EasyChair Preprint

№ 14672

Group Signatures with Designated Traceability over Openers' Attributes from Symmetric-Key Primitives

Hiroaki Anada, Masayuki Fukumitsu and Shingo Hasegawa

September 3, 2024

Group Signatures with Designated Traceability over Openers' Attributes from Symmetric-Key Primitives

Hiroaki Anada
Department of Mathematical Informatics
Meiji Gakuin University
Yokohama, 244-8539 Japan
hiroaki.anada@mi.meijigakuin.ac.jp

Masayuki Fukumitsu
Department of Information Security
University of Nagasaki
Nagayo, 851-2195 Japan
fukumitsu@sun.ac.jp

Shingo Hasegawa
Faculty of Symbiotic Systems Science
Fukushima University
Fukushima, 960-1296 Japan
hasegawa@sss.fukushima-u.ac.jp

***Abstract*—A group signature scheme in which signers are able to designate openers by specifying access structures over openers' attributes was introduced at CANDAR 2021, which is called GSdT. In this paper, we present a construction of GSdT from only symmetric-key primitives; pseudorandom functions, hash functions and commitments. Due to the features, our GSdT is expected to be secure against computational power of quantum computers. We first introduce syntax and security definitions in the static group model. Then, in our construction, the key ingredient is a non-interactive zero-knowledge proof of knowledge system that is constructed from the primitives in the "MPC-in-the-head" paradigm, owing the technique that was developed by Katz, Kolesnikov and Wang (ACM-CCS 2018). Our approach starts with their group signature scheme, but non-trivially extends the Merkle tree so that signers can treat (all-AND) boolean formulas as the access structures. According to our estimation, the signing time is less than 3.0 sec and the signature size is less than 0.5 MB in a scenario that the numbers of group members and attributes are 2^7 and 2^3, respectively, and security to be attained is 128 bit quantum security.**

***Index Terms*—group signatures, anonymity, traceability, accountability, attribute-based, post-quantum, symmetric-key primitives**

I. Introduction

Anonymity and *traceability* are the fundamental properties required to cryptography. However, it is seemingly difficult to make these two properties compatible. A typical example is the group signature. The group signature is a kind of digital signatures, proposed by Chaum and van Heyst [1] and formalized through the work of Bellare-Micciancio-Warinschi [2]. The group signature has both anonymity and traceability as its security requirements, but the excessive power of tracing by openers forces the anonymity opaque. This is because signers are unable to know by whom and when their signatures are opened. The feature motivated the study of balancing the anonymity and the traceability such as accountable ring signatures [3], group signatures with message-dependent opening [4], bifurcated anonymous signatures [5] and multimodal private signatures [6].

The group signatures with designated traceability over openers' attributes (GSdT) that were proposed by Anada, Fukumitsu and Hasegawa at CANDAR 2021 [7] (and [8]) are group signatures by which signers are able to designate openers with access structures. That is, the signers can control traceability by selecting the openers' attributes in the signing phase. This characteristic is expected to help accountability in the way that the openers with satisfying attributes can claim validity of opening because they are legitimately selected by signers themselves. In [7], [8], the notion of GSdT was introduced with the syntax of schemes and the definitions of security requirements. Also, the public-key based generic construction of GSdT was given, namely a construction from a ciphertext-policy attribute-based encryption (CP-ABE), a digital signature and a non-interactive zero-knowledge proof (NIZK). Then, the pairing-based instantiation was proposed by [9].

On the other hand, one of the most important research topics of the modern cryptography is the post-quantum cryptography. This is because the basic problems of the integer factorization and the discrete logarithm can be solved in polynomial-time by applying the Shor's algorithm [10] on a quantum computer. On GSdT, the known concrete construction is not quantum computer resistant as described above. Thus to construct a post-quantum GSdT is a remaining problem that should be addressed.

The aim of this paper is a post-quantum construction of GSdT. There are two approaches for the purpose. One is the quantum-resistant public-key based approach such as the lattice-based one. This approach basically follows the known generic constructions [7], [11]. The other approach is to employ the *symmetric-key* primitives such as pseudorandom functions (PRFs), hash functions (HFs) and commitments (Cmts).

