Published in H. Krawczyk, Ed., *Public Key Cryptography – PKC 2014*, vol. 8383 of *Lecture Notes in Computer Science*, pp. 592–610, Springer, 2014.

Traceable Group Encryption

Benoît Libert[1], Moti Yung[2], Marc Joye[1], and Thomas Peters[3,⋆]

[1] Technicolor
[2] Google Inc. and Columbia University
[3] Université catholique de Louvain

Abstract. Group encryption (GE) is the encryption analogue of group signatures. It allows a sender to verifiably encrypt a message for some certified but anonymous member of a group. The sender is further able to convince a verifier that the ciphertext is a well-formed encryption under some group member's public key. As in group signatures, an opening authority is empowered with the capability of identifying the receiver if the need arises. One application of such a scheme is secure repository at an unknown but authorized cloud server, where the archive is made accessible by a judge order in the case of misbehavior, like a server hosting illegal transaction records (this is done in order to balance individual rights and society's safety). In this work we describe Traceable GE system, a group encryption with refined tracing capabilities akin to those of the primitive of "traceable signatures" (thus, balancing better privacy vs. safety). Our primitive enjoys the properties of group encryption, and, in addition, it allows the opening authority to reveal a user-specific trapdoor which makes it possible to publicly trace all the ciphertexts encrypted for that user without harming the anonymity of other ciphertexts. In addition, group members are able to non-interactively prove that specific ciphertexts are intended for them or not. This work provides rigorous definitions, concrete constructions in the standard model, and security proofs.

Keywords: Group encryption, traceability, anonymity, provable security, standard model.

1 Introduction

Group signatures [10] are a fundamental privacy primitive allowing members of a group to sign messages on behalf of the group while hiding their identity. To deter abuses, an authority is capable of identifying the author of any valid signature using privileged information. Group encryption (GE) is a primitive suggested by Kiayias, Tsiounis and Yung [19], which is the encryption analogue of group signatures [10]. Namely, it allows the sender of a ciphertext to hide the identity of the receiver within a population of certified users —under the control of a group manager (GM)— while providing universally verifiable guarantees

⋆ This author was supported by the CAMUS Walloon Region Project.

that this receiver belongs to the group. If necessary, an opening authority (OA) is empowered with a key allowing it to "open" a ciphertext and pin down the receiver's identity in the same way as group signatures can be opened. Moreover, the system should support a mechanism allowing the sender to convince any verifier that (1) the ciphertext is well-formed and intended for some registered group member who will be able to decrypt; (2) the opening authority can identify the receiver if the need arises; (3) the plaintext satisfies certain properties such as being a witness for some public relation.

As a natural use case, group encryption allows a firewall to block all encrypted emails attempting to enter a network unless they are generated for some certified organization member and they carry a proof of malware-freeness. The GE primitive was also motivated by privacy applications such as anonymous trusted third parties (TTP) or oblivious retriever storage. In optimistic protocols, it allows verifiably encrypting messages to *anonymous* trusted third parties which remain offline most of their lifetime and only wake up when there is a problem to sort out. Group encryption provides a convenient way to hide the identity of users' preferred trusted third party, which can be a privacy-sensitive piece information by itself as it can betray, e.g., the participant's citizenship.

Group encryption also finds applications in cloud storage systems. When encrypting datasets on a remote storage server, the sender can convince this server that the data is intended for some legitimate certified user without disclosing the latter's identity.

As exemplified in [19], group encryption also allows constructing hierarchical group signatures [27], where signers can flexibly specify how a set of trustees should operate to open their signatures.

Here we suggest a primitive extending the group encryption primitive and describe a refined traceability mechanism analogous to the way traceable signatures [18] extend group signatures. Specifically, when a given group member is suspected of conducting illegal activities, the opening authority is able to release a trapdoor allowing anyone to publicly trace ciphertexts encrypted for this member *without affecting the anonymity of other users*. As in the case of traceable signatures, the tracing trapdoor can be distributed to several tracing agents who can proceed in parallel when it comes to search for a given group member's ciphertexts. In contrast, in ordinary GE schemes, this task requires the OA to sequentially operate on all ciphertexts.

Related work. Kiayias, Tsiounis and Yung (KTY) [19] formalized the notion of group encryption and provided a modular design using zero-knowledge proofs, digital signatures, anonymous CCA-secure public-key encryption and commitment schemes. They also gave an efficient instantiation using Paillier's cryptosystem [25] and Camenisch-Lysyanskaya signatures [8]. While efficient, their scheme uses interactive proof systems. It can be made non-interactive using the Fiat-Shamir paradigm [13] at the cost of relying on the random oracle model [4], which is understood to only provide heuristic arguments in terms of security.

Qin *et al.* [26] considered a sort of group encryption mechanism with non-interactive proofs and short ciphertexts. However, they appeal to random oracles

and interactive assumptions in their security analysis. A non-interactive realization in the standard model was put forth by Cathalo, Libert and Yung [9]. More recently, El Aimani and Joye [12] considered more efficient interactive and non-interactive constructions using various optimizations.

As a matter of fact, none of the above solutions makes it possible to trace specific users' ciphertexts and only those ones. If messages encrypted for a specific misbehaving user have to be identified within a collection of, say $n = 100000$ ciphertexts, the opening authority has to open all of these in order to find those it is looking for. This is clearly harmful to the privacy of honest users who lose their anonymity just because they belong to the same group as a rogue user. In [18], Kiayias, Tsiounis and Yung suggested a technique to address this concern in the context of group signatures. To our knowledge, no real encryption analogue of their primitive has been studied so far.

The closest work addressing the problem at hand is that of Izabachène, Pointcheval and Vergnaud [17] who focus on eliminating subliminal channels by means of randomizable encryption. However, their mediated traceable anonymous encryption primitive does not provide all the functionalities we are aiming at. First, their scheme only provides message confidentiality and anonymity against *passive* adversaries, who have no access to decryption oracles at any time. Second, while their constructions enable individual user traceability, they do not provide a mechanism allowing the authority to identify the receiver of a ciphertext in $O(1)$ time. If their scheme is set up for groups of up to n users, their opening algorithm requires $O(n)$ operations in the worst case. Finally, the schemes of [17] provide no method allowing users to claim or disclaim ciphertexts they are the recipients of or not without disclosing their private keys.

Our contribution. This paper suggests a primitive called *traceable group encryption* (TGE) as the direct encryption analogue of traceable signatures, as suggested by Kiayias, Tsiounis and Yung [18]. Beyond the usual functionalities of group encryption, a TGE system allows the opening authority to reveal trapdoors associated with specific group members. These trapdoors enable the recognition of ciphertexts intended for these group members and leak no information about the identity of other ciphertexts' recipients. For example, when an employee leaves a company, the firewall can use a tracing trapdoor to sieve out all incoming ciphertexts encrypted for that former employee without learning anything else. As in the traceable signature scenario [18], this implicit tracing process can be run in parallel by clerks equipped with a copy of the tracing trapdoor.

In addition, similarly to the claiming mechanism of traceable signatures [18], TGE schemes support a procedure whereby group members are able to claim and prove that they are the legitimate receiver of some initially anonymous ciphertexts. Moreover, we further consider the dual problem of allowing group members to disclaim ciphertexts that are *not* encrypted under their public keys (this feature was not part of the original traceable signature model but it can be added on top of it in a modular way). Of course, our security notions explicitly require that group members be unable to falsely claim or disclaim ciphertexts.

The above claiming and disclaiming capabilities can serve in certain applications like cloud storage. While storage servers may require anonymous data retrievers to hold a certificate from some authority, the disclaiming procedure allows group members to convince investigators that they are not the intended recipient of some suspicious ciphertext without revealing their private key.

The first contribution of this paper is to define the primitive and to further provide stringent security definitions for traceable group encryption systems: like its group encryption counterpart [19], our model considers powerful adversaries who have oracle access to the private key functionalities of all users and authorities. As a second contribution, we provide a concrete construction and prove its security in the standard model under non-interactive assumptions. Our system is not just a proof of concept. At the 128-bit security level, ciphertexts and proofs fit within 2.18 and 9.38 kB, respectively. The efficiency is thus competitive with that of state-of-the-art group signatures [15] or traceable signatures [22] relying on non-interactive assumptions in the standard model.

2 Background

In the paper, when S is a set, $x \xleftarrow{R} S$ denotes the action of choosing x at random in S. By $a \in \mathsf{poly}(\lambda)$, we mean that a is a polynomial in λ while $b \in \mathsf{negl}(\lambda)$ says that b is a negligible function of λ. When a and b are two binary strings, $a\|b$ stands for their concatenation. For equal-dimension vectors $\vec{A}$ and $\vec{B}$ containing group elements, $\vec{A} \odot \vec{B}$ stands for their component-wise product.

2.1 Complexity Assumptions

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

Definition 1 ([6]). *The* Decision Linear Problem *(DLIN) in $\mathbb{G}$, is to distinguish the distribution of $D_1 = \{(g, g^a, g^b, g^{ac}, g^{bd}, g^{c+d}) \mid a, b, c, d \xleftarrow{R} \mathbb{Z}_p\}$ from the distribution $D_2 = \{(g, g^a, g^b, g^{ac}, g^{bd}, g^z) \mid a, b, c, d, z \xleftarrow{R} \mathbb{Z}_p\}$.*

We also rely on a problem whose generic hardness of which was proved in [1].

Definition 2 ([1]). *In a group $\mathbb{G}$ of prime order p, the q-*Simultaneous Flexible Pairing Problem *(q-SFP) is, given $\left(g_z, h_z, g_r, h_r, a, \tilde{a}, b, \tilde{b}\right) \in \mathbb{G}^8$ as well as q tuples $(z_j, r_j, s_j, t_j, u_j, v_j, w_j) \in \mathbb{G}^7$ such that*

$$e(a,\tilde{a}) = e(g_z, z_j)\cdot e(g_r, r_j)\cdot e(s_j, t_j) \quad \text{and} \quad e(b,\tilde{b}) = e(h_z, z_j)\cdot e(h_r, u_j)\cdot e(v_j, w_j)\,,$$

to find a new tuple $(z^\star, r^\star, s^\star, t^\star, u^\star, v^\star, w^\star) \in \mathbb{G}^7$ satisfying the above equations and such that $z^\star \notin \{1_{\mathbb{G}}, z_1, \ldots, z_q\}$.

