $\mathsf{ShareDec}(\mathsf{pk}, \mathsf{sk}_i, t, C_{\text{dtbe}})$ takes as input the public key pk, the i-th server secret key sk_i, a tag t, and the ciphertext C_{dtbe}, and outputs the i-th server decryption share ν_i of C_{dtbe} or the reject symbol $\perp$; $\mathsf{ShareVerify}(\mathsf{pk}, \mathsf{svk}_i, t, C_{\text{dtbe}}, \nu_i)$ takes as input the public key pk, the i-th server verification key svk_i, a tag t, the ciphertext C_{dtbe}, and the i-th server decryption share ν_i and outputs 1 if the decryption share ν_i is valid or 0 otherwise. $\mathsf{Combine}(\mathsf{pk}, \{\mathsf{svk}_i\}_{i=1}^n, \{\nu_i\}_{i=1}^n, C_{\text{dtbe}}, t)$ outputs either the message m or the reject symbol $\perp$.

We say the scheme is *correct* if for every message $m \in \mathcal{M}_{\mathcal{DTBE}}$, every tag $t \in \mathcal{T}_{\mathcal{DTBE}}$ and every $(\mathsf{pk}, \{\mathsf{svk}\}_{i=1}^n, \{\mathsf{sk}\}_{i=1}^n)$ output by Setup, if $C_{\text{dtbe}} \leftarrow \mathsf{Enc}(\mathsf{pk}, t, m)$ then we have that:

1. $\forall i \in [n]$, if $\nu_i \leftarrow \mathsf{ShareDec}(\mathsf{pk}, \mathsf{sk}_i, t, C_{\text{dtbe}})$ then $\mathsf{ShareVerify}(\mathsf{pk}, \mathsf{svk}_i, t, C_{\text{dtbe}}, \nu_i) = 1$.
2. $m \leftarrow \mathsf{Combine}(\mathsf{pk}, \{\mathsf{svk}_i\}_{i=1}^n, \{\nu_i\}_{i=1}^n, C_{\text{dtbe}}, t)$.

Besides correctness, we require two security properties: *Selective-Tag weak Indistinguishability against Adaptive Chosen Ciphertext Attacks (ST-wIND-CCA)* [42] and *Decryption Consistency (DECCON)*. Informally, the former requires that an adversary who gets a decryption oracle for any ciphertext under a tag different from the target tag (which is chosen beforehand), cannot distinguish which challenge message was encrypted. The latter requires that an adversary cannot output two different sets of decryption shares of a ciphertext which open differently. The formal definitions of those can be found in Appendix A.

Our Distributed Tag-Based Encryption Scheme. We provide in Fig. 3 a new efficient construction of a distributed tag-based encryption schme with public verification that works in the efficient Type-III bilinear group setting. Our scheme which is secure in the standard model under a variant of the DLIN assumption, namely, the $\text{XDLIN}_{\mathbb{G}}$ assumption is based on Kiltz's tag-based encryption scheme [42] and

$\underline{\mathcal{DTBE}.\mathsf{Setup}(1^\lambda, n).}$
- $\mathcal{P} \leftarrow \mathsf{BGrpSetup}(1^\lambda)$.
- $h, w, z, \{u_i\}_{i=1}^n, \{v_i\}_{i=1}^n \leftarrow \mathbb{Z}_p$.
  $u := \sum_{i=1}^n u_i$, $v := \sum_{i=1}^n v_i$, $(H, \tilde{H}) := (G^h, \tilde{G}^h)$,
  $(U, \tilde{U}) := (H^u, \tilde{H}^u)$, $(V, \tilde{V}) := (U^{\frac{1}{v}}, \tilde{U}^{\frac{1}{v}})$,
  $(W, \tilde{W}) := (H^w, \tilde{H}^w)$, $(Z, \tilde{Z}) := (V^z, \tilde{V}^z)$.
- $\mathsf{sk}_i := (u_i, v_i)$, $\mathsf{svk}_i := (\tilde{U}_i := \tilde{H}^{u_i}, \tilde{V}_i := \tilde{V}^{v_i})$.
- $\mathsf{pk} := (\mathcal{P}, H, \tilde{H}, U, \tilde{U}, V, \tilde{V}, W, \tilde{W}, Z, \tilde{Z})$.

$\underline{\mathcal{DTBE}.\mathsf{Enc}(\mathsf{pk}, t, M)}$
- $r_1, r_2 \leftarrow \mathbb{Z}_p$; $C_1 := H^{r_1}$, $C_2 := V^{r_2}$,
  $C_3 := MU^{r_1+r_2}$, $C_4 := (U^t W)^{r_1}$, $C_5 := (U^t Z)^{r_2}$.
- $C_{\text{dtbe}} := \Big(C_1, C_2, C_3, C_4, C_5\Big)$.

$\underline{\mathcal{DTBE}.\mathsf{Combine}(\mathsf{pk}, \{\mathsf{svk}_i\}_{i=1}^n, \{\nu_i\}_{i=1}^n, C_{\text{dtbe}}, t)}$
- If $\mathcal{DTBE}.\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{dtbe}}) = 0$ Then Return $\perp$.
- Parse C_{dtbe} as $(C_1, C_2, C_3, C_4, C_5)$.
- Parse ν_i as $(C_{i,1}, C_{i,2})$ and svk_i as $(\tilde{U}_i, \tilde{V}_i)$.
- Return $\perp$ if $\exists i$ s.t. $\mathcal{DTBE}.\mathsf{ShareVerify}(\mathsf{pk}, \mathsf{svk}_i, t, C_{\text{dtbe}}, \nu_i) = 0$.
- Return $M := \frac{C_3}{\prod_{i=1}^n C_{i,1} C_{i,2}}$.

$\underline{\mathcal{DTBE}.\mathsf{ShareDec}(\mathsf{pk}, \mathsf{sk}_i, t, C_{\text{dtbe}})}$
- Return $\perp$ if $\mathcal{DTBE}.\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{dtbe}}) = 0$.
- Parse C_{dtbe} as $(C_i)_{i=1}^5$ and sk_i as (u_i, v_i).
- Return $\nu_i := (C_{i,1} := C_1^{u_i}, C_{i,2} := C_2^{v_i})$.

$\underline{\mathcal{DTBE}.\mathsf{ShareVerify}(\mathsf{pk}, \mathsf{svk}_i, t, C_{\text{dtbe}}, \nu_i)}$
- Parse svk_i as $(\tilde{U}_i, \tilde{V}_i)$ and ν_i as $(C_{i,1}, C_{i,2})$.
- Parse C_{dtbe} as $(C_1, C_2, C_3, C_4, C_5)$.
- Return 0 If $\mathcal{DTBE}.\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{dtbe}}) = 0$.
- If $e(C_{i,1}, \tilde{H}) \neq e(C_1, \tilde{U}_i)$ Or
  $e(C_{i,2}, \tilde{V}) \neq e(C_2, \tilde{V}_i)$ Then Return 0.
- Else Return 1.

$\underline{\mathcal{DTBE}.\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{tbe}})}$
- Parse C_{dtbe} as $(C_1, C_2, C_3, C_4, C_5)$.
- If $e(C_1, \tilde{U}^t \tilde{W}) \neq e(C_4, \tilde{H})$ Or
  $e(C_2, \tilde{U}^t \tilde{Z}) \neq e(C_5, \tilde{V})$ Then Return 0.
- Else Return 1.

Fig. 3. Our distributed tag-based encryption scheme

its Type-I threshold variant in [6]. Our scheme is efficient and yields ciphertexts of size $\mathbb{G}^5$. Note that in Type-III bilinear groups, elements of $\mathbb{G}$ are much smaller than their Type-I counterparts, especially now that small-characteristic symmetric bilinear groups are rendered insecure [9, 34]. To give a sense of comparison, we outline that at 128-bit security, the size of elements of $\mathbb{G}$ is 256 bits whereas that of their large-characteristic symmetric groups counterparts is 1536 bits. Therefore, our construction yields much shorter ciphertexts than the variants of Kiltz's scheme in symmetric bilinear groups. Our scheme is of independent interest and has other applications beyond the scope of this paper. For instance, combining it with a strongly unforgeable one-time signature scheme (e.g. the full Boneh-Boyen scheme) as per the transformation in [42], we get an efficient distributed (or threshold) IND-CCA secure encryption scheme [27, 28] in Type-III groups which is secure in the standard model under the $\text{XDLIN}_{\mathbb{G}}$ and q-SDH assumptions. In addition, when $n = 1$, we obtain a tag-based encryption scheme in the efficient Type-III setting with 29% shorter ciphertexts than the Type-III variant of Kiltz's scheme in [39] (which yields ciphertexts of size $\mathbb{G}^3 \times \tilde{\mathbb{G}}^2$). Unlike the scheme in [39], which only works for a polynomial (in the security parameter) message space, our scheme has no restriction on the message space.

For simplicity we consider the n-out-of-n case. However, our scheme can, in a straightforward manner, be adapted to the k-out-of-n case by deploying any k-out-of-n secret sharing scheme to compute the servers' secret keys.

We prove the following Theorem in Appendix B

Theorem 1. *The construction in Fig. 3 is a secure distributed tag-based encryption scheme if the $\text{XDLIN}_{\mathbb{G}}$ assumption holds.*

4.2 Digital Signatures

A *digital signature* for a message space $\mathcal{M}_{\mathcal{DS}}$ is a tuple of polynomial-time algorithms $\mathcal{DS} := (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ where KeyGen outputs a pair of secret/public keys $(\mathsf{sk}, \mathsf{pk})$; $\mathsf{Sign}(\mathsf{sk}, m)$ outputs a signature σ on the message m; $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$ outputs 1 if σ is a valid signature on m.

Besides correctness, we require existential unforgeability under adaptive chosen-message attack which demands that all PPT adversaries getting the public key and access to a sign oracle, have a negligible advantage in outputting a valid signature on a message that was not queried to the sign oracle. A weaker variant of existential unforgeability (i.e. existential unforgeability under a weak chosen-message attack) requires that the adversary sends all its queries before seeing the public key.

$\mathcal{DS}.\mathsf{KeyGen}(\mathcal{P})$	$\mathcal{DS}.\mathsf{KeyGen}(\mathcal{P})$
$\circ$ Choose $x, y \leftarrow \mathbb{Z}_p$ and set $(X, Y) := (G^x, G^y)$. $\circ$ Return $\mathsf{sk} := (x, y)$ and $\mathsf{pk} := (X, Y)$.	$\circ$ Choose $x \leftarrow \mathbb{Z}_p$ and set $X := G^x$. $\circ$ Return $\mathsf{sk} := x$ and $\mathsf{pk} := X$.
$\mathcal{DS}.\mathsf{Sign}(\mathsf{sk}, m)$	$\mathcal{DS}.\mathsf{Sign}(\mathsf{sk}, m)$
$\circ$ Choose $r \leftarrow \mathbb{Z}_p$ s.t. $x + ry + m \neq 0$, return $\tilde{\sigma} := \tilde{G}^{\frac{1}{x+ry+m}}$.	$\circ$ If $x + m \neq 0$, return $\tilde{\sigma} := \tilde{G}^{\frac{1}{x+m}}$.
$\mathcal{DS}.\mathsf{Verify}(\mathsf{pk}, m, \sigma)$	$\mathcal{DS}.\mathsf{Verify}(\mathsf{pk}, m, \sigma)$
$\circ$ Return 1 iff $e(XY^rG^m, \tilde{\sigma}) = e(G, \tilde{G})$ or 0 otherwise.	$\circ$ Return 1 iff $e(XG^m, \tilde{\sigma}) = e(G, \tilde{G})$ or 0 otherwise.