A. Our contribution

In this paper, we adopt the latter approach, which is justified due to the following situation. In Round 3 Submissions of NIST Post-Quantum Cryptography [12], "Alternate Candidates: Digital Signature Algorithms" have been selected. The Picnic signature algorithm [13] is one of the selection, which

is constructed from only the symmetric-key primitives. As is explained in the specification [13], the feature of Picnic is "conservative" assumptions on PRFs, HFs and Cmts, which means assumptions on symmetric-key primitives. The primitives can be instantiated from, for example, LowMC[1] [14] and SHAKE [15]. Thus, the approach is considered to be one of the hopeful directions of research for post-quantum cryptography.

Along the approach, our construction of GSdT is sketched as follows. It is basically a non-trivial extension of the group signature scheme (GS) that was given by Katz, Kolesnikov and Wang [16] (KKW), and a subsequent work by Anada, Fukumitsu and Hasegawa [17] for an accountable ring signature scheme from the symmetric-key primitives, where the main idea is for a group member to generate a signature by generating a non-interactive zero-knowledge proof of knowledge (NIZKPoK) of a secret key corresponding to the group public key. There the NIZKPoK is constructed from the symmetric-key primitives, and the properties of knowledge-soundness and computational zero-knowledge follow from the properties of them, in the random oracle model. A key observation must be on an effective usage of a Merkle tree [18] to shorten the size of a signature and to reduce the amount of computation to generate a signature. Analysing the construction of the Merkle tree, we extend the tree by patching subtrees whose leaves have labels that are hash values of random private-key strings each of which is for every attributes and for every group members (see Fig. 2). Then we can naturally (but non-trivially) apply the strategy of KKW [16].

Due to the simplicity of the static group model in which the keys of group members and openers are generated and fixed in the group-key generation phase, we will introduce syntax and security definitions for the static case, for the first time. More precisely, referring to the previous work by Bellare, Micciancio and Warinschi [2], we give definitions of correctness, anonymity and traceability for our setting of designated traceability over openers' attributes. Intuitively, the correctness is the property that honestly generated group signatures are accepted and can be traced to actual signers, with probability one. The anonymity is the property that, given a group signature, it is computationally difficult for any PPT adversary to correctly guess the actual signer with non-trivial probability. In our construction of GSdT, the property is guaranteed due to the zero-knowledge property of the employed NIZKPoK of KKW [16]. The traceability is the property that it is computationally difficult for any PPT adversary to generate a valid group signature that is traced to a group member who is not corrupted. In our construction, the property is guaranteed due to the knowledge-soundness of the NIZKPoK in the random oracle model and collision-resistance of hash functions.

Let l denote the number of group members and let k denote the number of attributes that appears in an all-AND boolean formula associated to a group signature of our GSdT. The number of leaf nodes of our GSdT is equal to lk, while that of the KKW GS is l. Since the structure of the above Merkle tree is similar to that of the KKW GS, we estimate that the asymptotic behaviors of the signing time and the size of a signature follow the behaviors of the KKW GS. Here in the case of the KKW GS the asymptotic behaviors are proportional to the number of leaf nodes, l. Therefore the signing time and the size of a signature of our GSdT are proportional to lk. Moreover, the expected concrete performance can be estimated based on the implementation results of the group signature scheme in the previous [16]. As a result, the signing time is expected to be less than 3.0 sec and the signature size is less than 0.5 MB in the situation that the number of group members is 2^7, the number of designated attributes is 2^3 and the target security is 128 bit quantum security. Thus, our GSdT from the symmetric-key primitives is considered to be efficient in realistic scenarios.

Finally, we note that the description of our GSdT in the later sections is for the case of all-AND formulas when the signers designate openers' attributes. To describe the construction for the case of general boolean formulas should be our future work.

[1]Low multiplicative complexity.