Definition 3 ([7]). *The* Decision 3-party Diffie-Hellman Problem *(D3DH) in $\mathbb{G}$, is to distinguish the distributions $(g, g^a, g^b, g^c, g^{abc})$ and (g, g^a, g^b, g^c, g^z), where $a, b, c, z \xleftarrow{R} \mathbb{Z}_p$.*

2.2 Groth-Sahai Proof Systems

In symmetric pairing configurations, the Groth-Sahai (GS) proof systems [16] use a common reference string (CRS) consisting of three vectors $\vec{g_1}, \vec{g_2}, \vec{g_3} \in \mathbb{G}^3$, where $\vec{g_1} = (g_1, 1, g)$, $\vec{g_2} = (1, g_2, g)$ for some $g_1, g_2 \in \mathbb{G}$. To commit to a group element $X \in \mathbb{G}$, the prover computes $\vec{C} = (1, 1, X) \odot \vec{g_1}^r \odot \vec{g_2}^s \odot \vec{g_3}^t$ with $r, s, t \xleftarrow{R} \mathbb{Z}_p$. When the proof system is configured to provide perfectly sound proofs, $\vec{g_3}$ is set as $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$ with $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p$. In this case, commitments $\vec{C} = (g_1^{r+\xi_1 t}, g_2^{s+\xi_2 t}, X \cdot g^{r+s+t(\xi_1+\xi_2)})$ can be interpreted as Boneh-Boyen-Shacham (BBS) ciphertexts as X can be recovered by running the BBS decryption algorithm using the private key $(\alpha_1, \alpha_2) = (\log_g(g_1), \log_g(g_2))$. When the CRS is set up to give perfectly witness indistinguishable (WI) proofs, $\vec{g_1}, \vec{g_2}$ and $\vec{g_3}$ are linearly independent vectors, so that $\vec{C}$ is a perfectly hiding commitment to $X \in \mathbb{G}$: a typical choice is $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2} \odot (1, 1, g)^{-1}$. Under the DLIN assumption, the two distributions of CRS are computationally indistinguishable.

To commit to an exponent $x \in \mathbb{Z}_p$, the prover computes $\vec{C} = \vec{\varphi}^x \odot \vec{g_1}^r \odot \vec{g_2}^s$, with $r, s \xleftarrow{R} \mathbb{Z}_p$, using a CRS containing $\vec{\varphi}, \vec{g_1}, \vec{g_2}$. In the perfect soundness setting $\vec{\varphi}, \vec{g_1}, \vec{g_2}$ are linearly independent (typically $\vec{\varphi} = \vec{g_3} \odot (1, 1, g)$ where $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$) whereas, in the perfect WI setting, choosing $\vec{\varphi} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$ yields perfectly hiding commitments since $\vec{C}$ is statistically independent of x.

Efficient NIWI proofs are available for pairing-product relations, which are equations of the form $\prod_{i=1}^n e(\mathcal{A}_i, \mathcal{X}_i) \cdot \prod_{i=1}^n \cdot \prod_{j=1}^n e(\mathcal{X}_i, \mathcal{X}_j)^{a_{ij}} = t_T$, for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$ and constants $t_T \in \mathbb{G}_T$, $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathbb{G}$, $a_{ij} \in \mathbb{Z}_p$, for $i, j \in \{1, \ldots, n\}$. Efficient proofs also exist for multi-exponentiation equations like $\prod_{i=1}^m \mathcal{A}_i^{y_i} \cdot \prod_{j=1}^n \mathcal{X}_j^{b_j} \cdot \prod_{i=1}^m \cdot \prod_{j=1}^n \mathcal{X}_j^{y_i \gamma_{ij}} = T$, for variables $\mathcal{X}_1, \ldots, \mathcal{X}_n \in \mathbb{G}$, $y_1, \ldots, y_m \in \mathbb{Z}_p$ and constants $T, \mathcal{A}_1, \ldots, \mathcal{A}_m \in \mathbb{G}$, $b_1, \ldots, b_n \in \mathbb{Z}_p$ and $\gamma_{ij} \in \mathbb{Z}_p$, for $i \in \{1, \ldots, m\}, j \in \{1, \ldots, n\}$.

Multi-exponentiation equations always admit non-interactive zero-knowledge (NIZK) proofs at no additional cost. On a perfectly witness indistinguishable CRS, a trapdoor (like the hidden exponents $(\xi_1, \xi_2) \in \mathbb{Z}_p^2$ when $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2} \odot (1, 1, g)^{-1}$) allows simulating proofs without knowing the witnesses and simulated proofs are perfectly indistinguishable from real proofs. As for pairing-product equations, zero-knowledge proofs are often possible – this is usually the case when the right-hand-side member t_T is a product of pairings involving known group elements – but the number of group elements per proof may not be constant anymore. Here, when using such NIZK simulators, we just introduce a constant number of extra group elements in the proofs.

2.3 Chameleon Hash Functions

A chameleon hash function [21] is a tuple $\mathsf{CMH} = (\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$ that contains an algorithm CMKg that, given a security parameter λ, outputs a key pair $(hk, tk) \leftarrow \mathcal{G}(\lambda)$. The hashing algorithm outputs $y = \mathsf{CMhash}(hk, m, r)$ given the public key hk, a message m and random coins $r \in \mathcal{R}_{hash}$. On input of messages m, m', random coins $r \in \mathcal{R}_{hash}$ and the trapdoor key tk, the

switching algorithm $r' \leftarrow \mathsf{CMswitch}(tk, m, r, m')$ computes $r' \in \mathcal{R}_{hash}$ such that $\mathsf{CMhash}(hk, m, r) = \mathsf{CMhash}(hk, m', r')$. The collision-resistance property mandates that it be infeasible to come up with pairs $(m', r') \neq (m, r)$ such that $\mathsf{CMhash}(hk, m, r) = \mathsf{CMhash}(hk, m', r')$ without knowing the trapdoor key tk. Uniformity guarantees that the distribution of hash values is independent of the message m: for all hk, and all m, m', the distributions $\{r \leftarrow \mathcal{R}_{hash} : \mathsf{CMHash}(hk, m, r)\}$ and $\{r \leftarrow \mathcal{R}_{hash} : \mathsf{CMHash}(hk, m', r)\}$ are identical.

3 Traceable Group Encryption

3.1 Syntax

Traceable group encryption (TGE) schemes involve a sender, a verifier, a group manager (GM) that manages the group of receivers and an opening authority (OA) that is able to uncover the identity of ciphertext receivers.

A group encryption system is formally specified by the description of a relation $\mathcal{R}$ and a collection $\mathsf{TGE} = \big(\mathsf{SETUP}, \mathsf{JOIN}, \langle \mathcal{G}_r, \mathcal{R}, \mathsf{sample}_{\mathcal{R}} \rangle, \mathsf{ENC}, \mathsf{DEC}, \langle \mathcal{P}, \mathcal{V} \rangle, \mathsf{OPEN}, \mathsf{REVEAL}, \mathsf{TRACE}, \mathsf{CLAIM/DISCLAIM}, \mathsf{CLAIM\text{-}VERIFY}, \mathsf{DISCLAIM\text{-}VERIFY}\big)$ of algorithms or protocols. Among these, SETUP is a set of initialization procedures that all take (explicitly or implicitly) a security parameter λ as input. They can be split into one that generates a set of public parameters param (a common reference string), one for the GM and another one for the OA. We call them $\mathsf{SETUP}_{\mathsf{init}}(\lambda)$, $\mathsf{SETUP}_{\mathsf{GM}}(\mathsf{param})$ and $\mathsf{SETUP}_{\mathsf{OA}}(\mathsf{param})$, respectively. The latter two procedures are used to produce key pairs $(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}})$, $(\mathsf{pk}_{\mathsf{OA}}, \mathsf{sk}_{\mathsf{OA}})$ for the GM and the OA. In the following, param is incorporated in the inputs of all algorithms although we sometimes omit to explicitly write it.

$\mathsf{JOIN} = (\mathsf{J}_{\mathsf{user}}, \mathsf{J}_{\mathsf{GM}})$ is an interactive protocol between the GM and the prospective user. As in [9], we will aim for two-message protocols: the first message is the user's public key pk sent by $\mathsf{J}_{\mathsf{user}}$ to J_{GM} and the latter's response is a certificate $\mathsf{cert}_{\mathsf{pk}}$ for pk vouching for the user's group membership. The user is not required to prove knowledge of his private key sk. Valid public keys are assumed to be publicly recognizable, so that proofs of validity are not necessary either. After the execution of JOIN, the GM stores the public key pk and its certificate $\mathsf{cert}_{\mathsf{pk}}$ in a public directory $\mathsf{database}$.

Algorithm sample allows sampling pairs $(x, w) \in \mathcal{R}$ (comprised of a public value x and a witness w) using public / secret parameters $(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}})$ produced by $\mathcal{G}_r$ for $\mathcal{R}$. Depending on the relation, $\mathsf{sk}_{\mathcal{R}}$ may be the empty string, as in the scheme we describe. The testing procedure $\mathcal{R}(x, w)$ returns 1 iff $(x, w) \in \mathcal{R}$. To encrypt a witness w such that $(x, w) \in \mathcal{R}$ for some public x, the sender picks the pair $(\mathsf{pk}, \mathsf{cert}_{\mathsf{pk}})$ from $\mathsf{database}$ and runs the encryption algorithm. The latter takes as input w, a label L, the receiver's pair $(\mathsf{pk}, \mathsf{cert}_{\mathsf{pk}})$ as well as public keys $\mathsf{pk}_{\mathsf{GM}}$ and $\mathsf{pk}_{\mathsf{OA}}$. Its output is a ciphertext $\psi \leftarrow \mathsf{ENC}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}, \mathsf{cert}_{\mathsf{pk}}, w, L)$. On input of the same elements, the certificate $\mathsf{cert}_{\mathsf{pk}}$, the ciphertext ψ and the random encryption coins $coins_{\psi}$, the non-interactive algorithm $\mathcal{P}$ generates a proof π_{ψ} that there exists a certified receiver whose public key was registered

in $\mathsf{database}$ and that is able to decrypt ψ and obtain a witness w such that $(x, w) \in \mathcal{R}$. The verification algorithm $\mathcal{V}$ takes as input ψ, $\mathsf{pk_{GM}}$, $\mathsf{pk_{OA}}$, π_ψ and the description of $\mathcal{R}$ and outputs 0 or 1. Given ψ, L and the receiver's private key sk, the output of DEC is either a witness w such that $(x, w) \in \mathcal{R}$ or $\perp$.

The next three algorithms provide explicit and implicit tracing capabilities. First, OPEN takes as input a ciphertext/label pair (ψ, L) and the OA's secret key $\mathsf{sk_{OA}}$ and returns a receiver's identity i. Algorithm REVEAL takes as input the joining transcript $\mathsf{transcript}_i$ of user i and allows the OA to extract a tracing trapdoor trace_i using its private key $\mathsf{sk_{OA}}$. This tracing trapdoor can be subsequently used to determine whether or not a given ciphertext-label pair (ψ, L) is a valid encryption under the public key pk_i of user i: namely, algorithm TRACE takes in public keys $\mathsf{pk_{GM}}$ and $\mathsf{pk_{OA}}$ as well as a pair (ψ, L) and the tracing trapdoor trace_i associated with user i. It returns 1 if and only if (ψ, L) is believed to be a valid encryption intended for user i.