Fig. 4. The Full Boneh-Boyen (Left) and the Weak Boneh-Boyen (Right) signatures

$\mathcal{SPDSS}.\mathsf{Setup}(1^\lambda)$	$\mathcal{SPDSS}.\mathsf{Setup}(1^\lambda)$
$\circ$ $\mathcal{P} \leftarrow \mathsf{BGrpSetup}(1^\lambda)$. $\circ$ $F \leftarrow \mathbb{G}$. Return $\mathsf{param} := (\mathcal{P}, F)$.	$\circ$ $\mathcal{P} \leftarrow \mathsf{BGrpSetup}(1^\lambda)$. $\circ$ Return $\mathsf{param} := \mathcal{P}$.
$\mathcal{SPDSS}.\mathsf{KeyGen}(\mathsf{param})$	$\mathcal{SPDSS}.\mathsf{KeyGen}(\mathsf{param})$
$\circ$ Choose $x \leftarrow \mathbb{Z}_p$ and set $\tilde{X} := \tilde{G}^x$. $\circ$ Set $\mathsf{sk} := x$ and $\mathsf{pk} := \tilde{X}$. Return $(\mathsf{sk}, \mathsf{pk})$.	$\circ$ $w, x, y_1, y_2 \leftarrow \mathbb{Z}_p$, $\tilde{W} := \tilde{G}^w, \tilde{X} := \tilde{G}^x$, $\tilde{Y_1} := \tilde{G}^{y_1}, \tilde{Y_2} := \tilde{G}^{y_2}$. $\circ$ $\mathsf{sk} := (w, x, y_1, y_2)$, $\mathsf{pk} := (\tilde{W}, \tilde{X}, \tilde{Y_1}, \tilde{Y_2})$. $\circ$ Return $\big(\mathsf{sk}, \mathsf{pk}\big)$.
$\mathcal{SPDSS}.\mathsf{Sign}(\mathsf{sk}, M)$	$\mathcal{SPDSS}.\mathsf{Sign}(\mathsf{sk}, M)$
$\circ$ $r \leftarrow \mathbb{Z}_p$, $\tilde{\Omega}_1 := \tilde{G}^r$, $\Omega_2 := M^{\frac{x}{r}} F^{\frac{1}{r}}$, $\Omega_3 := \Omega_2^{\frac{x}{r}} G^{\frac{1}{r}}$, the token is $\Omega_4 := G^{\frac{1}{r}}$. $\circ$ Return $\sigma := \Big(\tilde{\Omega}_1, \Omega_2, \Omega_3\Big)$.	$\circ$ $\Omega_1 \leftarrow \mathbb{G}$, $a \leftarrow \mathbb{Z}_p$, $\Omega_2 := G^a$, $\tilde{\Omega}_3 := \tilde{G}^{\frac{1}{a}}$, $\Omega_4 := G^{x-aw} \Omega_1^{-y_1} M^{-y_2}$. $\circ$ Return $\sigma := \Big(\Omega_1, \Omega_2, \tilde{\Omega}_3, \Omega_4\Big)$.
$\mathcal{SPDSS}.\mathsf{Randomize}(M, \sigma, \Omega_4)$	$\mathcal{SPDSS}.\mathsf{Verify}(\mathsf{pk}, M, \sigma)$
$\circ$ $r' \leftarrow \mathbb{Z}_p$, $\tilde{\Omega}'_1 := \Omega_1^{\frac{1}{r'}}$, $\Omega'_2 := \Omega_2^{r'}$, $\Omega'_3 := \Omega_3^{r'^2} \Omega_4^{r'(1-r')}$.	$\circ$ Return 1 if $e(\Omega_1, \tilde{\Omega}_3) = e(G, \tilde{G})$ and $e(G, \tilde{X}) = e(\Omega_2, \tilde{W}) e(\Omega_4, \tilde{G}) e(\Omega_1, \tilde{Y_1}) e(M, \tilde{Y_2})$.
$\mathcal{SPDSS}.\mathsf{Verify}(\mathsf{pk}, M, \sigma)$	
$\circ$ Return 1 if $e(\Omega_2, \tilde{\Omega}_1) = e(M, \tilde{X}) e(F, \tilde{G})$ and $e(\Omega_3, \tilde{\Omega}_1) = e(\Omega_2, \tilde{X}) e(G, \tilde{G})$.	

Fig. 5. The structure-preserving signature of [4] (Left) and that of [3] (Right)

In this paper, we use two digital signatures by Boneh and Boyen [17] which we refer to as the Full Boneh-Boyen signature (Fig. 4 (Left)) and the Weak Boneh-Boyen signature (Fig. 4 (Right)), respectively. Both schemes are secure under the q-SDH assumption. The weaker scheme is secure under a weak chosen-message attack.

Structure-Preserving Signatures. Structure-preserving signatures [2] are signature schemes where the message, the public key and the signature are all group elements, and signatures are verified by evaluating pairing product equations.

In this paper, we use two structure-preserving signatures from the literature. The first scheme is by Abe et al. [4] which offers controlled re-randomizability where a signature can only be re-randomized if the user has a special randomization token. The scheme in the asymmetric setting is illustrated in Fig. 5 (Left). The unforgeability of the scheme relies on an interactive assumption. Refer to [4] for details. The second scheme we use is that of Abe et al. [3]. The scheme in the asymmetric setting is given in Fig. 5 (Right). The strong unforgeability of the scheme relies on the non-interactive q-AGHO assumption.

4.3 Strongly Unforgeable One-Time Signatures

A one-time signature scheme is a signature scheme that is unforgeable against an adversary who is only allowed a single signing query. *Strong Unforgeability* requires that the adversary cannot even forge a new signature on a message that she queried the sign oracle on. In this paper, we will instantiate the one-time signature using the Full Boneh-Boyen signature scheme from Fig. 4.

- Pairing Product Equation (PPE): $\prod_{i=1}^{n} e(A_i, \underline{\tilde{Y}_i}) \prod_{i=1}^{m} e(\underline{X_i}, \tilde{B}_i) \prod_{i=1}^{m} \prod_{j=1}^{n} e(\underline{X_i}, \underline{\tilde{Y}_j})^{k_{i,j}} = t_T \cdot$
- Multi-Scalar Multiplication Equation (MSME) in $\mathbb{G}$: $\prod_{i=1}^{n'} A_i^{\underline{y_i}} \prod_{i=1}^{m} \underline{X_i}^{b_i} \prod_{i=1}^{m} \prod_{j=1}^{n'} \underline{X_i}^{k_{i,j}\underline{y_j}} = T \cdot$
- Multi-Scalar Multiplication Equation (MSME) in $\tilde{\mathbb{G}}$: $\prod_{i=1}^{m'} \tilde{B}_i^{\underline{x_i}} \prod_{i=1}^{n} \underline{\tilde{Y}_i}^{a_i} \prod_{i=1}^{m'} \prod_{j=1}^{n} \underline{\tilde{Y}_i}^{k_{i,j}\underline{x_j}} = \tilde{T} \cdot$
- Quadratic Equation (QE) in $\mathbb{Z}_p$: $\sum_{i=1}^{n'} a_i \underline{y_i} + \sum_{i=1}^{m'} \underline{x_i} b_i + \sum_{i=1}^{m'} \sum_{j=1}^{n'} \underline{x_i y_j} = t \cdot$

Fig. 6. Types of equations one can use Groth-Sahai proofs for

4.4 Non-Interactive Zero-Knowledge Proofs

Let $\mathcal{R}$ be an efficiently computable relation. For pairs $(x, w) \in \mathcal{R}$, we call x the statement and w the witness. We define the language $\mathcal{L}$ as all the statements x in $\mathcal{R}$. A Non-Interactive Zero-Knowledge (NIZK) proof system [16] for $\mathcal{R}$ is defined by a tuple of algorithms $\mathcal{NIZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{SimSetup}, \mathsf{SimProve})$.

$\mathsf{Setup}(1^\lambda)$ outputs a common reference string crs and an extraction key xk which allows for witness extraction. $\mathsf{Prove}(\mathsf{crs}, x, w)$ outputs a proof π that $(x, w) \in \mathcal{R}$. $\mathsf{Verify}(\mathsf{crs}, x, \pi)$ outputs 1 if the proof is valid, or 0 otherwise. $\mathsf{Extract}(\mathsf{crs}, \mathsf{xk}, x, \pi)$ outputs a witness. $\mathsf{SimSetup}(1^\lambda)$ outputs a simulated reference string $\mathsf{crs}_{\mathsf{Sim}}$ and a trapdoor key tr that allows for proof simulation. $\mathsf{SimProve}(\mathsf{crs}_{\mathsf{Sim}}, \mathsf{tr}, x)$ outputs a simulated proof π_{Sim} without a witness.

We require: completeness, soundness and zero-knowledge. Completeness requires that honestly generated proofs are accepted; Soundness requires that it is infeasible (but for a small probability) to produce a valid proof for a false statement; Zero-knowledge requires that a proof reveals no information about the witness used. For formal definitions refer to [16].

GROTH-SAHAI PROOFS. Groth-Sahai (GS) proofs [37] are efficient non-interactive proofs in the Common Reference String (CRS) model. The language for the system has the form

$$\mathcal{L} := \{\mathsf{statement} \mid \exists\, \mathsf{witness} : E_i(\mathsf{statement}, \mathsf{witness})_{i=1}^{n} \text{ hold } \},$$

where $E_i(\mathsf{statement}, \cdot)$ is one of the types of equation summarized in Fig. 6, where $X_1, \ldots, X_m \in \mathbb{G}, \tilde{Y}_1, \ldots, \tilde{Y}_n \in \tilde{\mathbb{G}}, x_1, \ldots, x_{m'}, y_1, \ldots, y_{n'} \in \mathbb{Z}_p$ are secret variables (hence underlined), whereas $A_i, T \in \mathbb{G}, \tilde{B}_i, \tilde{T} \in \tilde{\mathbb{G}}, a_i, b_i, k_{i,j}, t \in \mathbb{Z}_p, t_T \in \mathbb{G}_T$ are public constants.

The proof system has perfect completeness, (perfect) soundness, composable witness-indistinguishability/zero-knowledge. We use the SXDH-based instantiation of the proofs. Refer to [37] for details.

5 Our Generic Construction

In this section, we present our generic construction for dynamic group signatures with distributed traceability.