II. Preliminaries

$\mathbb{N}$ denotes the set of natural numbers. We write $[n]$ to denote the set of n numbers $\{1, \ldots, n\} \subset \mathbb{N}$. $\lambda (\in \mathbb{N})$ stands for the security parameter. Let *pp* be the set of public parameters. The number of elements in a finite set S is denoted by $|S|$. ε is the empty string. The concatenation of a string a followed by a string b is written by $a \parallel b$. $a \in_R S$ means that a uniform random sampling of an element a from a finite set S. We denote by $z \leftarrow A(a)$, or $A(a) \rightarrow z$ that an algorithm A with input a returns z. When a probabilistic algorithm A with input a and randomness r on a random tape returns z, we define it as $z \leftarrow A(a; r)$. Let $\mathbf{A}^{\mathsf{O}}$ denote an algorithm $\mathbf{A}$ which is allowed to access an oracle O. St stands for the inner state of an algorithm. "Probabilistic polynomial-time" is abbreviated as PPT.

A. Pseudo-random Functions [19]

We treat pseudo-random functions as a family. For a security parameter $\lambda \in \mathbb{N}$ and a key $k_\lambda \in \{0,1\}^\lambda$, we define a function $\mathsf{PRF}^\lambda_{k_\lambda} : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$. Then $\mathcal{PRF} = \{\mathsf{PRF}^\lambda_{k_\lambda}\}_{\lambda \in \mathbb{N}}$ is a function family. We also employ a uniform λ-bit function family $\mathcal{UF} = \{\mathsf{UF}^\lambda\}$. Namely, $\mathsf{UF}^\lambda : \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda$ maps any λ-bit string into a uniformly distributed λ-bit string. $\mathcal{PRF}$ should satisfy the following requirement.
(PRF: pseudorandomess) For any given PPT algorithm $\mathbf{A}$, $\mathbf{Adv}^{\text{rand}}_{\mathsf{PRF},\mathbf{A}}(\lambda) := |\Pr[\mathbf{A}^{\mathsf{PRF}^\lambda_{k_\lambda}}(1^\lambda) = 1] - \Pr[\mathbf{A}^{\mathsf{UF}^\lambda}(1^\lambda) = 1]|$ is negligible in λ.

The superscript $^\lambda$ of $\mathsf{PRF}^\lambda_{k_\lambda}$ is omitted as PRF_{k_λ}.

B. Hash Functions [20]

We also treat hash functions as a family. For a security parameter $\lambda \in \mathbb{N}$, we define a function $\mathsf{H}^\lambda : \{0,1\}^* \rightarrow \{0,1\}^\lambda$, and then $\mathsf{H} = \{\mathsf{H}^\lambda\}_{\lambda \in \mathbb{N}}$ is set as a function family. H should satisfy the following two requirements.

(HF1: Preimage resistance) For any given PPT algorithm $\mathbf{A}$, $\mathbf{Adv}^{\mathrm{preim}}_{\mathsf{H},\mathbf{A}}(\lambda) := \Pr[\mathsf{H}^\lambda(x) = y \mid y \in_R \{0,1\}^\lambda;\ x \leftarrow \mathbf{A}(y)]$ is negligible in λ.
(HF2: Collision resistance) For any given PPT algorithm $\mathbf{A}$, $\mathbf{Adv}^{\mathrm{col}}_{\mathsf{H},\mathbf{A}}(\lambda) := \Pr[x_1 \neq x_2 \wedge \mathsf{H}^\lambda(x_1) = \mathsf{H}^\lambda(x_2) \mid (x_1, x_2) \leftarrow \mathbf{A}]$ is negligible in λ.

The superscript $^\lambda$ of H^λ is omitted as H.

C. Merkle Trees [18]

A Merkle tree is one of the data structures of trees that are basically binary. Each leaf of the tree has a hash value $h \leftarrow \mathsf{H}(y)$ of some value y. Then, for each non-leaf node that has two children whose hash values are represented by h_1 and h_2, a hash function $h_p \leftarrow \mathsf{H}(h_1 \parallel h_2)$ is attached to the non-leaf node. The hash value attached to the Markle root is expressed by h^*. Path_j is said to be a Merkle proof at a leaf j or a Merkle path at a leaf j denoted as Path_j. Path_j is the minimum set of values to reconstruct h^*. Namely, it includes the value attached to the leaf j and all the values of the successive parents, the values attached to all the siblings. For the convenience, we also include h^* in Path_j.

D. Commitments [19]

Let $p(\cdot)$ be a polynomial. A commitment scheme Cmt consists of two PPT algorithms; Com and Vrf.