Finally, algorithms ($\mathsf{CLAIM/DISCLAIM}$, $\mathsf{CLAIM\text{-}VERIFY}$, $\mathsf{DISCLAIM\text{-}VERIFY}$) implement a functionality that allows user to convincingly claim or disclaim being the legitimate recipient of a given anonymous ciphertext. Concretely, $\mathsf{CLAIM/DISCLAIM}$ takes as input all public keys ($\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{pk}$), a ciphertext-label pair (ψ, L) and a private key sk. It reveals a publicly verifiable piece of evidence τ that (ψ, L) is or is not a valid encryption under the public key pk. Algorithms $\mathsf{CLAIM\text{-}VERIFY}$ and $\mathsf{DISCLAIM\text{-}VERIFY}$ are then used to verify the assertion established by τ. They take as input all public keys, a pair (ψ, L) and a claim/disclaimer τ and output 1 or 0.

3.2 Security Definitions

Beyond the standard correctness requirement, our security model involves four properties called message privacy, anonymity, soundness and claiming soundness. In the definitions hereunder, we use the notation $\langle \mathsf{output}_A | \mathsf{output}_B \rangle \leftarrow \langle A(\mathsf{input}_A), B(\mathsf{input}_B) \rangle (\mathsf{common\text{-}input})$ to denote the execution of a protocol between A and B obtaining their own outputs from their respective inputs.

Correctness. The following experiment should return 1 w.h.p.

Experiment $\mathbf{Expt}^{\text{correctness}}(\lambda)$
$\mathsf{param} \leftarrow \mathsf{SETUP}_{\mathsf{init}}(\lambda)$; $(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}}) \leftarrow \mathcal{G}_r(\lambda)$; $(x, w) \leftarrow \mathsf{sample}_{\mathcal{R}}(\mathsf{pk}_{\mathcal{R}}, \mathsf{sk}_{\mathcal{R}})$;
$(\mathsf{pk_{GM}}, \mathsf{sk_{GM}}) \leftarrow \mathsf{SETUP_{GM}}(\mathsf{param})$; $(\mathsf{pk_{OA}}, \mathsf{sk_{OA}}) \leftarrow \mathsf{SETUP_{OA}}(\mathsf{param})$;
$\langle \mathsf{pk}_i, \mathsf{sk}_i, \mathsf{cert}_{\mathsf{pk}_i} | \mathsf{pk}_i, \mathsf{cert}_{\mathsf{pk}_i} \rangle \leftarrow \langle \mathsf{J_{user}}, \mathsf{J_{GM}}(\mathsf{sk_{GM}}) \rangle (\mathsf{pk_{GM}})$;
$\psi \leftarrow \mathsf{ENC}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{pk}_i, \mathsf{cert}_{\mathsf{pk}_i}, w, L)$;
$\pi_\psi \leftarrow \mathcal{P}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \mathsf{pk}_i, \mathsf{cert}_{\mathsf{pk}_i}, x, w, L, \psi, coins_\psi)$;
If $\big((w \neq \mathsf{DEC}(\mathsf{sk}_i, \psi, L)) \vee (i \neq \mathsf{OPEN}(\mathsf{sk_{OA}}, \psi, L))$
$\vee (\mathcal{V}(\psi, L, \pi_\psi, \mathsf{pk_{GM}}, \mathsf{pk_{OA}}) = 0)\big)$ return 0 else return 1.

Message privacy. This property is defined by an experiment where the adversary has access to oracles that may be stateless or maintain a state across queries:

– $\mathsf{DEC}(\mathsf{sk})$: is an oracle for the user decryption function. When it is restricted not to decrypt a ciphertext-label pair (ψ, L), we denote it by $\mathsf{DEC}^{\neg\langle\psi,L\rangle}$.
– $\mathsf{CH}^b_{\mathsf{ror}}(\lambda, \mathsf{pk}, w, L)$: is a real-or-random challenge oracle that is only queried once. It returns $(\psi, coins_\psi)$ such that $\psi \leftarrow \mathsf{ENC}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}, \mathsf{cert}_{\mathsf{pk}}, w, L)$ if $b = 1$ whereas, if $b = 0$, $\psi \leftarrow \mathsf{ENC}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}, \mathsf{cert}_{\mathsf{pk}}, w', L)$ encrypts a random plaintext uniformly chosen in the space of plaintexts of length $O(\lambda)$. In either case, $coins_\psi$ are the random coins used to generate ψ.
– $\mathsf{PROVE}^b_{\mathcal{P},\mathcal{P}'}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}, \mathsf{cert}_{\mathsf{pk}}, \mathsf{pk}_{\mathcal{R}}, x, w, \psi, L, coins_\psi)$: is a stateful oracle that the adversary can query on multiple occasions. If $b = 1$, it runs the real prover $\mathcal{P}$ on the inputs to produce an actual proof π_ψ. If $b = 0$, the oracle runs a simulator $\mathcal{P}'$ that uses the same inputs as $\mathcal{P}$ except w and $coins_\psi$ and generates a simulated proof.
– $\mathsf{CLAIM/DISCLAIM}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \psi, L, \mathsf{sk})$: is a stateful oracle that generates claims or disclaimer proofs for arbitrary ciphertexts. Specifically, the oracle first uses the private key sk to determine whether (ψ, L) is a valid ciphertext-label pair w.r.t. the public key pk. If so, the oracle uses sk to compute and return a non-interactive claim τ for ψ. Otherwise, the oracle generates a disclaimer proof τ showing that (ψ, L) is not a valid encryption under pk. In either case, (ψ, L) is stored in a list claims, which is initially empty.

These oracles are used in an experiment where the adversary controls the GM, the OA and all members but the honest receiver. The adversary $\mathcal{A}$ is the dishonest GM that certifies the honest receiver in an execution of JOIN. It has oracle access to the decryption function DEC of that receiver. At the challenge phase, it probes the challenge oracle for a label and a pair $(x, w) \in \mathcal{R}$ of her choice. After the challenge phase, $\mathcal{A}$ can also invoke the PROVE oracle on multiple occasions and eventually aims to guess the bit b chosen by the challenger.

As pointed out in [19], designing an efficient simulator $\mathcal{P}'$ (for executing $\mathsf{PROVE}^b_{\mathcal{P},\mathcal{P}'}(.)$ when $b = 0$) is part of the security proof and might require a simulated common reference string.

Definition 4. *A TGE scheme satisfies message security if, for any PPT adversary $\mathcal{A}$, the experiment below returns 1 with probability at most $1/2 + \mathsf{negl}(\lambda)$.*

Experiment $\mathbf{Expt}^{\mathrm{sec}}_{\mathcal{A}}(\lambda)$
$\mathsf{param} \leftarrow \mathsf{SETUP}_{\mathsf{init}}(\lambda)$; $(\mathsf{aux}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}) \leftarrow \mathcal{A}(\mathsf{param})$;
$\langle \mathsf{pk}, \mathsf{sk}, \mathsf{cert}_{\mathsf{pk}} | \mathsf{aux} \rangle \leftarrow \langle \mathsf{J}_{\mathsf{user}}, \mathcal{A}(\mathsf{aux}) \rangle (\mathsf{pk}_{\mathsf{GM}})$;
$(\mathsf{aux}, x, w, L, \mathsf{pk}_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathsf{DEC}(\mathsf{sk},.),\ \mathsf{CLAIM/DISCLAIM}(\mathsf{pk}_{\mathsf{GM}},\mathsf{pk}_{\mathsf{OA}},\ldots,\mathsf{sk})}(\mathsf{aux})$;
If $(x, w) \notin \mathcal{R}$ *return* 0; $b \xleftarrow{R} \{0, 1\}$; $(\psi, coins_\psi) \leftarrow \mathsf{CH}^b_{\mathsf{ror}}(\lambda, \mathsf{pk}, w, L)$;
$b' \leftarrow \mathcal{A}^{\mathsf{PROVE}^b_{\mathcal{P},\mathcal{P}'}(\mathsf{pk}_{\mathsf{GM}},\mathsf{pk}_{\mathsf{OA}},\mathsf{pk},\mathsf{cert}_{\mathsf{pk}},\mathsf{pk}_{\mathcal{R}},x,w,\psi,L,coins_\psi),\mathsf{DEC}^{\neg\langle\psi,L\rangle}(\mathsf{sk},.),\ \mathsf{CLAIM/DISCLAIM}(\mathsf{pk}_{\mathsf{GM}},\mathsf{pk}_{\mathsf{OA}},.,.,\mathsf{sk})}(\mathsf{aux}, \psi)$;
If $b = b'$ *return* 1 *else return* 0.

Anonymity. In anonymity attacks, the adversary controls the entire system except the opening authority. One way to jeopardize the anonymity property is to mount a chosen-ciphertext attack on the encryption scheme used by the

OA. A difference with the usual group encryption scenario is that we must pay attention to the information revealed by the traceability components of ciphertexts. Throughout the game, the adversary can act as a dishonest group manager and register honest users in the system. In the challenge phase, the adversary A chooses a pair $(x, w) \in \mathcal{R}$ and the public keys $\mathsf{pk}_0, \mathsf{pk}_1$ of two honest users. In return, it receives an encryption of w under the public key pk_b for some $b \in \{0, 1\}$ chosen by the challenger. It has access to the following oracles:

- $\mathsf{USER}(\mathsf{pk}_{\mathsf{GM}})$: is a stateful oracle simulating executions of $\mathsf{J}_{\mathsf{user}}$ on behalf of new honest users who are requested to join the group. It uses an initially empty list keys. At its i-th invocation, the output $(i, \mathsf{pk}_i, \mathsf{sk}_i, \mathsf{cert}_{\mathsf{pk}_i})$ of $\mathsf{J}_{\mathsf{user}}$ is stored in keys if the adversary, which emulates the GM, provides a valid certificate $\mathsf{cert}_{\mathsf{pk}_i}$. If the JOIN protocol does not successfully terminate, the oracle stores $(i, \perp)$ in keys.
- $\mathsf{CORR}(.)$: is a stateful oracle that allows the adversary to corrupt honest group members. When invoked on input of an index i, the oracle first checks if the list keys contains an entry of the form $(i, \mathsf{pk}_i, \mathsf{sk}_i, \mathsf{cert}_{\mathsf{pk}_i})$. If so, it returns sk_i and adds i to the set Corr, which is initially empty.
- $\mathsf{DEC}(.,.)$: is a stateless decryption oracle that provides a decryption capability for each secret key. It takes as input an index i and a ciphertext-label pair (ψ, L). It first checks if the list keys contains an entry of the form $(i, \mathsf{pk}_i, \mathsf{sk}_i, \mathsf{cert}_{\mathsf{pk}_i})$. If no such entry exists, it returns $\perp$. Otherwise, it uses sk_i to run DEC on the input (ψ, L) and returns the result. When this oracle is restricted not to decrypt a ciphertext-label pair (ψ, L) for some user index $i \in \{i_0, i_1\}$, we denote it by $\mathsf{DEC}^{\neg\{i_0,i_1\}\times\langle\psi,L\rangle}$.
- $\mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}}, .)$: is a stateless oracle that runs the opening algorithm on behalf of the OA. On input of a TGE ciphertext, it returns the receiver's identity i.
- $\mathsf{REVEAL}(\mathsf{sk}_{\mathsf{OA}}, .)$: is an oracle that takes as input a user index i and simulates the REVEAL algorithm on behalf of the OA. If no user was assigned the index i in keys, it returns $\perp$. Otherwise, it recovers the transcript $\mathsf{transcript}_i$ of user i in $\mathsf{database}$ and uses $\mathsf{sk}_{\mathsf{OA}}$ to extract and return the i-th group member's tracing trapdoor trace_i. It also adds i to the set Revs.
- $\mathsf{CH}^b_{\mathsf{anon}}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}_0, \mathsf{pk}_1, w, L)$: is a challenge oracle that can only be queried once. It returns a pair $(\psi, coins_\psi)$ consisting of a ciphertext $\psi \leftarrow \mathsf{ENC}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}_b, \mathsf{cert}_{\mathsf{pk}_b}, w, L)$ and the coin tosses used to generate ψ.
- $\mathcal{P}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}_b, \mathsf{cert}_{\mathsf{pk}_b}, \mathsf{pk}_{\mathcal{R}}, x, w, \psi, L, coins_\psi)$: is a stateful oracle which can be queried several times after the challenge phase. It runs the real prover $\mathcal{P}$ on the inputs to produce an actual proof π_ψ using the random coins $coins_\psi$ involved in the generation of the challenge. It returns the resulting proof π_ψ.
- $\mathsf{CLAIM/DISCLAIM}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \psi, L, i)$: is a stateful oracle. It takes as input an index i and a ciphertext/label pair. It first checks whether keys contains a tuple $\mathsf{transcript}_i = (i, \mathsf{pk}_i, \mathsf{sk}_i, \mathsf{cert}_{\mathsf{pk}_i})$. If not, it returns $\perp$. Otherwise, it uses the private key sk_i to determine whether (ψ, L) is a valid ciphertext-label pair w.r.t. the public key pk_i. If yes, the oracle uses sk_i to generate a non-interactive claim τ for (ψ, L). Otherwise, the oracle generate a disclaimer τ guaranteeing that (ψ, L) is not a valid encryption under pk_i. In either case, (i, ψ, L) is stored in a list claims, which is initially empty.

Definition 5. *A TGE scheme satisfies anonymity if, for any PPT adversary $\mathcal{A}$, the experiment below returns 1 with a probability not exceeding $1/2 + \mathsf{negl}(\lambda)$.*

Experiment $\mathbf{Expt}^{\mathrm{anon}}_{\mathcal{A}}(\lambda)$
$\mathsf{param} \leftarrow \mathsf{SETUP}_{\mathsf{init}}(\lambda)$; $(\mathsf{pk}_{\mathsf{OA}}, \mathsf{sk}_{\mathsf{OA}}) \leftarrow \mathsf{SETUP}_{\mathsf{OA}}(\mathsf{param})$;
$(\mathsf{aux}, \mathsf{pk}_{\mathsf{GM}}) \leftarrow \mathcal{A}(\mathsf{param}, \mathsf{pk}_{\mathsf{OA}})$;
$(i_0, i_1, \mathsf{aux}, x, w, L, \mathsf{pk}_{\mathcal{R}}) \leftarrow \mathcal{A}^{\mathsf{USER}(\mathsf{pk}_{\mathsf{GM}}),\ \mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}},.),\ \mathsf{REVEAL}(\mathsf{sk}_{\mathsf{OA}},.),\ \mathsf{DEC}(.,.),\ \mathsf{CLAIM/DISCLAIM}(\mathsf{pk}_{\mathsf{GM}},\mathsf{pk}_{\mathsf{OA}},.,.,.),\ \mathsf{CORR}(.)}(\mathsf{aux})$;
If $(i_0, \mathsf{pk}_0, \mathsf{sk}_0, \mathsf{cert}_{\mathsf{pk}_0}) \notin \mathsf{keys} \ \vee \ (i_1, \mathsf{pk}_1, \mathsf{sk}_1, \mathsf{cert}_{\mathsf{pk}_1}) \notin \mathsf{keys}$ *return* 0;
If $(x, w) \notin \mathcal{R}$ *return* 0; $b \xleftarrow{R} \{0,1\}$;
$(\psi, coins_{\psi}) \leftarrow \mathsf{CH}^{b}_{\mathsf{anon}}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}_0, \mathsf{pk}_1, w, L)$;
$b' \leftarrow \mathcal{A}^{\mathsf{USER}(\mathsf{pk}_{\mathsf{GM}}),\ \mathcal{P}(\mathsf{pk}_{\mathsf{GM}},\mathsf{pk}_{\mathsf{OA}},\mathsf{pk}_b,\mathsf{cert}_{\mathsf{pk}_b},x,w,\psi,L,coins_{\psi}),\ \mathsf{OPEN}^{\neg\langle\psi,L\rangle}(\mathsf{sk}_{\mathsf{OA}},.),\ \mathsf{CORR}(.)\ \mathsf{REVEAL}^{\neg\{i_0,i_1\}}(\mathsf{sk}_{\mathsf{OA}},.),\ \mathsf{DEC}^{\neg\{i_0,i_1\}\times\langle\psi,L\rangle}(.,.),\ \mathsf{CLAIM/DISCLAIM}(\mathsf{pk}_{\mathsf{GM}},\mathsf{pk}_{\mathsf{OA}},.,.,.)}(\mathsf{aux}, \psi)$;
If $\big((i_0, \psi, L) \in \mathsf{claims}\big) \ \vee \ \big((i_1, \psi, L) \in \mathsf{claims}\big)$ *return* 0;
If $(i_0 \in \mathsf{Revs} \cup \mathsf{Corr}) \ \vee \ (i_1 \in \mathsf{Revs} \cup \mathsf{Corr})$ *return* 0;
If $b = b'$ *return* 1 *else return* 0.

As shown in [19], TGE schemes satisfying the above notion necessarily subsume a key-private (a.k.a. receiver anonymous) [3] cryptosystem.

Soundness. In a soundness attack, the adversary creates the group of receivers by interacting with the honest GM. Its goal is to create a ciphertext ψ and a convincing proof that ψ is valid w.r.t. a relation $\mathcal{R}$ of its choice but either (1) the opening fails to identify a certified group member as the legitimate recipient of ψ; (2) the implicit tracing mechanism TRACE does not point to the group member pinned down by OPEN; (3) the ciphertext C is not in the language $\mathcal{L}^{x,L,\mathsf{pk}_{\mathcal{R}},\mathsf{pk}_{\mathsf{GM}},\mathsf{pk}_{\mathsf{OA}},\mathsf{pk}_i} = \{\mathsf{ENC}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}_i, \mathsf{cert}_{\mathsf{pk}_i}, w, L) \mid (x, w) \in \mathcal{R};\ (\mathsf{pk}_i, \mathsf{cert}_{\mathsf{pk}_i}) \in \mathsf{valid}\}$, where valid is the set of properly certified keys. This notion is formalized by a game where the adversary is given access to a user registration oracle $\mathsf{REG}(\mathsf{sk}_{\mathsf{GM}}, .)$ that emulates J_{GM}. This oracle maintains a repository database where registered public keys and their certificates are stored.

Definition 6. *A TGE scheme is sound if, for any PPT adversary $\mathcal{A}$, the experiment below returns 1 with negligible probability.*

Experiment $\mathbf{Expt}^{\mathrm{soundness}}_{\mathcal{A}}(\lambda)$
$\mathsf{param} \leftarrow \mathsf{SETUP}_{\mathsf{init}}(\lambda)$; $(\mathsf{pk}_{\mathsf{OA}}, \mathsf{sk}_{\mathsf{OA}}) \leftarrow \mathsf{SETUP}_{\mathsf{OA}}(\mathsf{param})$;
$(\mathsf{pk}_{\mathsf{GM}}, \mathsf{sk}_{\mathsf{GM}}) \leftarrow \mathsf{SETUP}_{\mathsf{GM}}(\mathsf{param})$;
$(\mathsf{pk}_{\mathcal{R}}, x, \psi, \pi_{\psi}, L, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{REG}(\mathsf{sk}_{\mathsf{GM}},.)}(\mathsf{param}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{sk}_{\mathsf{OA}})$;
If $\mathcal{V}(\psi, L, \pi_{\psi}, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}) = 0$ *return* 0;
$i \leftarrow \mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}}, \psi, L)$;
If $\big((i = \perp) \vee (\psi \notin \mathcal{L}^{x,L,\mathsf{pk}_{\mathcal{R}},\mathsf{pk}_{\mathsf{GM}},\mathsf{pk}_{\mathsf{OA}},\mathsf{pk}_i})\big)$ *then return* 1;
$\mathsf{trace}_i \leftarrow \mathsf{REVEAL}(\mathsf{transcript}_i, \mathsf{sk}_{\mathsf{OA}})$;
If $\big(i \neq \mathsf{TRACE}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \psi, \mathsf{trace}_i)\big)$ *then return* 1;
Return 0.

The above properties are similar to those for group encryption. We need to introduce the new notion of *claiming soundness* (which is not part of the group encryption model [19]) that formalizes the soundness of the claiming process.

CLAIMING SOUNDNESS. The last security notion considers an adversary attacking the soundness of the claiming algorithm by either claiming other users' ciphertexts as its own or disclaiming ciphertexts that are actually encrypted under its public key. Moreover, the verifier of a claim/disclaimer should be convinced of the group member's intentionality to claim or repudiate ciphertexts. We require that only users be able to claim/disclaim ciphertexts encrypted under their key or not: even the sender (who knows the encryption coins) should not do this.

In the model, the adversary controls the GM and the OA. It has access to oracles $\mathsf{USER}(\mathsf{pk_{GM}})$, $\mathsf{CORR}(.)$, $\mathsf{DEC}(.,.)$ and $\mathsf{CLAIM/DISCLAIM}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \psi, L, i)$, which are identical to those of the anonymity property.

The adversary's goal is to create a public repository $\mathsf{database}$ satisfying the integrity check, a ciphertext ψ and a statement $\mathsf{statement}$ consisting of a claim/disclaimer τ and a public key pk but either: (1) the implicit tracing mechanism TRACE does not point to the group member i pinned down by OPEN; (2) $\mathsf{statement} = (\tau, \mathsf{pk})$ is a valid claim although $\mathsf{pk} \neq \mathsf{pk}_i$, where pk_i is associated with user i in $\mathsf{database}$; (3) $\mathsf{statement} = (\tau, \mathsf{pk})$ is a valid disclaimer whereas $\mathsf{pk} = \mathsf{pk}_i$ coincides with the public key associated with user i in $\mathsf{database}$; (4) $\mathsf{statement} = (\tau, \mathsf{pk}_j)$ is a valid claim/disclaimer for the public key pk_j of some uncorrupted user $j \in \mathsf{database} \backslash \mathsf{Corr}$ in the database and the pair (τ, pk_j) was not produced by the $\mathsf{CLAIM/DISCLAIM}$ oracle.