Overview of the construction. The idea behind our generic construction has some in common with Groth's scheme [36] in that we combine a standard NIZK proof system with a weakly secure tag-based encryption scheme and a strong one-time signature scheme to eliminate the need for the more expensive simulation-sound NIZK systems [47] and IND-CCA public-key encryption schemes which were required by the construction of Bellare et al. [13]. However, unlike [36], our framework provides distributed traceability, has a concurrent join protocol and achieves tracing soundness.

Our generic construction requires three digital signatures $\mathcal{DS}_1$, $\mathcal{DS}_2$ and $\mathcal{WDS}$ where the first two have to be unforgeable against a standard adaptive chosen-message attack, whereas it suffices for the third scheme to be unforgeable against a weak chosen-message attack. We also require a strongly

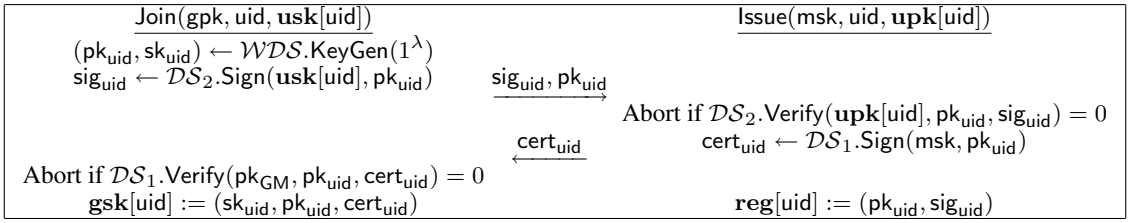

Fig. 7. The Join/Issue protocol for our construction

unforgeable one-time signature scheme $\mathcal{OTS}$ that is secure against an adaptive chosen-message attack. Additionally, we require a NIZK proof of knowledge system $\mathcal{NIZK}$ and a ST-wIND-CCA distributed tag-based encryption scheme $\mathcal{DTBE}$. In order to have efficient tracing, we ask that $\mathcal{DTBE}$ is non-interactive and robust. As was noted by [31], such a propety simplifies tracing even in traditional group signatures with a single tracing manager. Finally, we require a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{T}_{\mathcal{DTBE}}$. For simplicity and WLOG we assume that $\mathcal{T}_{\mathcal{DTBE}} = \mathcal{M}_{\mathcal{WDS}}$. Otherwise, one could use a second hash function. Note that one can use the same signature scheme for both $\mathcal{DS}_1$ and $\mathcal{DS}_2$ but using different key pairs.

The GKg algorithm runs $\mathcal{NIZK}.\mathsf{Setup}$ to generate a common reference string crs for $\mathcal{NIZK}$. It also runs $\mathcal{DTBE}.\mathsf{Setup}$ to generate $(\mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^{\kappa}, \{\mathsf{sk}_i\}_{i=1}^{\kappa})$, and $\mathcal{DS}_1.\mathsf{KeyGen}$ to generate $(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}})$. The group public key is $\mathsf{gpk} := (1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^{\kappa}, \mathcal{H})$. The group managers' secret key is $\mathsf{msk} := \mathsf{sk}_{\mathsf{GM}}$, whereas the tracing key of tracing manager TM_i is $\mathsf{tsk}_i := \mathsf{sk}_i$.

A new user creates her personal key pair by running $\mathcal{DS}_2.\mathsf{KeyGen}$ to generate $(\mathbf{upk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}])$. When the user wishes to join the group, she generates a key pair $(\mathsf{pk}_{\mathsf{uid}}, \mathsf{sk}_{\mathsf{uid}})$ for the signature scheme $\mathcal{WDS}$ and then signs $\mathsf{pk}_{\mathsf{uid}}$ using $\mathcal{DS}_2$ and her personal secret key $\mathbf{usk}[\mathsf{uid}]$ to obtain a signature $\mathsf{sig}_{\mathsf{uid}}$. We use $\mathsf{sig}_{\mathsf{uid}}$ as a proof when proving that the user has produced a group signature.

To join the group, the user sends $\mathsf{pk}_{\mathsf{uid}}$ and $\mathsf{sig}_{\mathsf{uid}}$ to the group manager. If $\mathsf{sig}_{\mathsf{uid}}$ is valid, the group manager issues a membership certificate $\mathsf{cert}_{\mathsf{uid}}$ (which is a $\mathcal{DS}_1$ signature on $\mathsf{pk}_{\mathsf{uid}}$ that verifies w.r.t. $\mathsf{pk}_{\mathsf{GM}}$).

To sign a message m, the member chooses a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature $\mathcal{OTS}$ and encrypts her public key $\mathsf{pk}_{\mathsf{uid}}$ with $\mathcal{DTBE}$ using $\mathcal{H}(\mathsf{otsvk})$ as tag (and possibly some randomness τ) to obtain a ciphertext C_{dtbe}. She then signs $\mathcal{H}(\mathsf{otsvk})$ using the digital signature scheme $\mathcal{WDS}$ and her secret key $\mathsf{sk}_{\mathsf{uid}}$ to obtain a signature σ. She then uses $\mathcal{NIZK}$ to produce a proof π proving that: she did the encryption correctly, she has a signature σ on $\mathcal{H}(\mathsf{otsvk})$ that verifies w.r.t. her public key $\mathsf{pk}_{\mathsf{uid}}$ and she has a certificate $\mathsf{cert}_{\mathsf{uid}}$ on $\mathsf{pk}_{\mathsf{uid}}$ from the group manager. Finally, she signs $(m, C_{\text{dtbe}}, \mathsf{otsvk}, \pi)$ using the one-time signature $\mathcal{OTS}$ to obtain σ_{ots}. The group signature is $\Sigma := (\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk})$. To verify the signature, one verifies the proof π and the one-time signature σ_{ots}, and ensures that the ciphertext C_{dtbe} is well-formed.

We remark here if $\mathcal{DS}_1$ and/or $\mathcal{WDS}$ schemes are re-randomizable, one can reveal the signature components which are independent of their respective messages after re-randomization. This simplifies the proof π and subsequently improves the efficiency. The revealed parts of those signatures can then be included as the part of the message to be signed by $\mathcal{OTS}$ to ensure that one achieves the stronger notion of non-frameability.

To trace a signature, the decryption shares ν_i of the ciphertext C_{dtbe} are obtained from the respective tracing managers and then combined together in order to recover the plaintext $\mathsf{pk}_{\mathsf{uid}}$. Then one just needs to search in the registration table $\mathbf{reg}$ to see if any entry $\mathbf{reg}[j].\mathsf{pk}$ matches $\mathsf{pk}_{\mathsf{uid}}$. If this is the case, $(j, (\{\nu\}_{i=1}^{\kappa}, \mathsf{pk}_{\mathsf{uid}}, \mathbf{reg}[j].\mathsf{sig}))$ is returned. Otherwise, $(0, (\{\nu\}_{i=1}^{\kappa}, \mathsf{pk}_{\mathsf{uid}}, \perp))$ is returned. Note that combining the different tracing shares does not require the knowledge of any secret key and hence this could be performed by any party. To verify the correctness of the tracing, one just needs to ensure that

$\underline{\mathsf{GKg}(1^\lambda, \kappa)}$

- $(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}}) \leftarrow \mathcal{DS}_1.\mathsf{KeyGen}(1^\lambda)$; $(\mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \{\mathsf{sk}_i\}_{i=1}^\kappa) \leftarrow \mathcal{DTBE}.\mathsf{Setup}(1^\lambda, \kappa)$.
- $(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathcal{NIZK}.\mathsf{Setup}(1^\lambda)$; Choose a collision-resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \mathcal{T}_{\mathcal{DTBE}}$.
- Let $\mathsf{tsk}_i := \mathsf{sk}_i$, $\mathsf{gpk} := (1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$ and $\mathsf{msk} := \mathsf{sk}_{\mathsf{GM}}$.
- Return $(\mathsf{gpk}, \{\mathsf{tsk}_i\}_{i=1}^\kappa, \mathsf{msk})$.

$\underline{\mathsf{UKg}(1^\lambda)}$

- $(\mathbf{upk}[i], \mathbf{usk}[i]) \leftarrow \mathcal{DS}_2.\mathsf{KeyGen}(1^\lambda)$. Return $(\mathbf{upk}[i], \mathbf{usk}[i])$.

$\underline{\mathsf{Sign}(\mathsf{gpk}, \mathbf{gsk}[\mathsf{uid}], m)}$

- $(\mathsf{otsvk}, \mathsf{otssk}) \leftarrow \mathcal{OTS}.\mathsf{KeyGen}(1^\lambda)$; $C_{\text{dtbe}} \leftarrow \mathcal{DTBE}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{DTBE}}, \mathcal{H}(\mathsf{otsvk}), \mathsf{pk}_{\mathsf{uid}}; \tau)$.
- $\sigma \leftarrow \mathcal{WDS}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{uid}}, \mathcal{H}(\mathsf{otsvk}))$.
- $\pi \leftarrow \mathcal{NIZK}.\mathsf{Prove}(\mathsf{crs}, \{\mathsf{pk}_{\mathsf{uid}}, \tau, \sigma, \mathsf{cert}_{\mathsf{uid}}\} : (C_{\text{dtbe}}, \mathcal{H}(\mathsf{otsvk}), \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}) \in \mathcal{L})$.
- $\sigma_{\text{ots}} \leftarrow \mathcal{OTS}.\mathsf{Sign}(\mathsf{otssk}, (m, C_{\text{dtbe}}, \mathsf{otsvk}, \pi))$.
- Return $\Sigma := (\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk})$.

$\underline{\mathsf{Verify}(\mathsf{gpk}, m, \Sigma)}$

- Parse Σ as $(\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk})$ and gpk as $(1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$.
- Return 1 if all the following verify; otherwise, return 0:
  - $\star$ $\mathcal{OTS}.\mathsf{Verify}(\mathsf{otsvk}, (m, C_{\text{dtbe}}, \mathsf{otsvk}, \pi), \sigma_{\text{ots}}) = 1 \wedge \mathcal{NIZK}.\mathsf{Verify}(\mathsf{crs}, \pi) = 1$.
  - $\star$ $\mathcal{DTBE}.\mathsf{IsValid}(\mathsf{pk}_{\mathcal{DTBE}}, \mathcal{H}(\mathsf{otsvk}), C_{\text{dtbe}}) = 1$.

$\underline{\mathsf{TraceShare}(\mathsf{gpk}, \mathsf{tsk}_i, m, \Sigma)}$

- Parse Σ as $(\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk})$ and gpk as $(1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$.
- Return $\perp$ if $\mathsf{Verify}(\mathsf{gpk}, m, \Sigma) = 0$ Else Return $\mathcal{DTBE}.\mathsf{ShareDec}(\mathsf{pk}_{\mathcal{DTBE}}, \mathsf{tsk}_i, \mathcal{H}(\mathsf{otsvk}), C_{\text{dtbe}})$.