- $\mathsf{Com}(m; r) \rightarrow \mathsf{com}$. On input a message $m \in \{0,1\}^{p(\lambda)}$ with a randomness r, this PPT algorithm generates a commitment com, and then returns com.
- $\mathsf{Vrf}(\mathsf{com}, m, r) \rightarrow d$. On input a commitment com, a message m and a randomness r that is used to generate com, this deterministic polynomial-time algorithm generates a boolean decision d, and then returns d.

Cmt should satisfy the following three requirements.
(Com1: Correctness) For any 1^λ, any $m \in \{0,1\}^{p(\lambda)}$ and any r, $\Pr[d = 1 \mid \mathsf{com} \leftarrow \mathsf{Com}(m; r);\ d \leftarrow \mathsf{Vrf}(\mathsf{com}, m, r)] = 1$.
(Com2: Hiding) We define the following experimental algorithm $\mathsf{Exp}^{\mathrm{hide}}_{\mathsf{Cmt},\mathbf{A}}$ for any given algorithm $\mathbf{A}$.

$\mathsf{Exp}^{\mathrm{hide}}_{\mathsf{Cmt},\mathbf{A}}(1^\lambda)$
$(m_0, m_1, \mathrm{St}) \leftarrow \mathbf{A}(1^\lambda);\ b \in_R \{0,1\};\ \mathsf{com} \leftarrow \mathsf{Com}(m_b; r)$
$b' \leftarrow \mathbf{A}(\mathrm{St}, \mathsf{com});$ If $b = b'$ then return 1 else return 0

Then, Cmt is said to be computationally hiding if, for any PPT algorithm $\mathbf{A}$, the advantage of $\mathbf{A}$, which is given by $\mathbf{Adv}^{\mathrm{hide}}_{\mathsf{Cmt},\mathbf{A}}(\lambda) := |\Pr[\mathsf{Exp}^{\mathrm{hide}}_{\mathsf{Cmt},\mathbf{A}}(1^\lambda) = 1] - (1/2)|$, is negligible in λ.
(Com3: Binding) We define the following experimental algorithm $\mathsf{Exp}^{\mathrm{bind}}_{\mathsf{Cmt},\mathbf{A}}$ for any given algorithm $\mathbf{A}$.

$\mathsf{Exp}^{\mathrm{bind}}_{\mathsf{Cmt},\mathbf{A}}(1^\lambda)$
$(\mathsf{com}, m, r, m', r') \leftarrow \mathbf{A}(1^\lambda);$ If $\mathsf{Vrf}(\mathsf{com}, m, r) =$
$\mathsf{Vrf}(\mathsf{com}, m', r') = 1 \wedge m \neq m'$ then return 1 else return 0

Then, Cmt is said to be computationally binding if, for any PPT algorithm $\mathbf{A}$, the advantage of $\mathbf{A}$, which is given as $\mathbf{Adv}^{\mathrm{bind}}_{\mathsf{Cmt},\mathbf{A}}(\lambda) := \Pr[\mathsf{Exp}^{\mathrm{bind}}_{\mathsf{Cmt},\mathbf{A}}(1^\lambda) = 1]$, is negligible in λ.

E. Zero-Knowledge Proofs from "MPC-in-the-Head" [16], [21]

Zero-knowledge Proofs from MPC-in-the-Head are zero-knowledge proofs proposed by Ishai, Kushilevitz, Ostrovsky and Sahai ([21], IKOS). We employ the modified version proposed by Katz, Kolesnikov and Wang ([16], KKW), which is summarized below. The protocol Π_{ZK} is a three-round protocol for which a prover convinces the possession of a witness w satisfying a given circuit C. Π_{ZK} is an honest-verifier zero-knowledge (HVZK) proof of knowledge. Π_{ZK} is generically constructed from Π_{MPC} that is secure against semi-honest corruption of all-but-one of the n parties, and can evaluate the circuit C in the "preprocessing model" (see Appendix A.2 of [16]). Let $P_1, \ldots, P_n$ be the parties in Π_{MPC}. In such a preprocessing phase, for each $i \in [n]$, a seed seed_i is chosen for the party P_i to derive a longer randomness used in Π_{MPC}, and then a $|C|$-bit string aux for the party P_n is also generated, where $|C|$ denotes the number of AND-gates of C. (For more details, refer to [16].) In the KKW protocol, a prover P emulates the preprocessing phase, then M copies of Π_{MPC} are executed. A verifier V in Π_{ZK} selects τ emulators whose transcript is opened, where $\tau \in [M]$. Thus, Π_{ZK} is specified by the parameters (M, n, τ).