Definition 7. *A TGE scheme provides claiming-soundness if, for any PPT adversary $\mathcal{A}$, the experiment below returns 1 with negligible probability.*

Experiment $\mathbf{Expt}_{\mathcal{A}}^{\text{claiming-soundness}}(\lambda)$
$\mathsf{param} \leftarrow \mathsf{SETUP}_{\mathsf{init}}(\lambda)$; $(\mathsf{pk_{GM}}, \mathsf{aux}_0) \leftarrow \mathcal{A}(\mathsf{param})$;
$(\mathsf{pk_{OA}}, \mathsf{sk_{OA}}) \leftarrow \mathsf{SETUP_{OA}}(\mathsf{param})$;
$(\mathsf{pk}^\star_{\mathcal{R}}, x^\star, \psi^\star, L^\star, \pi^\star_\psi, \mathsf{statement}^\star, \mathsf{database}^\star, \mathsf{aux}) \leftarrow \mathcal{A}^{\mathsf{USER}(\mathsf{pk_{GM}}),\ \mathsf{CORR}(.),\ \mathsf{DEC}(.,.),\ \mathsf{CLAIM/DISCLAIM}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, ., ., ., .)}(\mathsf{param}, \mathsf{pk_{OA}}, \mathsf{sk_{OA}}, \mathsf{aux}_0)$;
If $\mathsf{DATABASE\text{-}CHECK}(\mathsf{param}, \mathsf{database}) = 0$ *return* 0;
If $\mathcal{V}(\psi^\star, L^\star, \pi^\star_\psi, \mathsf{pk_{GM}}, \mathsf{pk_{OA}}) = 0$ *return* 0;
$i \leftarrow \mathsf{OPEN}(\mathsf{sk_{OA}}, \psi^\star, L^\star)$; $\mathsf{trace}_i \leftarrow \mathsf{REVEAL}(\mathsf{transcript}_i, \mathsf{sk_{OA}})$;
If $\big(i \neq \mathsf{TRACE}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \psi^\star, \mathsf{trace}_i)\big)$ *then return* 1;
If $\big(\mathsf{statement}^\star = (\tau^\star, \mathsf{pk}^\star)$ *s.t.* $(\mathsf{pk}^\star \neq \mathsf{pk}_i)$
$\wedge\ \mathsf{CLAIM\text{-}VERIFY}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \psi^\star, L^\star, \mathsf{pk}^\star, \tau^\star) = 1\big)$ *then return* 1;
If $\big(\mathsf{statement}^\star = (\tau^\star, \mathsf{pk}^\star)$ *s.t.* $(\mathsf{pk}^\star = \mathsf{pk}_i)$
$\wedge\ \mathsf{DISCLAIM\text{-}VERIFY}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \psi^\star, L^\star, \mathsf{pk}^\star, \tau^\star) = 1\big)$ *then return* 1;
If $\Big(\mathsf{statement}^\star = (\tau^\star, \mathsf{pk}_j)$ *s.t.* $(j, \mathsf{pk}_j, \mathsf{cert}_j, .) \in \mathsf{database}\ \wedge\ (j \notin \mathsf{Corr})$
$\wedge\ (\psi^\star, L^\star, \mathsf{pk}_j) \notin Q_c\ \wedge\ \big(\mathsf{CLAIM\text{-}VERIFY}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \psi^\star, L^\star, \mathsf{pk}_j, \tau^\star) = 1$
$\vee\ \mathsf{DISCLAIM\text{-}VERIFY}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \psi^\star, L^\star, \mathsf{pk}_j, \tau^\star) = 1\big)\Big)$ *then return* 1;
Return 0.

In the above notations, Q_c is the set of CLAIM/DISCLAIM queries made by $\mathcal{A}$.

We note that there is no need for a REVEAL oracle in the definition. Indeed, since $\mathcal{A}$ knows $\mathsf{sk}_{\mathsf{OA}}$, it can obtain tracing trapdoors by itself, by decrypting the verifiable encryptions sent by honest users when the USER oracle is invoked.

4 A Non-Interactive Traceable Group Encryption Scheme

We use the Libert-Yung (LY) scheme [23], which is a publicly verifiable variant of Cramer-Shoup [11]. We take advantage of the observation that, if certain public key components are shared by all users as common public parameters, the scheme can simultaneously provide receiver anonymity *and* publicly verifiable ciphertexts. In other words, anyone can publicly verify that a ciphertext is valid without knowing who the receiver is. When proofs are generated for the ciphertext, this saves the prover from having to provide evidence that the ciphertext is valid and thus yields shorter proofs.

The message is encrypted under the receiver's public key using the LY scheme. At the same time, the two last components of the receiver's public key is encrypted under the public key of the opening authority using Kiltz's encryption scheme [20]. We use this scheme because it is the most efficient DLIN-based CCA2-secure cryptosystem where the validity of ciphertexts is publicly verifiable and we do not need it to hide the public key under which it is generated.

When new users join the group, the GM provides them with a membership certificate made of a structure-preserving signature [14,1,2] on their public key which comprises group elements (X_1, X_2). We chose to work with the scheme of Abe, Haralambiev and Ohkubo (AHO) [1,2] because it allows working exclusively with linear pairing-product equations and thus obtain a better efficiency.

The implicit tracing mechanism must allow the OA to disclose user-specific tracing trapdoors. To this end, we include in each membership certificate a pair $(\Gamma_1, \Gamma_2) = (g^{\gamma_1}, g^{\gamma_2}) \in \mathbb{G}^2$, where $(\gamma_1, \gamma_2) \in \mathbb{Z}_p^2$ are part of the user's private key. When users join the group, they are thus requested to produce a pair $(\Gamma_1, \Gamma_2) = (g^{\gamma_1}, g^{\gamma_2})$ for which $g^{\gamma_1\gamma_2}$ will serve as a tracing trapdoor for them. Since $g^{\gamma_1\gamma_2}$ cannot be publicly revealed, we appeal to a verifiable encryption mechanism as was suggested in [5] in a related context: namely, the prospective user provides the GM with an encryption Φ_{venc} of $g^{\gamma_1\gamma_2}$ under the OA's public key and generates a non-interactive proof that the encrypted value is indeed an element $g^{\gamma_1\gamma_2}$ such that $(g, g^{\gamma_1}, g^{\gamma_2}, g^{\gamma_1\gamma_2})$ is a Diffie-Hellman tuple. The REVEAL algorithm thus uses the OA's private key to decrypt Φ_{venc} so as to expose $g^{\gamma_1\gamma_2}$. Armed with the information $\mathsf{trace}_i = g^{\gamma_1\gamma_2}$, a tracing agent can test whether a ciphertext ψ is prepared for user i as follows. We require each ciphertext ψ to contain elements of the form $(T_1, T_2, T_3) = (g^{\delta}, \Gamma_1^{\delta/\varrho}, \Gamma_2^{\varrho})$, where $\delta, \varrho \in_R \mathbb{Z}_p$ are chosen by the sender. Since $(\Gamma_1, \Gamma_2) = (g^{\gamma_1}, g^{\gamma_2})$, the TRACE algorithm concludes that user i is indeed the receiver if $e(T_1, g^{\gamma_1\gamma_2}) = e(T_2, T_3)$. At the same time, we can show that recognizing ciphertexts encrypted for user i without trace_i is as hard as solving the D3DH problem.

For technical reasons, we need to introduce an extra traceability component

$T_4 = (\Lambda_0^{\mathsf{VK}} \cdot \Lambda_1)^{\delta}$, where $\Lambda_0, \Lambda_1 \in \mathbb{G}$ are part of common public parameters and VK is the verification key of a one-time signature. The reason is that, in order to prove anonymity in our model, we need to bind (T_1, T_2, T_3) to the one-time verification key VK in a non-malleable way. Otherwise, an anonymity adversary could break the anonymity by having access to a $\mathsf{CLAIM}/\mathsf{DISCLAIM}$ oracle.

In order to prove or disprove that he is the intended recipient of a given pair (ψ, L), a user i can use the traceability components $(T_1, T_2, T_3) = (g^{\delta}, \Gamma_1^{\delta/\varrho}, \Gamma_2^{\varrho})$ of ψ and his private key $\gamma_1 = \log_g(\Gamma_1)$ to compute $\Gamma_1^{\delta} = T_1^{\gamma_1}$ (although he does not know δ), which allows anyone to realize that $(g, T_1, \Gamma_1, \Gamma_1^{\delta})$ forms a Diffie-Hellman tuple and that $e(\Gamma_1^{\delta}, \Gamma_2) = e(T_2, T_3)$. This is sufficient for proving that (ψ, L) was created for the public key $\mathsf{pk} = (X_1, X_2, \Gamma_1, \Gamma_2)$. In order to make sure that only the user will be able to compute non-interactive claims, we also require him to provide a non-interactive proof of knowledge of $\Gamma_{-1} = g^{1/\gamma_1}$ satisfying $e(\Gamma_1^{\delta}, \Gamma_{-1}) = e(T_1, g)$. Moreover, the claim is non-malleably bound to (ψ, L, pk) – where pk is the claimer's public key —by generating the non-interactive Groth-Sahai proof for a CRS $(\vec{g_1}, \vec{g_2}, \vec{h}_{\mathsf{v}})$ that depends on the ciphertext which is being claimed and the receiver's public key (the idea of data-dependent CRS is borrowed from [24]): this prevents malicious users from convincingly claiming other users' ciphertexts. To eliminate an annoying case in the proof of anonymity, we chose to derive the vector $\vec{h}_{\mathsf{v}}$ from a bit string obtained by applying a chameleon hash function [21] (rather than a an ordinary hash function) to (ψ, L, pk).

We build a non-interactive group encryption scheme for the Diffie-Hellman relation $\mathcal{R} = \{(X, Y), W\}$ where $e(g, W) = e(X, Y)$, for which the keys are $\mathsf{pk}_{\mathcal{R}} = \{\mathbb{G}, \mathbb{G}_T, g\}$ and $\mathsf{sk}_{\mathcal{R}} = \varepsilon$.

$\mathsf{SETUP}_{\mathsf{init}}(\lambda)$: Let $\ell \in \mathsf{poly}(\lambda)$ be a polynomial, where $\lambda \in \mathbb{N}$ is the security parameter.