$\underline{\mathsf{ShareVerify}(\mathsf{gpk}, \mathsf{tid}, \nu, m, \Sigma)}$

- Parse Σ as $(\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk})$ and gpk as $(1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$.
- Return $\mathcal{DTBE}.\mathsf{ShareVerify}(\mathsf{pk}_{\mathcal{DTBE}}, \mathsf{svk}_{\mathsf{tid}}, \mathcal{H}(\mathsf{otsvk}), C_{\text{dtbe}}, \nu)$.

$\underline{\mathsf{TraceCombine}(\mathsf{gpk}, \{\nu_i\}_{i=1}^\kappa, m, \Sigma, \mathbf{reg})}$

- Parse Σ as $(\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk})$ and gpk as $(1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$.
- Return $(0, (\{\nu\}_{i=1}^\kappa, \perp, \perp))$ if for any $i \in [\kappa]$ $\mathcal{DTBE}.\mathsf{ShareVerify}(\mathsf{pk}_{\mathcal{DTBE}}, \mathsf{svk}_i, \mathcal{H}(\mathsf{otsvk}), C_{\text{dtbe}}, \nu_i) = 0$.
- $\mathsf{pk}_{\mathsf{uid}} \leftarrow \mathcal{DTBE}.\mathsf{Combine}(\mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \{\nu_i\}_{i=1}^\kappa, C_{\text{dtbe}}, \mathcal{H}(\mathsf{otsvk}))$.
- Return $(0, (\{\nu\}_{i=1}^\kappa, \perp, \perp))$ if $\mathsf{pk}_{\mathsf{uid}} = \perp$.
- If $\exists j$ s.t. $\mathbf{reg}[j].\mathsf{pk} = \mathsf{pk}_{\mathsf{uid}}$ Then Return $(j, (\{\nu\}_{i=1}^\kappa, \mathsf{pk}_{\mathsf{uid}}, \mathbf{reg}[j].\mathsf{sig}))$.
- Return $(0, (\{\nu\}_{i=1}^\kappa, \mathsf{pk}_{\mathsf{uid}}, \perp))$.

$\underline{\mathsf{TraceVerify}(\mathsf{gpk}, \mathsf{uid}, \theta_{\text{Trace}}, \mathbf{upk}[\mathsf{uid}], m, \Sigma)}$

- Parse θ_{Trace} as $(\{\nu\}_{i=1}^\kappa, \mathsf{pk}_{\mathsf{uid}}, \mathsf{sig}_{\mathsf{uid}})$ and Σ as $(\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk})$.
- Parse gpk as $(1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$.
- $\mathsf{pk}'_{\mathsf{uid}} \leftarrow \mathcal{DTBE}.\mathsf{Combine}(\mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \{\nu_i\}_{i=1}^\kappa, C_{\text{dtbe}}, \mathcal{H}(\mathsf{otsvk}))$.
- If $\mathsf{uid} = 0$ Then Return $(\mathsf{pk}'_{\mathsf{uid}} = \perp \ \vee \ \mathsf{Verify}(\mathsf{gpk}, m, \Sigma) = 0)$.
- If $\mathsf{pk}'_{\mathsf{uid}} \neq \mathsf{pk}_{\mathsf{uid}}$ Then Return 0 Else Return $\mathcal{DS}_2.\mathsf{Verify}(\mathbf{upk}[\mathsf{uid}], \mathsf{pk}_{\mathsf{uid}}, \mathsf{sig}_{\mathsf{uid}})$.

Fig. 8. Our generic construction

all decryption shares verify correctly and in the case that $j > 0$, one needs to verify that the signature sig verifies w.r.t. $\mathbf{upk}[j]$.

The construction is detailed in Fig. 8, whereas the Join/Issue protocol is given in Fig. 7. The language associated with the NIZK proof is as follows, where for clarity we underline the elements of the witness:

$$\mathcal{L} : \Big\{ \big((C_{\text{dtbe}}, \mathcal{H}(\mathsf{otsvk}), \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}), (\mathsf{pk}_{\mathsf{uid}}, \tau, \sigma, \mathsf{cert}_{\mathsf{uid}})\big) : \mathcal{DS}_1.\mathsf{Verify}(\mathsf{pk}_{\mathsf{GM}}, \underline{\mathsf{pk}_{\mathsf{uid}}}, \underline{\mathsf{cert}_{\mathsf{uid}}}) = 1 \ \wedge \ \mathcal{WDS}.\mathsf{Verify}(\underline{\mathsf{pk}_{\mathsf{uid}}}, \mathcal{H}(\mathsf{otsvk}), \underline{\sigma}) = 1 \ \wedge \ \mathcal{DTBE}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{DTBE}}, \mathcal{H}(\mathsf{otsvk}), \underline{\mathsf{pk}_{\mathsf{uid}}}; \underline{\tau}) = C_{\text{dtbe}} \Big\}.$$

Theorem 2. *The construction in Fig. 7 and Fig. 8 is a secure dynamic group signature with distributed traceability providing that the building blocks are secure w.r.t. their security requirements.*

The full proof of this Theorem can be found in Appendix D.

Next, we present two example instantiations of the generic construction in the standard model.

6 Instantiations in the Standard Model

Here we provide two example instantiations of the generic framework in the standard model.

6.1 Instantiation I

Here we instantiate the signature schemes $\mathcal{DS}_1$ and $\mathcal{DS}_2$ using the recent structure-preserving signature scheme by Abe et al. [4] (shown in Fig. 5 (Left)), and instantiate $\mathcal{WDS}$ and $\mathcal{OTS}$ with the weak and full Boneh-Boyen signature schemes, respectively. We also instantiate $\mathcal{NIZK}$ using the Groth-Sahai proof system. For the distributed tag-based encryption scheme, we use our new scheme from Fig 3. The size of the signature of this instantiation is $\mathbb{G}^{24} \times \tilde{\mathbb{G}}^{21} \times \mathbb{Z}_p^5$. The details are in Appendix C. The proof for the following Theorem follows from that of Theorem 2.

Theorem 3. *Instantiation I is secure if the Abe et al. signature scheme [4] is unforgeable and the SXDH, XDLIN$_{\mathbb{G}}$ and q-SDH assumptions hold.*

6.2 Instantiation II

To eliminate the need for interactive intractability assumptions, we instead use the strongly unforgeable signature scheme by Abe et al. [3] (Fig. 5 (Right)) to instantiate $\mathcal{DS}_1$ and $\mathcal{DS}_2$ signature schemes. The rest of the tools remain the same as in Instantiation I. The size of the group signature of this instantiation is $\mathbb{G}^{28} \times \tilde{\mathbb{G}}^{24} \times \mathbb{Z}_p^3$. The details are in Appendix C. The proof for the following Theorem follows from that of Theorem 2.

Theorem 4. *Instantiation II is secure if the SXDH, q-AGHO, XDLIN$_{\mathbb{G}}$ and q-SDH assumptions hold.*

6.3 Efficiency Comparison

Since there are no existing constructions which simultaneously offer all the properties as our constructions, we compare the size of the signature of our instantiations with that of Groth's scheme [35] for the single tracing manager setting which is considered the-state-of-the-art. Groth's scheme yields signatures of size $\mathbb{G}^{46} \times \mathbb{Z}_p$ in symmetric groups. Besides the extra distributed traceability feature, our instantiations involve fewer rounds in the join protocol than [35] and, in addition, satisfy tracing soundness.

Acknowledgments. The author was supported by ERC Advanced Grant ERC-2010-AdG-267188-CRIPTO and EPSRC via grant EP/H043454/1.

References

1. M.Abe, M. Chase, B. David, M. Kohlweiss, R. Nishimaki and M. Ohkubo. Constant-Size Structure-Preserving Signatures: Generic Constructions and Simple Assumptions. In *ASIACRYPT 2012*, Springer LNCS 7658, 4–24, 2012.
2. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev and M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *CRYPTO 2010*, Springer LNCS 6223, 209–236, 2010.
3. M. Abe, J. Groth, K. Haralambiev and M. Ohkubo. Optimal Structure-Preserving Signatures in Asymmetric Bilinear Groups. In *CRYPTO 2011*, Springer LNCS 6841, 649–666, 2011.
4. M. Abe, J. Groth, M. Ohkubo and M. Tibouchi. Unified, Minimal and Selectively Randomizable Structure-Preserving Signatures. In TCC 2014, Springer LNCS 8349, 688–712, 2014.
5. M. Abe, K. Haralambiev and M. Ohkubo. Signing on Elements in Bilinear Groups for Modular Protocol Design. *Cryptology ePrint Archive, Report 2010/133*. `http://eprint.iacr.org/2010/133`.
6. S. Arita and K. Tsurudome. Construction of Threshold Public-Key Encryptions through Tag-Based Encryptions. In *ACNS 2009*, Springer LNCS 5536, 186–200, 2009.
7. G. Ateniese, J. Camenisch, S. Hohenberger and B. de Medeiros. Practical group signatures without random oracles. *Cryptology ePrint Archive*. Report 2005/385, available at `http://eprint.iacr.org/2005/385`.
8. G. Ateniese, J. Camenisch, M. Joye and G. Tsudik. A Practical and Provably Secure Coalition-Resistant Group Signature Scheme. In *CRYPTO 2000*, Springer LNCS 1880, 255–270, 2000.