3-round HVZK protocol of [16] for a circuit C.
Input) A prover P is given a circuit C and a witness s.t. $C(w) = 1$, whereas a verifier V is given only the circuit C.
Round 1: P) For $j \in [M]$, the following processes are run:

1) Choose $\mathsf{seed}^*_j \in_R \{0,1\}^\lambda$ and then pseudo-randomly generate $\mathsf{seed}_{j,1}, r_{j,1}, \ldots, \mathsf{seed}_{j,n}, r_{j,n} \in \{0,1\}^\lambda$ by using seed^*_j. Also, generate $\mathsf{aux}_j \in \{0,1\}^{|C|}$ by the specific way explained in Section 2 of [16]. Set $\mathsf{state}_{j,i} := \mathsf{seed}_{j,i}$ for $i \in [n-1]$. As for $i = n$ set $\mathsf{state}_{j,n} := \mathsf{seed}_{j,n} \parallel \mathsf{aux}_j$.
2) For $i \in [n]$, set $\mathsf{com}_{j,i} \leftarrow \mathsf{Com}(\mathsf{state}_{j,i}; r_{j,i})$.
3) Simulate the online phase of Π_{MPC} by using $(\mathsf{state}_{j,i})_{i \in [n]}$. During that, store the messages broadcast by P_i as $\mathsf{msgs}_{j,i}$.
4) Set $h_j \leftarrow \mathsf{H}(\mathsf{com}_{j,1} \parallel \cdots \parallel \mathsf{com}_{j,n})$, $r_j \in_R \{0,1\}^\lambda$ and $h'_j \leftarrow \mathsf{H}((\hat{z}_{j,\alpha})_{\alpha \in [|C|]} \parallel \mathsf{msgs}_{j,1} \parallel \ldots \parallel \mathsf{msgs}_{j,n} \parallel r_j)$, where $\hat{z}_{j,\alpha}$ is a "masked value" that is formalized in Section 2.1 of [16].

Set $h \leftarrow \mathsf{H}(h_1 \parallel \cdots \parallel h_M)$, $h' \leftarrow \mathsf{H}(h'_1 \parallel \cdots \parallel h'_M)$ and $h^* \leftarrow \mathsf{H}(h, h')$, and then send h^* to V.
Round 2: V) Choose a subset $\mathcal{C} \subset [M]$ including τ elements uniformly at random. For $j \in \mathcal{C}$, choose $p_j \in [n]$ uniformly at random. Set $\mathcal{P} := (p_j)_{j \in \mathcal{C}}$, then send $(\mathcal{C}, \mathcal{P})$ to P.
Round 3: P) Send $(\mathsf{seed}^*_j, h'_j)_{j \in [M] \backslash \mathcal{C}}$ and $((\mathsf{state}_{j,i}, r_{j,i})_{i \neq p_j}, \mathsf{com}_{j,p_j}, (\hat{z}_{j,\alpha})_{\alpha \in [|C|]}, r_j, \mathsf{msgs}_{j,p_j})_{j \in \mathcal{C}}$ to V.
Verify: V) the following processes are run:

1) For $j \in \mathcal{C}$ and $i \neq p_j$, set $\mathsf{com}_{j,i} \leftarrow \mathsf{Com}(\mathsf{state}_{j,i}; r_{j,i})$ and $h_j \leftarrow \mathsf{H}(\mathsf{com}_{j,1} \parallel \cdots \parallel \mathsf{com}_{j,n})$.
2) For $j \in [M] \backslash \mathcal{C}$, set h_j as P does, and $h \leftarrow \mathsf{H}(h_1 \parallel \cdots \parallel h_M)$.