1. Choose bilinear groups $(\mathbb{G}, \mathbb{G}_T)$ of prime order $p > 2^{\lambda}$ with g, g_1, g_2, $\Lambda_0, \Lambda_1 \xleftarrow{R} \mathbb{G}$. Construct a perfectly sound Groth-Sahai CRS $\mathbf{g} = (\vec{g_1}, \vec{g_2}, \vec{g_3})$ using $\vec{g_1} = (g_1, 1, g)$, $\vec{g_2} = (1, g_2, g)$ and $\vec{g_3} = \vec{g_1}^{\xi_1} \odot \vec{g_2}^{\xi_2}$ with $\xi_1, \xi_2 \xleftarrow{R} \mathbb{Z}_p$.
2. For $i = 0$ to ℓ choose $\zeta_{i,1}, \zeta_{i,2} \xleftarrow{R} \mathbb{Z}_p$ and set $\vec{h}_i = \vec{g_1}^{\zeta_{i,1}} \odot \vec{g_2}^{\zeta_{i,2}}$ so as to obtain vectors $\{\vec{h}_i\}_{i=0}^{\ell}$.
3. Choose $\eta_1, \eta_2 \xleftarrow{R} \mathbb{Z}_p$ and compute $\vec{f} = \vec{g_1}^{\eta_1} \odot \vec{g_2}^{\eta_2} = (f_{3,1}, f_{3,2}, f_{3,3})$ so as to form another CRS $\mathbf{f} = (\vec{g_1}, \vec{g_2}, \vec{f})$.
4. Select a strongly unforgeable one time signature $\Sigma = (\mathcal{G}, \mathcal{S}, \mathcal{V})$ and a chameleon hash function $\mathcal{CMH} = (\mathsf{CMKg}, \mathsf{CMhash}, \mathsf{CMswitch})$ with a key pair $(hk, tk) \leftarrow \mathcal{G}(\lambda)$. Public parameters are
$\mathsf{param} = \{\lambda, \mathbb{G}, \mathbb{G}_T, g, \vec{g_1}, \vec{g_2}, \vec{g_3}, \vec{f}, \{\vec{h}_i\}_{i=0}^{\ell}, \Lambda_0, \Lambda_1, \Sigma, \mathcal{CMH}, hk\}$.

$\mathsf{SETUP}_{\mathsf{GM}}(\mathsf{param})$: This algorithm runs the setup algorithm of the structure-preserving signature of Abe *et al.* [1] for messages of length $n = 4$. The secret key is $\mathsf{sk}_{\mathsf{GM}} = \big(\alpha_a, \alpha_b, \gamma_z, \delta_z, \{\gamma_i, \delta_i\}_{i=1}^{4}\big)$ while the public key consists of $\mathsf{pk}_{\mathsf{GM}} = \big(G_r, H_u, G_z, H_z, \{G_i, H_i\}_{i=1}^{4},\ \Omega_a,\ \Omega_b\big) \in \mathbb{G}^8 \times \mathbb{G}_T^2$.

$\mathsf{SETUP}_{\mathsf{OA}}(\mathsf{param})$: generates $\mathsf{pk}_{\mathsf{OA}} = (Y_1, Y_2, Y_3, Y_4) = (g^{y_1}, g^{y_2}, g^{y_3}, g^{y_4})$, as a public key for Kiltz's encryption scheme [20], and the corresponding private key as $\mathsf{sk}_{\mathsf{OA}} = (y_1, y_2, y_3, y_4)$.

JOIN : The prospective user $\mathcal{U}_i$ and the GM run the following protocol.

1. $\mathcal{U}_i$ picks $x_1, x_2, z, \gamma_1, \gamma_2 \xleftarrow{R} \mathbb{Z}_p$ and computes $\mathsf{pk} = (X_1, X_2, \Gamma_1, \Gamma_2)$, where

$$X_1 = g_1^{x_1} \cdot g^z\,, \qquad X_2 = g_2^{x_2} \cdot g^z\,, \qquad \Gamma_1 = g^{\gamma_1}\,, \qquad \Gamma_2 = g^{\gamma_2}\ .$$

   The private key is defined to be $\mathsf{sk} = (x_1, x_2, z, \gamma_1, \gamma_2)$. Here, (X_1, X_2) form a public key for the LY encryption scheme recalled in [23] whereas (Γ_1, Γ_2) will provide user traceability.

2. $\mathcal{U}_i$ defines $\Gamma_0 = g^{\gamma_1\gamma_2}$ and generates a verifiable encryption of Γ_0 under $\mathsf{pk}_{\mathsf{OA}}$. To this end, he chooses $w_1, w_2 \xleftarrow{R} \mathbb{Z}_p$ and computes $\Phi_{venc} = (\Phi_0, \Phi_1, \Phi_2) = \big(\Gamma_0 \cdot g^{w_1+w_2},\ Y_1^{w_1},\ Y_2^{w_2}\big)$. Then, $\mathcal{U}_i$ generates a NIZK proof π_{venc} that Φ_{venc} encrypts Γ_0 such that $e(\Gamma_0, g) = e(\Gamma_1, \Gamma_2)$. Namely, $\mathcal{U}_i$ uses the CRS $\mathbf{f} = (\vec{g_1}, \vec{g_2}, \vec{f})$ to generate GS commitments $\vec{C}_{W_1}, \vec{C}_{W_2}$ to the group elements $W_1 = g^{w_1}$ and $W_2 = g^{w_2}$, respectively, and non-interactively prove that $e(\Phi_0, g) = e(\Gamma_1, \Gamma_2) \cdot e(g, W_1) \cdot e(g, W_2)$ and

$$e(\Phi_1, g) = e(Y_1, W_1) \qquad\qquad e(\Phi_2, g) = e(Y_2, W_2)\ .$$

   These are linear pairing product equations. However, since their proofs must be NIZK proofs, they cost 21 group elements to prove altogether We denote by π_{venc} the resulting NIZK proof. The prospective user $\mathcal{U}_i$ then sends to the group manager a certification request consisting of $\big(\mathsf{pk} = (X_1, X_2, \Gamma_1, \Gamma_2), \Phi_{venc}, \vec{C}_{W_1}, \vec{C}_{W_2}, \pi_{venc}\big)$.

3. If $\mathsf{database}$ already contains a record $\mathsf{transcript}_j$ for which the certified public key $\mathsf{pk}_j = (X_{j,1}, X_{j,2}, \Gamma_{j,1}, \Gamma_{j,2})$ is such that $(X_1, X_2) = (X_{j,1}, X_{j,2})$ or $e(\Gamma_{j,1}, \Gamma_{j,2}) = e(\Gamma_1, \Gamma_2)$, the GM returns $\perp$. Otherwise, the GM generates a certificate $\mathsf{cert}_{\mathsf{pk}} = (Z, R, S, T, U, V, W) \in \mathbb{G}^7$ for pk, which consists of an AHO signature on the tuple $(X_1, X_2, \Gamma_1, \Gamma_2)$. Then, it stores the entire interaction transcript

$$\mathsf{transcript}_i = \Big(\mathsf{pk} = (X_1, X_2, \Gamma_1, \Gamma_2), (\Phi_{venc}, \vec{C}_{W_1}, \vec{C}_{W_2}, \pi_{venc}), \mathsf{cert}_{\mathsf{pk}}\Big)$$

   in $\mathsf{database}$. We also define the DATABASE-CHECK algorithm in such a way that it returns 0 (meaning that $\mathsf{database}$ is not well-formed) if $\mathsf{database}$ contains two distinct records $\mathsf{transcript}_i$ and $\mathsf{transcript}_j$ for which the corresponding public keys $\mathsf{pk}_i = (X_{i,1}, X_{i,2}, \Gamma_{i,1}, \Gamma_{i,2})$ and $\mathsf{pk}_j = (X_{j,1}, X_{j,2}, \Gamma_{j,1}, \Gamma_{j,2})$ are such that $(X_{i,1}, X_{i,2}) = (X_{j,1}, X_{j,2})$ or $e(\Gamma_{i,1}, \Gamma_{i,2}) = e(\Gamma_{j,1}, \Gamma_{j,2})$. Otherwise, it returns 1.

$\mathsf{ENC}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}, \mathsf{cert}_{\mathsf{pk}}, M, L)$: To encrypt $M \in \mathbb{G}$ s.t. $((A, B), M) \in \mathcal{R}_{dh}$ (for public $A, B \in \mathbb{G}$), parse $\mathsf{pk}_{\mathsf{GM}}$, $\mathsf{pk}_{\mathsf{OA}}$ and pk as $(X_1, X_2, \Gamma_1, \Gamma_2) \in \mathbb{G}^4$.

1. Generate a one-time signature key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathcal{G}(\lambda)$.
2. Generate traceability components $(T_1, T_2, T_3, T_4) \in \mathbb{G}^4$ by choosing $\delta, \varrho \xleftarrow{R} \mathbb{Z}_p$ and computing $T_1 = g^\delta$, $T_2 = \Gamma_1^{\delta/\varrho}$, $T_3 = \Gamma_2^{\varrho}$ and $T_4 = (\Lambda_0^{\mathsf{VK}} \cdot \Lambda_1)^\delta$.
3. Compute a LY encryption of M under the label L. Namely,
   (a) Choose $\theta_1, \theta_2 \xleftarrow{R} \mathbb{Z}_p$ and compute $C_0 = M \cdot X_1^{\theta_1} \cdot X_2^{\theta_2}$, $C_1 = g_1^{\theta_1}$, $C_2 = g_2^{\theta_2}$ and $C_3 = g^{\theta_1+\theta_2}$.

(b) Construct a vector $\vec{g}_{\mathsf{VK}} = \vec{g_3} \cdot (1, 1, g)^{\mathsf{VK}}$ and use $\mathbf{g}_{\mathsf{VK}} = (\vec{g_1}, \vec{g_2}, \vec{g}_{\mathsf{VK}})$ as a Groth-Sahai CRS to generate a NIZK proof that $(g, g_1, g_2, C_1, C_2, C_3)$ form a linear tuple. More precisely, generate commitments $\vec{C}_{\theta_1}$, $\vec{C}_{\theta_2}$ to $\theta_1, \theta_2 \in \mathbb{Z}_p$ (namely, compute $\vec{C}_{\theta_i} = \vec{g}_{\mathsf{VK}}^{\theta_i} \cdot \vec{g_1}^{r_i} \cdot \vec{g_2}^{s_i}$ with $r_i, s_i \xleftarrow{R} \mathbb{Z}_p$ for each $i \in \{1, 2\}$) and a proof π_{LIN} that they satisfy

$$C_1 = g_1^{\theta_1}, \qquad C_2 = g_2^{\theta_2}, \qquad C_3 = g^{\theta_1 + \theta_2} . \tag{1}$$

The whole proof for (1) consists of $\vec{C}_{\theta_1}$, $\vec{C}_{\theta_2}$ and π_{LIN} is obtained as

$$\pi_{\text{LIN}} = (\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6) = \left(g_1^{r_1}, g_1^{s_1}, g_2^{r_2}, g_2^{s_2}, g^{r_1 + r_2}, g^{s_1 + s_2}\right) .$$

(c) Define the partial LY ciphertext $\psi_{\mathsf{LY}} = (C_0, C_1, C_2, C_3, \vec{C}_{\theta_1}, \vec{C}_{\theta_2}, \pi_{\text{LIN}})$.