9. R. Barbulescu, P. Gaudry, A. Joux and E. Thom. A Heuristic Quasi-Polynomial Algorithm for Discrete Logarithm in Finite Fields of Small Characteristic. In *EUROCRYPT 2014*, Springer LNCS 8441, 1–16, 2014.
10. G. R. Blakley. Safeguarding Cryptographic Keys. In AFIPS National Computer Conference, AFIPS Press 48, 313–317, 1979.
11. 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*, Springer LNCS 2656, 614–629, 2003.
12. M. Bellare and P. Rogaway. Random oracles are practical: A Paradigm for Designing Efficient Protocols. In *ACM-CCS 1993*, ACM, pp. 62–73, 1993.
13. M. Bellare, H. Shi and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *CT-RSA 2005*, Springer LNCS 3376, 136–153, 2005.
14. V. Benjumea, S. Choi, J. Lopez and M. Yung. Fair Traceable Multi-Group Signatures. In *FC 2008*, Springer LNCS 5143, 231–246, 2008.
15. J. Bichsel, J. Camenisch, G. Neven, N.P. Smart and B. Warinschi. Get Shorty via Group Signatures without Encryption. In *SCN 2010*, Springer LNCS 6280, 381–398, 2010.
16. M. Blum, P. Feldman and S. Micali. Non-interactive zero-knowledge and its applications. In *STOC 1988*, 103–112, 1988.
17. D. Boneh and X. Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. In Journal of Cryptology, volume 21(2), 149–177, 2008.
18. D. Boneh, X. Boyen and H. Shacham. Short Group Signatures. In *CRYPTO 2004*, Springer LNCS 3152, 227–242, 2004.
19. X. Boyen and B.Waters. Compact Group Signatures Without Random Oracles. In *EUROCRYPT 2006*, Springer LNCS 4004, 427–444, 2006.
20. X. Boyen and B.Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC 2007*, Springer LNCS 4450, 1–15, 2007.
21. J. Camenisch and J. Groth. Group Signatures: Better Efficiency and New Theoretical Aspects. In *SCN 2004*, Springer LNCS 2045, 120–133, 2004.
22. J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO 2004*, Springer LNCS 3152, 56–72, 2004.
23. J. Camenisch and M. Michels. A group signature scheme with improved efficiency. In *ASIACRYPT 1998*, Springer LNCS 1514, 160–174, 1998.
24. J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *CRYPTO 1997*, Springer LNCS 1294, 410–424, 1997.
25. D. Chaum and E. van Heyst. Group signatures. In *EUROCRYPT 1991*, Springer LNCS 547, 257–265, 1991.
26. C. Delerablée and D. Pointcheval. Dynamic Fully Anonymous Short Group Signatures. In *VIETCRYPT 2006*, Springer LNCS 4341, 193–210, 2006.
27. Y. Desmedt and Y. Frankel. Threshold Cryptosystems. In *CRYPTO 1989*, Springer LNCS 435, 307–315, 1990.
28. Yair Frankel. A Practical Protocol for Large Group Oriented Networks. In *EUROCRYPT 1989*, Springer LNCS 434, 56–61, 1989.
29. J. Furukawa and H. Imai. An Efficient Group Signature Scheme from Bilinear Maps. In *ACISP 2005*, Springer LNCS 3574, 455–467, 2005.
30. J. Furukawa and S. Yonezawa. Group Signatures with Separate and Distributed Authorities. In *SCN 2004*, Springer LNCS 3352, 77–90, 2004.
31. D. Galindo, B. Libert, M. Fischlin, G. Fuchsbauer, A. Lehmann, M. Manulis and D. Schröder. Public-Key Encryption with Non-interactive Opening: New Constructions and Stronger Definitions. In *AFRICACRYPT 2010*, Springer LNCS 6055, 333–350, 2010.
32. S. Galbraith, K. Paterson and N.P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, **156**, 3113–3121, 2008.
33. E. Ghadafi, N.P. Smart and B. Warinschi. Groth-Sahai proofs revisited. In *PKC 2010*, Springer LNCS 6056, 177–192, 2010.
34. R. Granger, T. Kleinjung and J. Zumbrägel. Breaking '128-bit Secure' Supersingular Binary Curves (or how to solve discrete logarithms in $\mathbb{F}_{2^{4\cdot 1223}}$ and $\mathbb{F}_{2^{12\cdot 367}}$). In *Cryptology ePrint Archive, Report 2014/119*, `http://eprint.iacr.org/2014/119.pdf`.
35. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT 2006*, Springer LNCS 4284, 444–459, 2006.
36. J. Groth. Fully anonymous group signatures without random oracles. In *ASIACRYPT 2007*, Springer LNCS 4833, 164–180, 2007.
37. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In SIAM Journal on Computing, volume 41(5), 1193–1232, 2012.
38. C. Hlauschek, J. Black, G. Vigna and C. Kruegel. Limited-linkable Group Signatures with Distributed-Trust Traceability. In *Technical Report*, Vienna University of Technology, 2012. `http://www.iseclab.org/people/haku/tr-llgroupsig12.pdf`.
39. S.A. Kakvi. Efficient fully anonymous group signatures based on the Groth group signature scheme. Masters thesis, University College London, 2010. `http://www5.rz.rub.de:8032/mam/foc/content/publ/thesis_kakvi10.pdf`.

40. A. Kiayias, Y. Tsiounis and M. Yung. Traceable Signatures. In *EUROCRYPT 2004*, Springer LNCS 3027, 571–589, 2004.
41. A. Kiayias and M. Yung. Group Signatures with Efficient Concurrent Join. In *EUROCRYPT 2005*, Springer LNCS 3494, 198–214, 2005.
42. E. Kiltz. Chosen-Ciphertext Security from Tag-Based Encryption. In *TCC 2006*, Springer LNCS 3876, 581–600, 2006.
43. B. Libert, T. Peters and M. Yung. Scalable Group Signatures with Revocation. In *EUROCRYPT 2012*, Springer LNCS 7237, 609–627, 2012.
44. B. Libert, T. Peters and M. Yung. Group Signatures with Almost-for-Free Revocation. In *CRYPTO 2012*, Springer LNCS 7417, 571–589, 2012.
45. M. Manulis. Democratic group signatures: on an example of joint ventures. In *ASIACCS 2006*, ACM, 365–365, 2006.
46. L. Nguyen and R. Safavi-Naini. Efficient and Provably Secure Trapdoor-Free Group Signature Schemes from Bilinear Pairings. In *ASIACRYPT 2004*, Springer LNCS 3329, 372–386, 2004.
47. A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *FOCS 1999*, 543–553, 1999.
48. Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, T. Matsuda and K. Omote. Group Signatures with Message-Dependent Opening. In *PAIRING 2012*, Springer LNCS 7708, 270–294, 2013.
49. Y. Sakai, J. C. N. Schuldt, K. Emura, G. Hanaoka and K. Ohta. On the Security of Dynamic Group Signatures: Preventing Signature Hijacking. In *PKC 2012*, Springer LNCS 7293, 715–732, 2012.
50. A. Shamir. How to share a secret. In *Communications of the ACM*, 22:11, 612–613, 1979.
51. D. Zheng, X. Li, C. Ma, K. Chen and J. Li. Democratic Group Signatures with Threshold Traceability. In *Cryptology ePrint Archive, Report 2008/112*, `http://eprint.iacr.org/2008/112.pdf`.

A Security of Distributed Tag-Based Encryption

Here we define the security requirements of a distributed tag-based encryption.

- Selective-Tag Indistinguishability against Adaptive Chosen Ciphertext Attacks (ST-IND-CCA): This requires that for all $\lambda, n \in \mathbb{N}$, for all polynomial-time adversaries $\mathcal{A}$ the advantage

$$\mathsf{Adv}^{\text{ST-IND-CCA}}_{\mathcal{DTBE},\mathcal{A},n}(\lambda) := \left|\Pr[\mathsf{Exp}^{\text{ST-IND-CCA-1}}_{\mathcal{DTBE},\mathcal{A},n}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\text{ST-IND-CCA-0}}_{\mathcal{DTBE},\mathcal{A},n}(\lambda) = 1]\right|$$

  is negligible in λ, where the game is shown in Fig. 9. In the game, $\mathsf{ShareDec}^{t^*}$ returns $\perp$ if queried on (t^*, C^*_{dtbe}). A weaker variant of the ST-IND-CCA definition, which we use here, is called *Selective-Tag weak Indistinguishability against Adaptive Chosen Ciphertext Attacks (ST-wIND-CCA)* [6], which is analogous to that defined by Kiltz [42] for traditional tag-based encryption, is when in the above game $\mathsf{ShareDec}^{t^*}$ rejects any query on $(t^*, \cdot)$. Note that by eliminating the requirement that the adversary outputs the target tag beforehand, one obtains stronger variants of the above two notions.
- Decryption Consistency (DEC-CON): This requirement is similar to that defined in [31] for traditional public key encryption and is stronger than the selective-tag variant defined in [6]. This requires that for all $\lambda, n \in \mathbb{N}$, for all PPT adversaries $\mathcal{A}$, the advantage $\mathsf{Adv}^{\text{DEC-CON}}_{\mathcal{DTBE},\mathcal{A},n}(\lambda) := \Pr[\mathsf{Exp}^{\text{DEC-CON}}_{\mathcal{DTBE},\mathcal{A},n}(\lambda) = 1]$ is negligible in λ, where the game is defined in Fig. 10.

Experiment: $\mathsf{Exp}^{\text{ST-IND-CCA-b}}_{\mathcal{DTBE},\mathcal{A},n}(\lambda)$:

- $(S \subset [n], t^*, \mathsf{st}_{\mathsf{init}}) \leftarrow \mathcal{A}_{\mathsf{init}}\left(1^\lambda, n\right)$, where $|S| \leq n-1$.
- $(\mathsf{pk}, \{\mathsf{svk}_i\}_{i=1}^n, \{\mathsf{sk}_i\}_{i=1}^n) \leftarrow \mathsf{Setup}(1^\lambda, n)$.
- $(m_0, m_1, \mathsf{st}_{\mathsf{find}}) \leftarrow \mathcal{A}_{\mathsf{find}}\left(\mathsf{st}_{\mathsf{init}}, \mathsf{pk}, \{\mathsf{svk}_i\}_{i=1}^n, \{\mathsf{sk}_i\}_{i\in S} : \mathsf{ShareDec}^{t^*}(\mathsf{pk}, \mathsf{sk}_i, \cdot, \cdot)\right)$, where $|m_0| = |m_1|$.
- $C^*_{\text{dtbe}} \leftarrow \mathsf{Enc}(\mathsf{pk}, t^*, m_b)$.
- $b^* \leftarrow \mathcal{A}_{\mathsf{guess}}\left(\mathsf{st}_{\mathsf{find}}, C^*_{\text{dtbe}} : \mathsf{ShareDec}^{t^*}(\mathsf{pk}, \mathsf{sk}_i, \cdot, \cdot)\right)$.
- Return b^*.

Fig. 9. The ST-IND-CCA security game for distributed tag-based encryption

Experiment: $\mathsf{Exp}^{\text{DEC-CON}}_{\mathcal{DTBE},\mathcal{A},n}(\lambda)$:

- $(\mathsf{pk}, \{\mathsf{svk}_i\}_{i=1}^n, \{\mathsf{sk}_i\}_{i=1}^n) \leftarrow \mathsf{Setup}(1^\lambda, n)$.
- $(t, C_{\text{dtbe}}, \{\nu_i\}_{i=1}^n, \{\nu'_i\}_{i=1}^n) \leftarrow \mathcal{A}(\mathsf{pk}, \{\mathsf{svk}_i\}_{i=1}^n, \{\mathsf{sk}_i\}_{i=1}^n)$.
- If $\mathsf{IsValid}(\mathsf{pk}, t, C_{\text{dtbe}}) = 0$ Then Return 0.
- If $\exists i \in [n]$ s.t. $\mathsf{ShareVerify}(\mathsf{pk}, \mathsf{svk}_i, t, C_{\text{dtbe}}, \nu_i) = 0$ or $\mathsf{ShareVerify}(\mathsf{pk}, \mathsf{svk}_i, t, C_{\text{dtbe}}, \nu'_i) = 0$ Then Return 0.
- If $\mathsf{Combine}(\mathsf{pk}, \{\mathsf{svk}_i\}_{i=1}^n, \{\nu_i\}_{i=1}^n, C_{\text{dtbe}}, t) \neq \mathsf{Combine}(\mathsf{pk}, \{\mathsf{svk}_i\}_{i=1}^n, \{\nu'_i\}_{i=1}^n, C_{\text{dtbe}}, t)$ Then Return 1.
- Return 0.