3) For $j \in \mathcal{C}$, generate $(\mathsf{msgs}_{j,i})_{i\in[n]}$ and $(\hat{z}_{j,\alpha})_{\alpha\in[|C|]}$, compute the output bit b of C and check $b \stackrel{?}{=} 1$. set $h'_j \leftarrow \mathsf{H}((\hat{z}_{j,\alpha})_{\alpha\in[|C|]} \parallel \mathsf{msgs}_{j,1} \parallel \ldots \parallel \mathsf{msgs}_{j,n} \parallel r_j)$ and $h' \leftarrow \mathsf{H}(h'_1 \parallel \cdots \parallel h'_M)$.
4) Check $\mathsf{H}(h, h') \stackrel{?}{=} h^*$.

We can apply the Fiat-Shamir transformation [22] to the above 3-round protocol to obtain a non-interactive zero-knowledge proof of knowledge in the random oracle model. We denote the proof system by NIZKPoK, and use it in the later sections. NIZKPoK should satisfy the following three requirements. (NIZKPoK1: Completeness), (NIZKPoK2: Knowledge-Soundness) and (NIZKPoK3: Computational Zero-Knowledge). For the details, see Section 2 of [16].

F. Predicates over Attributes

To treat an attribute-based primitive, the following notations for a function $\mathcal{R}^\kappa$ that describes a relation between two kinds of attributes are introduced.

- κ: A vector in $\mathbb{N}^c$ for a constant $c \in \mathbb{N}$, which is an index that indicates a structure of a predicate.
- $\mathbb{X}^\kappa$: The set of openers' attributes.
- $\mathbb{Y}^\kappa$: The set of signature attributes.
- $\mathcal{R}^\kappa : \mathbb{X}^\kappa \times \mathbb{Y}^\kappa \to \{0,1\}$: A predicate that determines a relation between $\mathbb{X}^\kappa$ and $\mathbb{Y}^\kappa$ as $\{(X,Y) \in \mathbb{X}^\kappa \times \mathbb{Y}^\kappa \mid \mathcal{R}^\kappa(X,Y) = 1\}$.

III. Group Signatures with Designated Traceability over Openers' Attributes - Static Case -

In this section, we define syntax and security of our scheme GSdT of group signatures with designated traceability over openers' attributes. In the previous work [7], the proposed syntax of GSdT is partially dynamic group [11]; that is, a user can join the group and a candidate opener can get an opening key based on its attributes. In contrast, our syntax is static [2]; that is, the keys of group members are generated and fixed, and also, the keys of openers are generated and fixed, in the group-key generation phase. Then security definitions are described in the static case.

A. Syntax

Our GSdT consists of four PPT algorithms $(\mathsf{GKG}, \mathsf{GSign}, \mathsf{GVrfy}, \mathsf{Open})$.

- $\mathsf{GKG}(1^\lambda, 1^l, 1^L, \kappa) \to (pp, gpk, \boldsymbol{gsk}, \boldsymbol{ok})$. This PPT algorithm of group-key generation is executed by the group manager (abbreviated as "GM"). GKG takes as input a string 1^λ that indicates the security parameter λ, a string 1^l that indicates the number of group members l, a string 1^L that indicates the number of openers L and an index κ that indicates a structure of a predicate. Here l and L are assumed to be upper bounded by a polynomial in λ. Also, κ is assumed to include a number $K \in \mathbb{N}$ that determines the attribute universe $\mathcal{U} = [K]$ (a small universe). Moreover, for every $\mathsf{o} \in [L]$, κ is assumed to include data that determine the opener's attribute X_o. GKG returns a set of public parameters *pp*, a group public key *gpk*, a vector of group member private secret keys $\boldsymbol{gsk}$ (with l entries) and a vector of openers' opening keys $\boldsymbol{ok}$ (with L entries). Here, for each $\mathsf{o} \in [L]$, $\boldsymbol{ok}[\mathsf{o}]$ is assumed to include the data of the opener's attribute X_o.
- $\mathsf{GSign}(gpk, \boldsymbol{gsk}[\mathsf{i}], Y, m) \to (Y, \sigma_0)$. This PPT algorithm of group-signature generation is executed by a group member. GSign takes as input *gpk*, $\boldsymbol{gsk}[\mathsf{i}]$, a signature attribute Y and a message m. GSign returns a group signature (Y, σ_0).
- $\mathsf{GVrfy}(gpk, m, (Y, \sigma_0)) \to 1/0$. This deterministic polynomial-time algorithm of group-signature verification is executed by an arbitrary verifier of a group signature. GVrfy takes as input *gpk*, m and (Y, σ_0). GVrfy returns a boolean decision 0 or 1.
- $\mathsf{Open}(gpk, \boldsymbol{ok}[\mathsf{o}], m, (Y, \sigma_0)) \to \mathsf{i}'$. This deterministic polynomial-time algorithm of group-signature opening is executed by an opener of a group signature. Open takes as input *gpk*, $\boldsymbol{ok}[\mathsf{o}]$, m and (Y, σ_0). Open returns a group member ID, i'.