4. For $i = 1, 2$, choose $z_{i,1}, z_{i,2} \xleftarrow{R} \mathbb{Z}_p$ and encrypt Γ_i under $\mathsf{pk}_{\mathsf{OA}}$ using Kiltz's cryptosystem using the same one-time verification key VK as in step 1. Let $\{\psi_{\mathsf{K}_i}\}_{i=1,2}$ be the ciphertexts.
5. Set the TGE ciphertext ψ as $\psi = \mathsf{VK} \| (T_1, T_2, T_3, T_4) \| \psi_{\mathsf{LY}} \| \psi_{\mathsf{K}_1} \| \psi_{\mathsf{K}_2} \| \sigma$ where $\sigma = \mathcal{S}(\mathsf{SK}, ((T_1, T_2, T_3, T_4) \| \psi_{\mathsf{LY}} \| \psi_{\mathsf{K}_1} \| \psi_{\mathsf{K}_2} \| L))$.

Return (ψ, L) and $coins_\psi$ consist of δ, ϱ, $\{(z_{i,1}, z_{i,2})\}_{i=1}^2$ and (θ_1, θ_2). If the one-time signature of [14] is used, the pair (VK, σ) takes 5 group elements, so that ψ comprises 35 elements of $\mathbb{G}$.

$\mathcal{P}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \mathsf{pk}, \mathsf{cert}_{\mathsf{pk}}, (X, Y), M, \psi, L, coins_\psi)$: Parse $\mathsf{pk}_{\mathsf{GM}}$, $\mathsf{pk}_{\mathsf{OA}}$, pk and ψ as above. Using the vectors $\mathbf{f} = (\vec{g_1}, \vec{g_2}, \vec{f})$ as a Groth-Sahai CRS, generate a non-interactive proof for ψ.

1. Parse $\mathsf{cert}_{\mathsf{pk}}$ as $(Z, R, S, T, U, V, W) \in \mathbb{G}^7$ and re-randomize it to obtain $(Z', R', S', T', U', V') \leftarrow \mathsf{ReRand}(\mathsf{pk}_{\mathsf{GM}}, (Z, R, S, T, U, V, W))$ (as explained in [1]). Generate GS commitments $\vec{C}_{Z'}, \vec{C}_{R'}, \vec{C}_{U'}$ to Z', R' and U'. Then, set $com_{\mathsf{cert}_{\mathsf{pk}}} = (\vec{C}_{Z'}, \vec{C}_{R'}, \vec{C}_{U'}, S', T', V', W') \in \mathbb{G}^{13}$.
2. Generate Groth-Sahai commitments to the components of the public key $\mathsf{pk} = (X_1, X_2, \Gamma_1, \Gamma_2)$ and obtain the set $com_{\mathsf{pk}} = \{\vec{C}_{X_i}, \vec{C}_{\Gamma_i}\}_{i=1,2}$, which consists of 12 group elements.
3. Generate a proof $\pi_{\mathsf{cert}_{\mathsf{pk}}}$ that $com_{\mathsf{cert}_{\mathsf{pk}}}$ is a commitment to a valid certificate for the public key contained in com_{pk}. The proof $\pi_{\mathsf{cert}_{\mathsf{pk}}}$ is a NIWI that (Z', R', S', T', U', V') is a valid AHO signature on pk.
4. Generate a NIZK proof π_T that $(T_1, T_2, T_3) = (g^\delta, \Gamma_1^{\delta/\varrho}, \Gamma_2^\varrho)$ for some $\delta, \varrho \in \mathbb{Z}_p$. To this end, generate a commitment $\vec{C}_\Upsilon$ to the group element $\Upsilon = g^{\delta/\varrho}$ and generate a NIZK proof that

$$e(\Upsilon, T_3) = e(T_1, \Gamma_2), \qquad e(T_2, g) = e(\Gamma_1, \Upsilon) .$$

5. For $i = 1, 2$, generate NIZK proofs $\pi_{eq\text{-}key,i}$ that $\vec{C}_{\Gamma_i}$ and ψ_{K_i} are encryptions of the same Γ_i. If $\psi_{\mathsf{K}_i} = (V_{i,0}, V_{i,1}, V_{i,2}, V_{i,3}, V_{i,4})$ is a Kiltz encryption comprising $\left(V_{i,0}, V_{i,1}, V_{i,2}\right) = \left(\Gamma_i \cdot g^{z_{i,1} + z_{i,2}}, Y_1^{z_{i,1}}, Y_2^{z_{i,2}}\right)$ and $\vec{C}_{\Gamma_i}$ is parsed as $(c_{\Gamma_{i1}}, c_{\Gamma_{i2}}, c_{\Gamma_{i3}}) = \left(g_1^{\rho_{i1}} \cdot f_{3,1}^{\rho_{i3}},\ g_2^{\rho_{i2}} \cdot f_{3,2}^{\rho_{i3}},\ \Gamma_i \cdot g^{\rho_{i1} + \rho_{i2}} \cdot f_{3,3}^{\rho_{i3}}\right)$, where $z_{i,1}, z_{i,2} \in coins_\psi$, $\rho_{i1}, \rho_{i2}, \rho_{i3} \in \mathbb{Z}_p$ and $\vec{f} = (f_{3,1}, f_{3,2}, f_{3,3})$, this

amounts to prove knowledge of values $z_{i,1}, z_{i,2}, \rho_{i1}, \rho_{i2}, \rho_{i3} \in \mathbb{Z}_p$ such that $\left(\frac{V_{i,1}}{c_{\Gamma_{i1}}}, \frac{V_{i,2}}{c_{\Gamma_{i2}}}, \frac{V_{i,0}}{c_{\Gamma_{i3}}}\right)$ is of the form

$$\left(Y_1^{z_{i,1}} \cdot g_1^{-\rho_{i1}} \cdot f_{3,1}^{-\rho_{i3}}, Y_2^{z_{i,2}} \cdot g_2^{-\rho_{i2}} \cdot f_{3,2}^{-\rho_{i3}}, g^{z_{i,1}+z_{i,2}-\rho_{i1}-\rho_{i2}} \cdot f_{3,3}^{-\rho_{i3}}\right) .$$

6. Generate a NIZK proof $\pi_{\mathcal{R}}$ that ψ_{LY} encrypts a group element $M \in \mathbb{G}$ such that $((A, B), M) \in \mathcal{R}$. To this end, generate a commitment $com_M = (c_{M,1}, c_{M,2}, c_{M,3}) = \left(g_1^{\rho_1} \cdot f_{3,1}^{\rho_3}, g_2^{\rho_2} \cdot f_{3,2}^{\rho_3}, M \cdot g^{\rho_1+\rho_2} \cdot f_{3,3}^{\rho_3}\right)$ and prove that the underlying M is the same as the one for which $C_0 = M \cdot X_1^{\theta_1} \cdot X_2^{\theta_2}$ in ψ_{LY}. In other words, prove knowledge of $\theta_1, \theta_2, \rho_1, \rho_2, \rho_3$ such that $\left(C_1, C_2, \frac{C_1}{c_{M,1}}, \frac{C_2}{c_{M,2}}, \frac{C_0}{c_{M,3}}\right)$ equals

$$\left(g_1^{\theta},\ g_2^{\theta},\ g_1^{\theta_1-\rho_1} \cdot f_{3,1}^{-\rho_3},\ g_2^{\theta_2-\rho_2} \cdot f_{3,2}^{-\rho_3},\ g^{-\rho_1-\rho_2} \cdot f_{3,3}^{-\rho_3} \cdot X_1^{\theta_1} \cdot X_2^{\theta_2}\right) .$$

The entire proof $\pi_\psi = com_{\mathsf{cert}_{\mathsf{pk}}} \| com_{\mathsf{pk}} \| \pi_{\mathsf{cert}_{\mathsf{pk}}} \| \pi_T \| \pi_{eq\text{-}key,1} \| \pi_{eq\text{-}key,2} \| \pi_{\mathcal{R}}$ takes 150 elements.

$\mathcal{V}(\mathsf{param}, \psi, L, \pi_\psi, \mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}})$: Parse $\mathsf{pk}_{\mathsf{GM}}$, $\mathsf{pk}_{\mathsf{OA}}$, pk, ψ and π_ψ as above. Return 1 if and only if the conditions below are all satisfied.

1. $\mathcal{V}(\mathsf{VK}, \sigma, ((T_1, T_2, T_3, T_4) \| \psi_{\mathsf{LY}} \| \psi_{\mathsf{K}_1} \| \psi_{\mathsf{K}_2} \| L)) = 1$.
2. $e(T_1, \Lambda_0^{\mathsf{VK}} \cdot \Lambda_1) = e(g, T_4)$ and ψ_{LY} is a valid LY ciphertext.
3. All proofs verify and if $\{\psi_{\mathsf{K}_i}\}_{i=1}^2$ are valid Kiltz encryptions w.r.t. VK.

$\mathsf{DEC}(\mathsf{sk}, \psi, L)$: Parse ψ as $\mathsf{VK} \| (T_1, T_2, T_3, T_4) \| \psi_{\mathsf{LY}} \| \psi_{\mathsf{K}_1} \| \psi_{\mathsf{K}_2} \| \sigma$. Return $\perp$ in the event that either: (i) $\mathcal{V}(\mathsf{VK}, \sigma, ((T_1, T_2, T_3, T_4) \| \psi_{\mathsf{LY}} \| \psi_{\mathsf{K}_1} \| \psi_{\mathsf{K}_2} \| L)) = 0$; (ii) $e(T_1, \Lambda_0^{\mathsf{VK}} \cdot \Lambda_1) \neq e(g, T_4)$ or ψ_{LY} and $\{\psi_{\mathsf{K}_i}\}_{i=1,2}$ are not all valid ciphertexts. Otherwise, use sk to decrypt (ψ_{LY}, L).

$\mathsf{REVEAL}(\mathsf{transcript}_i, \mathsf{sk}_{\mathsf{OA}})$: Parse $\mathsf{transcript}_i$ as

$$\left((X_{i,1}, X_{i,2}, \Gamma_{i,1}, \Gamma_{i,2}), (\Phi_{venc,i}, \vec{C}_{W_{i,1}}, \vec{C}_{W_{i,2}}, \pi_{venc,i}), \mathsf{cert}_{\mathsf{pk},i}\right) .$$

Parse $\Phi_{venc,i}$ as a BBS ciphertext $(\Phi_{i,0}, \Phi_{i,1}, \Phi_{i,2}) \in \mathbb{G}^3$ and verify that $(\vec{C}_{W_{i,1}}, \vec{C}_{W_{i,2}}, \pi_{venc,i})$ form a valid proof fo. If not, return $\perp$. Otherwise, use $\mathsf{sk}_{\mathsf{OA}} = (y_1, y_2, y_3, y_4)$ to compute $\Gamma_{i,0} = \Phi_{i,0} \cdot \Phi_{i,1}^{-1/y_1} \cdot \Phi_{i,2}^{-1/y_2}$. Return the resulting plaintext $\mathsf{trace}_i = \Gamma_{i,0} \in \mathbb{G}$ which can serve as a tracing trapdoor for user i as it is necessarily of the form $\Gamma_{i,0} = \Gamma_{i,2}^{\log_g(\Gamma_{i,1})}$.