Fig. 10. The decryption consistency game for distributed tag-based encryption

B Proof of Theorem 1

Proof. **Lemma 1.** *The construction is selective-tag weakly indistinguishable against adaptive chosen ciphertext attacks.*

Proof. We show if there exists an adversary $\mathcal{A}$ winning the ST-wIND-CCA game, we can construct an adversary $\mathcal{B}$ that breaks the $\text{XDLIN}_{\mathbb{G}}$ assumption such that

$$\mathsf{Adv}^{\text{ST-wIND-CCA}}_{\mathcal{DTBE},\mathcal{A},n}(\lambda) \leq \mathsf{Adv}^{\text{XDLIN}_{\mathbb{G}}}_{\mathcal{B}}(\lambda)$$

Adversary $\mathcal{B}$ gets $\mathcal{P}$ and the tuple $(H, U, V, H^r, V^s, U^t, \tilde{H}, \tilde{U}, \tilde{V}, \tilde{H}^r, \tilde{V}^s) \in \mathbb{G}^6 \times \tilde{\mathbb{G}}^5$ from its game and its aim is to decide whether or not $t = r+s$. It starts $\mathcal{A}$ to get the set S and the target tag t^*. Adversary $\mathcal{B}$ then selects an index j from the set $[n] \setminus S$. For all servers $i \in [n] \setminus \{j\}$, $\mathcal{B}$ chooses $\alpha_i, \beta_i \leftarrow \mathbb{Z}_p$ that will be used as the secret key for server i. The corresponding server verification key is computed as $(\tilde{U}_i, \tilde{V}_i) := (\tilde{H}^{\alpha_i}, \tilde{V}^{\beta_i})$. For the remaining server j, $\mathcal{B}$ computes its verification key as $(\tilde{U}_j, \tilde{V}_j) := (\frac{\tilde{U}}{\prod_{i=1,i\neq j}^n \tilde{U}_i}, \frac{\tilde{U}}{\prod_{i=1,i\neq j}^n \tilde{V}_i})$. It also chooses $d_1, d_2 \in \mathbb{Z}_p$ and computes $(W, \tilde{W}) := \left(U^{-t^*} H^{d_1}, \tilde{U}^{-t^*} \tilde{H}^{d_1}\right)$, $(Z, \tilde{Z}) := \left(U^{-t^*} V^{d_2}, \tilde{U}^{-t^*} \tilde{V}^{d_2}\right)$. It then sets $\mathsf{pk} := (\mathcal{P}, H, \tilde{H}, U, \tilde{U}, V, \tilde{V}, W, \tilde{W}, Z, \tilde{Z})$ and sends pk, $\{\mathsf{svk}\}_{i=1}^n$ and $\{\mathsf{sk}_i\}_{i\in S}$ to $\mathcal{A}$.

When answering $\mathsf{ShareDec}^{t^*}$ queries by server $i \neq j$ on any tag $t \neq t^*$, $\mathcal{B}$ first checks that the ciphertext is valid and if so, it returns $(C_{i,1} := C_1^{\alpha_i}, C_{i,2} := C_2^{\beta_i})$. For the j-th server, after verifying that the ciphertext is valid, it returns $\left(C_{j,1} := \frac{(C_4 C_1^{-d_1})^{\frac{1}{t-t^*}}}{\prod_{i=1,i\neq j}^n C_1^{\alpha_i}}, C_{j,2} := \frac{(C_5 C_2^{-d_2})^{\frac{1}{t-t^*}}}{\prod_{i=1,i\neq j}^n C_2^{\beta_i}}\right)$. This is a valid decryption share satisfying $e(C_{j,1}, \tilde{H}) = e(C_1, \tilde{U}_j)$ and $e(C_{j,2}, \tilde{V}) = e(C_2, \tilde{V}_j)$.

In the challenge phase, $\mathcal{B}$ receives two messages M_0 and M_1. It randomly chooses $b \leftarrow \{0, 1\}$ and responds with a challenge ciphertext $(C_1, C_2, C_3, C_4, C_5) := (H^r, V^s, H^{rd_1}, V^{sd_2}, M_b U^t)$. Note this is a valid ciphertext satisfying $e(C_1, \tilde{U}^t \tilde{W}) = e(C_4, \tilde{H})$ and $e(C_2, \tilde{U}^t \tilde{Z}) = e(C_5, \tilde{V})$.

Any further decryption queries made by $\mathcal{A}$ are answered by $\mathcal{B}$ as above. Eventually, when $\mathcal{A}$ outputs its guess b^*, $\mathcal{B}$ outputs b^*.

Lemma 2. *The construction satisfies decryption consistency.*

Proof. Suppose that an adversary $\mathcal{A}$ outputs two valid sets $\{(C_{i,1}, C_{i,2})\}_{i=1}^n$ and $\{(C'_{i,1}, C'_{i,2})\}_{i=1}^n$ of decryption shares of a ciphertext C_{dtbe}. For it to win, both sets must contain valid shares, i.e. all $2n$ shares must be accepted by the $\mathsf{ShareVerify}$ algorithm. Thus, we must have $e(C_{i,1}, \tilde{H}) = e(C_1, \tilde{U}_i)$ and $e(C_{i,2}, \tilde{V}) = e(C_2, \tilde{V}_i)$ for $i = 1, \ldots, n$. This means we must have $\prod_{i=1}^n C_{i,1} = C_1^u$ and $\prod_{i=1}^n C_{i,2} = C_2^v$. Similarly, we must have $e(C'_{i,1}, \tilde{H}) = e(C_1, \tilde{U}_i)$ and $e(C'_{i,2}, \tilde{V}) = e(C_2, \tilde{V}_i)$ for $i = 1, \ldots, n$. This means that we must have $\prod_{i=1}^n C'_{i,1} = C_1^u$ and $\prod_{i=1}^n C'_{i,2} = C_2^v$. Thus, we must have $C_{i,1} = C'_{i,1}$ and $C_{i,2} = C'_{i,2}$ for $i = 1, \ldots, n$ which contradicts the winning condition. This concludes the proof.

C Instantiations Details

C.1 Instantiation I

Here we have $\mathsf{cert}_{\mathsf{uid}} = (\tilde{\Omega}_1, \Omega_2, \Omega_3) \in \tilde{\mathbb{G}} \times \mathbb{G}^2$ and the user has the re-randomization token $\Omega_4 \in \mathbb{G}$, $C_{\text{dtbe}} = (C_1, C_2, C_3, C_4, C_5) \in \mathbb{G}^5$, $\mathsf{pk}_{\mathsf{GM}} = \tilde{X}$, $\mathsf{pk}_{\mathsf{uid}} \in \mathbb{G}$, and $\mathsf{pk}_{\mathcal{DTBE}} = (\mathcal{P}, H, \tilde{H}, U, \tilde{U}, V, \tilde{V}, W, \tilde{W}, Z, \tilde{Z})$.

- To prove $\mathcal{DS}_1.\mathsf{Verify}(\mathsf{pk}_{\mathsf{GM}}, \underline{\mathsf{pk}_{\mathsf{uid}}}, \underline{\mathsf{cert}_{\mathsf{uid}}}) = 1$, the user re-randomizes $\mathsf{cert}_{\mathsf{uid}}$ using the token Ω_4 to obtain a new membership certificate $\mathsf{cert}'_{\mathsf{uid}} = (\tilde{\Omega}'_1, \Omega'_2, \Omega'_3)$ and then proves the following linear equations

$$e(\underline{\Omega'_2}, \tilde{\Omega}'_1) = e(\underline{\mathsf{pk}_{\mathsf{uid}}}, \tilde{X})e(\underline{F}, \tilde{G}) \qquad e(\underline{\Omega'_3}, \tilde{\Omega}'_1) = e(\underline{\Omega'_2}, \tilde{X})e(\underline{G}, \tilde{G}) \qquad \underline{G} - G = 0 \qquad \underline{F} - F = 0$$

- To prove $\mathcal{WDS}.\mathsf{Verify}(\underline{\mathsf{pk}_{\mathsf{uid}}}, \mathcal{H}(\mathsf{otsvk}), \underline{\tilde{\sigma}}) = 1$ where $\mathsf{pk}_{\mathsf{uid}} \in \mathbb{G}$, $\tilde{\sigma} \in \tilde{\mathbb{G}}$. The signer proves

$$e(\underline{\mathsf{pk}_{\mathsf{uid}}}, \underline{\tilde{\sigma}})e(G^{\mathcal{H}(\mathsf{otsvk})}, \underline{\tilde{\sigma}})e(\underline{G}, \tilde{G}) = 1$$

- To prove that $\mathcal{DTBE}.\mathsf{Enc}(\mathsf{pk}_{\mathcal{DTBE}}, \mathcal{H}(\mathsf{otsvk}), \underline{\mathsf{pk}_{\mathsf{uid}}}; (\underline{r_1}, \underline{r_2})) = C_{\text{dtbe}}$, the signer proves C_1, C_2 and C_3 were computed correctly and the rest can be verified by checking that $e(C_4, \tilde{H}) = e(C_1, \tilde{U}^{\mathcal{H}(\mathsf{otsvk})}\tilde{W})$ and $e(C_5, \tilde{V}) = e(C_2, \tilde{U}^{\mathcal{H}(\mathsf{otsvk})}\tilde{Z})$. Thus, this requires proving

$$C_1 = H^{\underline{r_1}} \qquad C_2 = V^{\underline{r_2}} \qquad C_3 = \underline{\mathsf{pk}_{\mathsf{uid}}} U^{\underline{r_1}} U^{\underline{r_2}}$$

Note that all the proven equations are simulatable and hence the corresponding proofs are zero-knowledge. The total size of the signature is $\mathbb{G}^{24} + \tilde{\mathbb{G}}^{21} + \mathbb{Z}_p^5$.

C.2 Instantiation II

The only difference from Instantiation I is the manner in which $\mathsf{cert}_{\mathsf{uid}}$ is verified. We have $\mathsf{pk}_{\mathsf{uid}} \in \mathbb{G}$, $\mathsf{cert}_{\mathsf{uid}} = (\Omega_1, \Omega_2, \tilde{\Omega}_3, \Omega_4)$ and $\mathsf{pk}_{\mathsf{GM}} = (\tilde{W}, \tilde{X}, \tilde{Y}_1, \tilde{Y}_2)$.

To prove $\mathcal{DS}_1.\mathsf{Verify}(\mathsf{pk}_{\mathsf{GM}}, \underline{\mathsf{pk}_{\mathsf{uid}}}, \underline{\mathsf{cert}_{\mathsf{uid}}}) = 1$, the user proves the following equations

$$e(\underline{\Omega_2}, \underline{\tilde{\Omega}_3}) = e(\underline{G}, \tilde{G}) \qquad e(\underline{G}, \tilde{X}) = e(\underline{\Omega_2}, \tilde{W})e(\underline{\Omega_4}, \tilde{G})e(\underline{\Omega_1}, \tilde{Y_1})e(\underline{\mathsf{pk}_{\mathsf{uid}}}, \tilde{Y_2}) \qquad \underline{G} - G = 0$$

The rest of the proofs are the same as in Instantiation I. Again, all the proven equations are simulatable and hence the corresponding proofs are zero-knowledge. The size of the signature is $\mathbb{G}^{28} \times \tilde{\mathbb{G}}^{24} \times \mathbb{Z}_p^3$.