B. Security Definitions

For a scheme of GSdT in the static case, we introduce three security notions according to the previous work of Bellare et al. [2]; correctness, anonymity and traceability.

Firstly, we define oracles that are needed to define the security notions, by Fig.1. GSignO is a group-signing oracle which returns the output of GSign. OpenO is an opening oracle which returns the output of Open. CrptOO is a corrupting-opener oracle which returns $\boldsymbol{ok}[\mathsf{o}]$. CrptUO is a corrupting-user oracle which returns $\boldsymbol{gsk}[\mathsf{i}]$. ChaO_b is a challenge oracle about a bit b which returns a challenge signature (Y, σ_0). We note that ChaO_b responds to only the first query. Besides, in the following, MS is the set of $(m, (Y, \sigma_0))$ which is queried to OpenO. CO is the set of openers' IDs ('o's) which are corrupted by an adversary. CU is the set of group members' IDs ('i's) which are corrupted by the adversary.

GSdT1: Correctness In the following experimental algorithm $\mathsf{Expr}^{\mathrm{corr}}_{\mathsf{GSdT},\mathbf{A}}$, $\mathbf{A}$ is an algorithm.

$\mathsf{Expr}^{\mathrm{corr}}_{\mathsf{GSdT},\mathbf{A}}(1^\lambda, 1^l, 1^L, \kappa)$
$(pp, gpk, \boldsymbol{gsk}, \boldsymbol{ok}) \leftarrow \mathsf{GKG}(1^\lambda, 1^l, 1^L, \kappa)$
$(\mathsf{i}, m, Y) \leftarrow \mathbf{A}(gpk)$
If $\mathsf{i} \notin [l]$ then return 0
$(Y, \sigma_0) \leftarrow \mathsf{GSign}(gpk, \boldsymbol{gsk}[\mathsf{i}], Y, m)$
$\mathrm{OS}_Y \leftarrow \{\mathsf{o} \in [L] \mid \mathcal{R}^\kappa(X_\mathsf{o}, Y) = 1 \text{ for } (X_\mathsf{o}, \bar{ok}) \leftarrow \boldsymbol{ok}[\mathsf{o}]\}$
If $\mathrm{OS}_Y \neq \emptyset$ and $\mathsf{GVrfy}(gpk, m, (Y, \sigma_0)) = 0$ then return 1
For $\mathsf{o} \in \mathrm{OS}_Y$ do
  $\mathsf{i}' \leftarrow \mathsf{Open}(gpk, \boldsymbol{ok}[\mathsf{o}], m, (Y, \sigma_0))$
  If $\mathsf{i} \neq \mathsf{i}'$ then return 1
Return 0

Then the advantage of the adversary algorithm $\mathbf{A}$ is defined as

$$\mathbf{Adv}^{\mathrm{corr}}_{\mathsf{GSdT},\mathbf{A}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{Expr}^{\mathrm{corr}}_{\mathsf{GSdT},\mathbf{A}}(1^\lambda, 1^l, 1^L, \kappa) = 1].$$