$\mathsf{TRACE}(\mathsf{pk}_{\mathsf{GM}}, \mathsf{pk}_{\mathsf{OA}}, \psi, \mathsf{trace}_i)$: Given $\psi = \mathsf{VK} \| (T_1, T_2, T_3, T_4) \| \psi_{\mathsf{LY}} \| \psi_{\mathsf{K}_1} \| \psi_{\mathsf{K}_2} \| \sigma$ and the tracing trapdoor trace_i as a group element $\Gamma_{i,0} \in \mathbb{G}$. If the equality $e(T_1, \Gamma_{i,0}) = e(T_2, T_3)$ holds, it returns 1. Otherwise, it outputs 0.

$\mathsf{OPEN}(\mathsf{sk}_{\mathsf{OA}}, \psi, L)$: Parse ψ as $\mathsf{VK} \| (T_1, T_2, T_3, T_4) \| \psi_{\mathsf{LY}} \| \psi_{\mathsf{K}_1} \| \psi_{\mathsf{K}_2} \| \sigma$. Return $\perp$ if $\{\psi_{\mathsf{K}_i}\}_{i=1}^2$ are not both valid ciphertexts w.r.t. VK or if σ is an invalid one-time signature for VK. Otherwise, decrypt $\{\psi_{\mathsf{K}_i}\}_{i=1,2}$ to obtain $\Gamma_1, \Gamma_2 \in \mathbb{G}$ and look up $\mathsf{database}$ in order to find a record $\mathsf{transcript}_i$ containing a key $\mathsf{pk}_i = (X_{i,1}, X_{i,2}, \Gamma_{i,1}, \Gamma_{i,2})$ such that $(\Gamma_{i,1}, \Gamma_{i,2}) = (\Gamma_1, \Gamma_2)$ (note that, unless $\mathsf{database}$ is ill-formed, such a record is unique if it exists). If such a record is found, output the matching i. Otherwise, output $\perp$.

$\mathsf{CLAIM/DISCLAIM}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \psi, L, \mathsf{sk})$: Given $\mathsf{sk} = (x_1, x_2, z, \gamma_1, \gamma_2)$, parse ψ as $\mathsf{VK}\|(T_1, T_2, T_3, T_4)\|\psi_{\mathsf{LY}}\|\psi_{\mathsf{K}_1}\|\psi_{\mathsf{K}_2}\|\sigma$. To generate a claim/disclaimer τ for the ciphertext ψ, first verify that $e(T_1, \Lambda_0^{\mathsf{VK}} \cdot \Lambda_1) = e(g, T_4)$ and that σ is a valid one-time signature. If these conditions, do not hold, return $\perp$. Otherwise, compute $T_{\delta,1} = T_1^{\gamma_1} = \Gamma_1^{\delta}$, where $\delta = \log_g(T_1)$. Then, compute a collision-resistant hash $\mathsf{v} = \mathsf{CMhash}(hk, (\psi, L, \mathsf{pk}), s_{hash}) \in \{0,1\}^{\ell}$, where $s_{hash} \xleftarrow{R} \mathcal{R}_{hash}$. Then, parse v as $\mathsf{v}[1] \ldots \mathsf{v}[\ell] \in \{0,1\}^{\ell}$ and assemble the vector $\vec{h}_{\mathsf{v}} = \vec{h}_0 \odot \bigodot_{i=1}^{\ell} \vec{h}_i^{\mathsf{v}[i]}$. Using $(\vec{g_1}, \vec{g_2}, \vec{h}_{\mathsf{v}})$ as a Groth-Sahai CRS, generate a commitment $\vec{C}_{\Gamma_{-1}}$ to $\Gamma_{-1} = g^{1/\gamma_1}$ and a NIZK proof that Γ_{-1} satisfies $e(T_{\delta,1}, \Gamma_{-1}) = e(T_1, g)$. To this end, generate a commitment $\vec{C}_{\mathcal{X}_\tau}$ to the auxiliary variable $\mathcal{X}_\tau = g$ and non-interactive proofs $\pi_{\tau,1}$, $\pi_{\tau,2}$ for the equations

$$e(T_{\delta,1}, \Gamma_{-1}) = e(T_1, \mathcal{X}_\tau)\,, \qquad e(g, \mathcal{X}_\tau) = e(g, g)\ . \tag{2}$$

The claim/disclaimer is $\tau = \left(T_{\delta,1}, \vec{C}_{\Gamma_{-1}}, \vec{C}_{\mathcal{X}_\tau}, \pi_{\tau,1}, \pi_{\tau,2}, s_{hash}\right) \in \mathbb{G}^{14}$.

$\mathsf{CLAIM\text{-}VERIFY}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \psi, L, \mathsf{pk}, \tau)$: Given $\mathsf{pk} = (X_1, X_2, \Gamma_1, \Gamma_2)$ and the ciphertext $\psi = \mathsf{VK}\|(T_1, T_2, T_3, T_4)\|\psi_{\mathsf{LY}}\|\psi_{\mathsf{K}_1}\|\psi_{\mathsf{K}_2}\|\sigma$, parse τ as above. Return 1 if and only if $e(T_{\delta,1}, \Gamma_2) = e(T_2, T_3)$ and $e(T_1, \Gamma_1) = e(g, T_{\delta,1})$ and $\pi_{\tau,1}, \pi_{\tau,2}$ are valid proofs for (2) w.r.t. the Groth-Sahai CRS $(\vec{g_1}, \vec{g_2}, \vec{h}_{\mathsf{v}})$, where $\vec{h}_{\mathsf{v}} = \vec{h}_0 \odot \bigodot_{i=1}^{\ell} \vec{h}_i^{\mathsf{v}[i]}$ and $\mathsf{v} = \mathsf{CMhash}(hk, (\psi, L, \mathsf{pk}), s_{hash}) \in \{0,1\}^{\ell}$.

$\mathsf{DISCLAIM\text{-}VERIFY}(\mathsf{pk_{GM}}, \mathsf{pk_{OA}}, \psi, L, \mathsf{pk}, \tau)$: Parse pk, ψ and τ as previously. Return 1 if and only if $e(T_{\delta,1}, \Gamma_2) \neq e(T_2, T_3)$, $e(T_1, \Gamma_1) = e(g, T_{\delta,1})$ and $\pi_{\tau,1}, \pi_{\tau,2}$ are valid proofs for (2) and the Groth-Sahai CRS $(\vec{g_1}, \vec{g_2}, \vec{h}_{\mathsf{v}})$, where $\vec{h}_{\mathsf{v}} = \vec{h}_0 \odot \bigodot_{i=1}^{\ell} \vec{h}_i^{\mathsf{v}[i]}$ and $\mathsf{v} = \mathsf{CMhash}(hk, (\psi, L, \mathsf{pk}), s_{hash}) \in \{0,1\}^{\ell}$.

The length of ciphertexts is about 2.18 kB using symmetric pairings with a 512-bit representation for each group element (at the 128-bit security level). Our proofs only require 9.38 kB (against roughly 32 kB for the same security in [9]). More detailed comparisons with [19,9] are given in the full version of the paper.

The correctness of the scheme stems from that of Groth-Sahai proofs. From a security point of view, we prove the security properties under the q-SFP, D3DH and DLIN assumptions and also require the one-time signatures to be strongly unforgeable. All proofs are given in the full version of the paper.

References

1. M. Abe, K. Haralambiev, M. Ohkubo. Signing on elements in bilinear groups for modular protocol design. Cryptology ePrint Archive: Report 2010/133, 2010.
2. M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, M. Ohkubo. Structure-preserving signatures and commitments to group elements. In *Crypto'10*, LNCS 6223, 2010.
3. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *Asiacrypt'01*, LNCS 2248, 2001.
4. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS'93*, 1993.

5. V. Benjumea, S. G. Choi, J. Lopez, M. Yung. Fair traceable multi-group signatures. In *Financial Cryptography 2008*, LNCS 5143, Springer, 2008.
6. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Crypto'04*, LNCS 3152, 2004.
7. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003. Extended abstract in Crypto'01, LNCS 2139, 2001.
8. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN'02*, LNCS 2576, 2003.
9. J. Cathalo, B. Libert, M. Yung. Group encryption: Non-interactive realization in the standard model. In *Asiacrypt'09*, LNCS 5912, 2009.
10. D. Chaum and E. van Heyst. Group signatures. In *Eurocrypt'91*, LNCS 547, 1991.
11. R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Crypto'98*, LNCS 1462, 1998.
12. L. El Aimani, M. Joye. Toward practical group encryption. In *ACNS 2013*, LNCS 7954, 2013.
13. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Crypto'86*, LNCS 263, 1986.
14. J. Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *Asiacrypt'06*, LNCS 4284, 2006.
15. J. Groth. Fully anonymous group signatures without random oracles. In *Asiacrypt'07*, LNCS 4833, 2007.
16. J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *Eurocrypt'08*, LNCS 4965, 2008.
17. M. Izabachène, D. Pointcheval, D. Vergnaud. Mediated traceable anonymous encryption. In *Latincrypt'08*, LNCS 6212, 2010.
18. A. Kiayias, Y. Tsiounis, and M. Yung. Traceable signatures. In *Eurocrypt 2004*, LNCS 3027, Springer, 2004.
19. A. Kiayias, Y. Tsiounis, and M. Yung. Group encryption. In *Asiacrypt'07*, LNCS 4833, 2007.
20. E. Kiltz. Chosen-ciphertext security from tag-based encryption. In *TCC'06*, LNCS 3876, 2006.
21. H. Krawczyk and T. Rabin. Chameleon signatures. In *NDSS'00*, 2000.
22. B. Libert and M. Yung. Efficient Traceable Signatures in the Standard Model. In *Pairing'09*, LNCS 5671, 2009.
23. B. Libert, M. Yung. Non-interactive CCA2-secure threshold cryptosystems with adaptive security: new framework and constructions. In *TCC'12*, LNCS 7194, Springer, 2012.
24. T. Malkin, I. Teranishi, Y. Vahlis, M. Yung. Signatures resilient to continual leakage on memory and computation. In *TCC'11*, LNCS 6597, 2011.
25. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Eurocrypt'99*, LNCS 1592, 1999.
26. B. Qin, Q. Wu, W. Susilo, Y. Mu, Y. Wang. Publicly verifiable privacy-preserving group decryption. In *Inscrypt'08*, LNCS 5487, 2008.
27. M. Trolin, D. Wiström. Hierarchical group signatures. In *ICALP'05*, LNCS 3580, 2005.