D Proof of Theorem 2

Proof. Correctness of the construction follows from that of the underlying building blocks.

Lemma 3. *If $\mathcal{NIZK}$ is zero-knowledge, the tag-based encryption scheme $\mathcal{DTBE}$ is selective-tag weakly IND-CCA secure, the one-time signature $\mathcal{OTS}$ is strongly existentially unforgeable, and the hash function $\mathcal{H}$ is collision-resistant then the construction satisfies strong full anonymity against full-key exposure.*

Proof. We show that if there exists an adversary $\mathcal{B}$ which breaks the anonymity of the construction, we can build adversaries $\mathcal{A}_1$ against the collision-resistance of the hash function $\mathcal{H}$, $\mathcal{A}_2$ against the strong unforgeability of the one-time signature $\mathcal{OTS}$, $\mathcal{A}_3$ against the NIZK property of the proof system

The collision-resistance of $\mathcal{H}$ ensures that $\mathcal{B}$ has a negligible probability in finding otsvk' s.t. $\mathcal{H}(\mathsf{otsvk}')$ collides with the tag $\mathcal{H}(\mathsf{otsvk}^*)$ we use for the challenge signature. If this is not the case, we can use $\mathcal{B}$ to build an adversary $\mathcal{A}_1$ that breaks the collision-resistance of $\mathcal{H}$.

The strong existential unforgeability of $\mathcal{OTS}$ ensures that $\mathcal{B}$ has a negligible probability in forging a one-time signature under otsvk^* we use in the challenge signature. If this is not the case, we can build an adversary $\mathcal{A}_2$ that wins the strong unforgeability game of $\mathcal{OTS}$.

We start $\mathcal{NIZK}$ in the simulation setting which means that the proof π is now zero-knowledge and hence does not reveal any information about the witness.

We now proceed to show how to use $\mathcal{B}$ to build an adversary $\mathcal{A}_4$ against the selective-tag weakly IND-CCA security of $\mathcal{DTBE}$. Adversary $\mathcal{A}_4$ starts $\mathcal{B}$ to get the list BTL of the tracing managers it would like to corrupt. Adversary $\mathcal{A}_4$ runs the GKg algorithm where it starts by randomly choosing a key pair $(\mathsf{otsvk}^*, \mathsf{otssk}^*)$ for $\mathcal{OTS}$ that it will use in answering $\mathcal{B}$'s challenge signature. This needed to be chosen beforehand because the tag-based encryption scheme is only selective-tag secure. $\mathcal{A}_4$ sends $(\mathcal{H}(\mathsf{otsvk}^*), \mathsf{BTL})$ to its own game and gets back $\mathsf{pk}_{\mathcal{DTBE}}$, $\{\mathsf{svk}_i\}_{i\in[\kappa]}$ and $\{\mathsf{sk}_i\}_{i\in\mathsf{BTL}}$. In its game, $\mathcal{A}_4$ has access to a share decryption oracle $\mathsf{ShareDec}^{\mathcal{H}(\mathsf{otsvk}^*)}$ which it can query on any ciphertext under any tag different from the target tag $\mathcal{H}(\mathsf{otsvk}^*)$. $\mathcal{A}_4$ now chooses a simulated string crs for $\mathcal{NIZK}$ and runs the rest of the setup normally. It sends $\mathsf{gpk} := (1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^{\kappa}, \mathcal{H})$, $\mathsf{msk} := \mathsf{sk}_{\mathsf{GM}}$ and $\{\mathsf{sk}_i\}_{i\in\mathsf{BTL}}$ to $\mathcal{B}$.

All $\mathcal{B}$'s queries are answered normally as in Fig. 1 except the following oracles.

- CH_b: To answer the challenge query $\mathsf{CH}_b(\mathsf{uid}_0, \mathsf{uid}_1, m)$, $\mathcal{A}_4$ sends $(\mathsf{pk}_{\mathsf{uid}_0}, \mathsf{pk}_{\mathsf{uid}_1})$ as its challenge in its ST-wIND-CCA game and gets a ciphertext under the tag $\mathcal{H}(\mathsf{otsvk}^*)$ of either the plaintext $\mathsf{pk}_{\mathsf{uid}_0}$ or $\mathsf{pk}_{\mathsf{uid}_1}$ which it needs to distinguish. $\mathcal{A}_4$ then constructs the rest of the challenge signature by simulating the proof π and signing the whole thing with otssk^* to obtain σ_{ots}.
- $\mathsf{TraceShare}$: $\mathcal{A}_4$ just uses its $\mathsf{ShareDec}^{\mathcal{H}(\mathsf{otsvk}^*)}$ oracle to get the decryption share of C_{dtbe} contained within the signature. Note that since we have chosen otsvk^* uniformly at random and since we already eliminated any case where any signature sent to $\mathsf{TraceShare}$ uses the same tag as that we used in the challenge signature, such a query will be accepted by $\mathcal{A}_4$'s $\mathsf{ShareDec}^{\mathcal{H}(\mathsf{otsvk}^*)}$ oracle because the tag is different from the target tag used in the challenge signature.
- Trace: To answer this, $\mathcal{A}_4$ just uses its $\mathsf{ShareDec}^{\mathcal{H}(\mathsf{otsvk}^*)}$ oracle to get all the decryption shares $\{\nu_i\}_{i=1}^{\kappa}$ of C_{dtbe} contained within the signature.

Finally, when $\mathcal{B}$ outputs its guess, $\mathcal{A}_4$'s output is that of $\mathcal{B}$

Lemma 4. *The construction is non-frameable if $\mathcal{NIZK}$ is sound, the hash function $\mathcal{H}$ is collision-resistant, and the digital signatures $\mathcal{DS}_2$ and $\mathcal{WDS}$ as well as the one-time signature $\mathcal{OTS}$ are existentially unforgeable.*

Proof. We show that if there exists an adversary $\mathcal{B}$ that wins the non-frameability game then we can build adversaries $\mathcal{A}_1$ against the unforgeability of signature scheme $\mathcal{DS}_2$, adversary $\mathcal{A}_2$ against the unforgeability of the weak digital signature $\mathcal{WDS}$, adversary $\mathcal{A}_3$ against the strong unforgeability of the one-time signature scheme $\mathcal{OTS}$, $\mathcal{A}_4$ against the collision-resistance of the hash function $\mathcal{H}$ and $\mathcal{A}_5$ against the soundness of $\mathcal{NIZK}$ respectively, such that

$$\begin{aligned}\mathsf{Adv}^{\text{Non-Frame}}_{\mathcal{DGSDT},\mathcal{B},\kappa}(\lambda) \leq & \gamma(\lambda)\cdot\left(\mathsf{Adv}^{\text{Unfor}}_{\mathcal{DS}_2,\mathcal{A}_1}(\lambda) + \mathsf{Adv}^{\text{Unfor}}_{\mathcal{WDS},\mathcal{A}_2}(\lambda)\right) + \delta(\lambda)\cdot\mathsf{Adv}^{\text{Unfor}}_{\mathcal{OTS},\mathcal{A}_3}(\lambda) \\ & + \mathsf{Adv}^{\text{Coll}}_{\mathcal{H},\mathcal{A}_4}(\lambda) + \mathsf{Adv}^{\text{Sound}}_{\mathcal{NIZK},\mathcal{A}_5}(\lambda),\end{aligned}$$

where $\gamma(\lambda)$ and $\delta(\lambda)$ are polynomials in λ representing an upper bound on the number of honest users and signing queries, respectively, $\mathcal{B}$ is allowed to make in the game.

We start by instantiating the proof system $\mathcal{NIZK}$ in the soundness setting and hence the adversary cannot frame users by faking proofs for a false statement. Also, we have by the collision-resistance of

the hash function $\mathcal{H}$ that $\mathcal{B}$ has a negligible probability in finding two different one-time signature keys $\mathsf{otsvk} \neq \mathsf{otsvk}'$ s.t. $\mathcal{H}(\mathsf{otsvk}) = \mathcal{H}(\mathsf{otsvk}')$. If this is not the case, we can use $\mathcal{B}$ to build an adversary $\mathcal{A}_4$ that breaks the collision-resistance of $\mathcal{H}$. Thus, from now on we assume that there are no hash collisions.

- **Adversay $\mathcal{A}_1$:** Adversary $\mathcal{A}_1$ gets the signature scheme's verification key pk^* from its game and has access to a Sign oracle. Adversary $\mathcal{A}_1$ starts by running $(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathcal{NIZK}.\mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \{\mathsf{sk}_i\}_{i=1}^\kappa) \leftarrow \mathcal{TPKE}.\mathsf{Setup}(1^\lambda, \kappa)$ and $(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}}) \leftarrow \mathcal{DS}_1.\mathsf{KeyGen}(1^\lambda)$. It then forwards $\mathsf{gpk} := (1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$, $\mathsf{msk} := \mathsf{sk}_{\mathsf{GM}}$ and $\{\mathsf{tsk}_i\}_{i=1}^\kappa := \{\mathsf{sk}_i\}_{i=1}^\kappa$ to $\mathcal{B}$.
  Adversary $\mathcal{A}_1$ randomly chooses $i \leftarrow \{1, \ldots, \gamma(\lambda)\}$ and guesses that $\mathcal{B}$ will attempt to frame user i. Thus, we have a probability $1/\gamma(\lambda)$ of guessing the correct user.
  All $\mathcal{B}$'s queries are answered normally as in Fig. 1 except the following oracles.
  - SndU: For all honest users other than user i, $\mathcal{A}_1$ chooses both the personal public/secret key pair $(\mathbf{upk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}])$ and the signing public/secret key pair $(\mathsf{pk}_{\mathsf{uid}}, \mathsf{sk}_{\mathsf{uid}})$ of the users itself. However, for user i, $\mathcal{A}_1$ sets $\mathbf{upk}[i] = \mathsf{pk}^*$ (i.e. the verification key it got from its $\mathcal{DS}_2$ unforgeability game and hence it does not know the corresponding secret key $\mathbf{usk}[i]$) and chooses the signing public/secret key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ for the user. To obtain the corresponding signature sig on pk_i, it queries its signing oracle.
    If in the game $\mathcal{B}$ asks for a RevealU query on user i, $\mathcal{A}_1$ aborts.

  Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{A}_1$ aborts if the framed user is different from i. Otherwise, $\mathcal{A}_1$ returns $\mathsf{pk}^*_{\mathsf{uid}}$ and $\mathsf{sig}^*_{\mathsf{uid}}$ as its forgery in its game.
  By the existential unforgeability of $\mathcal{DS}_2$, this only happens with a negligible probability.
- **Adversary $\mathcal{A}_2$:** Similarly to $\mathcal{A}_1$, $\mathcal{A}_2$ randomly chooses $i \leftarrow \{1, \ldots, \gamma(\lambda)\}$ and guesses that $\mathcal{B}$ will attempt to frame user i. Thus, we have a probability of $1/\gamma(\lambda)$ of guessing the correct user. Let $\zeta(\lambda)$ be an upper bound on the number of signing queries by user i $\mathcal{B}$ is allowed to make. $\mathcal{A}_2$ randomly choose $\zeta(\lambda)$ key pairs $\{(\mathsf{otsvk}_j, \mathsf{otssk}_j)\}_{j=1}^{\zeta(\lambda)}$ for the one-time signature $\mathcal{OTS}$. It then forwards $\{\mathcal{H}(\mathsf{otsvk}_j)\}_{j=1}^{\zeta(\lambda)}$ to its game to get the verification key pk^* and signatures $\{\sigma_j\}_{j=1}^{\zeta(\lambda)}$ on $\{\mathcal{H}(\mathsf{otsvk}_j)\}_{j=1}^{\zeta(\lambda)}$.
  $\mathcal{A}_2$ starts $\mathcal{B}$ with input $\mathsf{gpk} := (1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$, $\mathsf{msk} := \mathsf{sk}_{\mathsf{GM}}$ and $\{\mathsf{tsk}_i\}_{i=1}^\kappa := \{\mathsf{sk}_i\}_{i=1}^\kappa$. All $\mathcal{B}$'s queries are answered normally as in Fig. 1 except the following oracles.
  - SndU: For all honest users other than user i, $\mathcal{A}_2$ chooses both the personal key pair $(\mathbf{upk}[\mathsf{uid}], \mathbf{usk}[\mathsf{uid}])$ and the signing public/secret key pair $(\mathsf{pk}_{\mathsf{uid}}, \mathsf{sk}_{\mathsf{uid}})$ of the user by itself. For user i, $\mathcal{A}_2$ chooses the personal public/secret key pair $(\mathbf{upk}[i], \mathbf{usk}[i])$ and sets the user's public key $\mathsf{pk}_i = \mathsf{pk}^*$ (i.e. the verification key it got from its $\mathcal{WDS}$ unforgeability game) and hence it does not know the corresponding secret key sk_i. Also, $\mathcal{A}_2$ uses $\mathbf{usk}[i]$ to generate sig.
    If in the game $\mathcal{B}$ issues a RevealU query on user i, $\mathcal{A}_2$ aborts.
    Note that all $\zeta(\lambda)$ signing queries on behalf of user i can be answered by $\mathcal{A}_2$ bu using the signatures it received from its own signing oracle.

  Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{A}_2$ aborts if the framed user is different from user i. Otherwise, $\mathcal{A}_2$ uses $\mathcal{NIZK}$'s extraction key xk to extract the signature σ^* on $\mathcal{H}(\mathsf{otsvk}^*)$ that it did not query its signing oracle on from the proof π^* and returns $(\sigma^*, \mathcal{H}(\mathsf{otsvk}^*))$ as its forgery in its own game.
  By the existential unforgeability against weak chosen-message attack of $\mathcal{WDS}$, this only happens with a negligible probability.
- **Adversary $\mathcal{A}_3$:** Adversary $\mathcal{A}_3$ gets the one-time scheme's verification key otsvk^* from its game and has access to a one-time Sign oracle that it can query on a message of its choice.
  $\mathcal{A}_3$ runs $(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathcal{NIZK}.\mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \{\mathsf{sk}_i\}_{i=1}^\kappa) \leftarrow \mathcal{TPKE}.\mathsf{Setup}(1^\lambda, \kappa)$ and $(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}}) \leftarrow \mathcal{DS}_1.\mathsf{KeyGen}(1^\lambda)$. It then forwards $\mathsf{gpk} := (1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$, $\mathsf{msk} := \mathsf{sk}_{\mathsf{GM}}$ and $\{\mathsf{tsk}_i\}_{i=1}^\kappa := \{\mathsf{sk}_i\}_{i=1}^\kappa$ to $\mathcal{B}$.

Adversary $\mathcal{A}_3$ randomly chooses $j \leftarrow \{1, \ldots, \delta(\lambda)\}$ and guesses that $\mathcal{B}$'s forgery will involve forging a one-time signature that verifies under otsvk^* used in answering the j-th signing query.
All $\mathcal{B}$'s queries are answered normally as in Fig. 1 except the following oracles.

- Sign: When asked for the i-th Sign query (uid, m), if $j \neq i$, $\mathcal{A}_3$ chooses a fresh key pair $(\mathsf{otsvk}, \mathsf{otssk})$ for the one-time signature scheme and answers the query by itself. If $j = i$, $\mathcal{A}_3$ encrypts $\mathsf{pk}_{\mathsf{uid}}$ using $\mathcal{H}(\mathsf{otsvk}^*)$ as a tag to obtain C_{dtbe} and generates the proof π. It then forwards $(m, \pi, C_{\text{dtbe}}, \mathsf{otsvk}^*)$ as the message to its one-time signing oracle to get a one-time signature σ_{ots}. $\mathcal{A}_3$ then sends the signature $\Sigma := (\sigma_{\text{ots}}, \pi, C_{\text{dtbe}}, \mathsf{otsvk}^*)$ to $\mathcal{B}$.

Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{A}_2$ aborts if the $\mathcal{B}$'s forgery did not involve forging a one-time signature that verifies w.r.t otsvk^* it got from its game. The probability that $\mathcal{B}$ forges a one-time signature that verifies w.r.t the same otsvk^* is $\frac{1}{\delta(\lambda)}$.
By the strong existential unforgeability of the one-time signature $\mathcal{OTS}$, this only happens with a negligible probability.

This concludes the proof.

Lemma 5. *The construction is traceable if the $\mathcal{NIZK}$ proof system is sound, and the digital signature scheme $\mathcal{DS}_1$ is existentially unforgeable.*

Proof. We show that if there exists an adversary $\mathcal{B}$ that wins the traceability game then we can build adversaries $\mathcal{A}_1$ against the unforgeability of signature scheme $\mathcal{DS}_1$ and adversary $\mathcal{A}_2$ against the soundness of $\mathcal{NIZK}$, respectively, such that

$$\mathsf{Adv}^{\text{Trace}}_{\mathcal{DGSDT},\mathcal{B},\kappa}(\lambda) \leq \mathsf{Adv}^{\text{Unfor}}_{\mathcal{DS}_1,\mathcal{A}_1}(\lambda) + \mathsf{Adv}^{\text{Sound}}_{\mathcal{NIZK},\mathcal{A}_2}(\lambda).$$

We start by instantiating the proof system $\mathcal{NIZK}$ in the soundness setting and hence the adversary cannot break traceability by faking proofs for a false statement.

- **Adversay $\mathcal{A}_1$:** Adversary $\mathcal{A}_1$ gets the signature scheme's verification key pk^* from its game and has access to a Sign oracle that it can query on messages its choice. Adversary $\mathcal{A}_1$ starts by running $(\mathsf{crs}, \mathsf{xk}) \leftarrow \mathcal{NIZK}.\mathsf{Setup}(1^\lambda)$, $(\mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \{\mathsf{sk}_i\}_{i=1}^\kappa) \leftarrow \mathcal{TPKE}.\mathsf{Setup}(1^\lambda, \kappa)$ and setting $\mathsf{pk}_{\mathsf{GM}} := \mathsf{pk}^*$ and hence it does not know the corresponding secret key $\mathsf{sk}_{\mathsf{GM}}$. $\mathcal{A}_1$ forwards $\mathsf{gpk} := (1^\lambda, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^\kappa, \mathcal{H})$ and $\{\mathsf{tsk}_i\}_{i=1}^\kappa := \{\mathsf{sk}_i\}_{i=1}^\kappa$ to $\mathcal{B}$.
  All $\mathcal{B}$'s queries are answered normally as in Fig. 1 except the following oracles.
  - SndM and AddU: Since $\mathcal{A}_1$ does not know the group manager's secret key $\mathsf{sk}_{\mathsf{GM}}$, in order to generate the membership certificate $\mathsf{cert}_{\mathsf{uid}}$, $\mathcal{A}_1$ forwards $\mathsf{pk}_{\mathsf{uid}}$ to its sign oracle and sends back the signature $\mathsf{cert}_{\mathsf{uid}}$ to $\mathcal{B}$ as the membership certificate.

  Eventually, when $\mathcal{B}$ outputs its forgery, $\mathcal{A}_1$ uses $\mathcal{NIZK}$'s extraction key xk to extract $\mathsf{pk}^*_{\mathsf{uid}}$ and $\mathsf{cert}^*_{\mathsf{uid}}$ and returns $(\mathsf{pk}^*_{\mathsf{uid}}, \mathsf{cert}^*_{\mathsf{uid}})$ as its forgery in its game. By the existential unforgeability of $\mathcal{DS}_1$, this only happens with a negligible probability.

This concludes the proof.

Lemma 6. *The construction satisfies tracing soundness if the $\mathcal{DTBE}$ scheme satisfies decryption consistency.*

Proof. We show that if there exists an adversary $\mathcal{B}$ that wins the tracing soundness game then we can build an adversary $\mathcal{A}$ that breaks the decryption consistency requirement of the distributed tag-based encryption scheme $\mathcal{DTBE}$ such that

$$\mathsf{Adv}^{\text{Trace-Sound}}_{\mathcal{DGSDT},\mathcal{B},\kappa}(\lambda) \leq \mathsf{Adv}^{\text{DEC-CON}}_{\mathcal{DTBE},\mathcal{A},\kappa}(\lambda).$$

Adversary $\mathcal{A}$ gets $(\mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^{\kappa}, \{\mathsf{sk}_i\}_{i=1}^{\kappa})$ from its game and runs the rest of the GKg algorithm normally. $\mathcal{A}$ forwards $\mathsf{gpk} := (1^{\lambda}, \mathsf{crs}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathcal{DTBE}}, \{\mathsf{svk}_i\}_{i=1}^{\kappa}, \mathcal{H})$, $\mathsf{msk} := \mathsf{sk}_{\mathsf{GM}}$ and $\{\mathsf{tsk}_i\}_{i=1}^{\kappa} := \{\mathsf{sk}_i\}_{i=1}^{\kappa}$ to $\mathcal{B}$. All $\mathcal{B}$'s queries are answered normally as in Fig. 1. Eventually, when $\mathcal{B}$ halts, $\mathcal{A}$ returns $\{\nu_{1,i}\}_{i=1}^{\kappa}$ and $\{\nu_{2,i}\}_{i=1}^{\kappa}$ which are contained within $\theta^*_{\text{Trace}_1}$ and $\theta^*_{\text{Trace}_2}$, respectively.

By the decryption consistency property of the distributed tag-based encryption scheme $\mathcal{DTBE}$, this only happens with a negligible probability.

This concludes the proof.