$\mathsf{GSignO}(\mathsf{i}, Y, m)$
If $\mathsf{i} \notin [l]$ then return $\perp$
Else return $\mathsf{GSign}(gpk, \boldsymbol{gsk}[\mathsf{i}], Y, m)$
$\mathsf{OpenO}(\mathsf{o}, m, (Y, \sigma_0))$
If $(m, (Y, \sigma_0)) \in$ MS then return $\perp$
Return $\mathsf{Open}(gpk, \boldsymbol{ok}[\mathsf{o}], m, (Y, \sigma_0))$
$\mathsf{CrptOO}(\mathsf{o})$
If $\mathsf{o} \notin [L]$ then return ε
$(X_{\mathsf{o}}, \bar{ok}) \leftarrow \boldsymbol{ok}[\mathsf{o}]$
If $\exists (m, (Y, \sigma_0)) \in$ MS s.t. $\mathcal{R}^{\kappa}(X, Y) = 1$ then return ε
CO $\leftarrow$ CO $\cup \{\mathsf{o}\}$
Return $\boldsymbol{ok}[\mathsf{o}]$
$\mathsf{CrptUO}(\mathsf{i})$
If $\mathsf{i} \in$ CU then return ε
CU $\leftarrow$ CU $\cup \{\mathsf{i}\}$
Return $\boldsymbol{gsk}[\mathsf{i}]$
$\mathsf{ChaO}_b(\mathsf{i}_0, \mathsf{i}_1, m, Y)$
If $\mathsf{i}_0 \notin [l]$ or $\mathsf{i}_1 \notin [l]$ then return $\perp$
If $\boldsymbol{gsk}[\mathsf{i}_0] = \varepsilon$ or $\boldsymbol{gsk}[\mathsf{i}_1] = \varepsilon$ then return $\perp$
If $\exists \mathsf{o} \in$ CO s.t. $\mathcal{R}^{\kappa}(X, Y) = 1$ for $(X_{\mathsf{o}}, \bar{ok}) \leftarrow \boldsymbol{ok}[\mathsf{o}]$ then return $\perp$
$(Y, \sigma_0) \leftarrow \mathsf{GSign}(gpk, \boldsymbol{gsk}[\mathsf{i}_b], Y, m)$
MS $\leftarrow$ MS $\cup \{(m, (Y, \sigma_0))\}$
Return (Y, σ_0)

Fig. 1. Definition of oracles.

Definition 1: If for any unbounded $\mathbf{A}$ $\mathbf{Adv}^{\mathrm{corr}}_{\mathsf{GSdT},\mathbf{A}}(\lambda) = 0$, then GSdT is said to be correct.

GSdT2: Full anonymity In the following experimental algorithm $\mathsf{Expr}^{\text{anon-}b}_{\mathsf{GSdT},\mathbf{A}}$, $\mathbf{A}$ is an algorithm.

$\mathsf{Expr}^{\text{anon-}b}_{\mathsf{GSdT},\mathbf{A}}(1^{\lambda}, 1^{l}, 1^{L}, \kappa)$ // $b \in \{0, 1\}$
$(pp, gpk, \boldsymbol{gsk}, \boldsymbol{ok}) \leftarrow \mathsf{GKG}(1^{\lambda}, 1^{l}, 1^{L}, \kappa)$
MS $\leftarrow \emptyset$, CO $\leftarrow \emptyset$
$d \leftarrow \mathbf{A}(gpk, \boldsymbol{gsk})^{\mathsf{OpenO},\mathsf{CrptOO},\mathsf{ChaO}_b}$
Return d

In the above experiment, after a query to the challenge oracle ChaO_b, it is prohibited for $\mathbf{A}$ to issue queries to the opening oracle OpenO and the corrupting-opener oracle CrptOO that $\mathbf{A}$ wins trivially. Then the advantage of the adversary algorithm $\mathbf{A}$ is defined as

$$\mathbf{Adv}^{\mathrm{anon}}_{\mathsf{GSdT},\mathbf{A}}(\lambda) \stackrel{\mathrm{def}}{=} |\Pr[\mathsf{Expr}^{\text{anon-0}}_{\mathsf{GSdT},\mathbf{A}}(1^{\lambda}, 1^{l}, 1^{L}, \kappa) = 1] - \Pr[\mathsf{Expr}^{\text{anon-1}}_{\mathsf{GSdT},\mathbf{A}}(1^{\lambda}, 1^{l}, 1^{L}, \kappa) = 1]|.$$

Definition 2: If for any PPT $\mathbf{A}$ $\mathbf{Adv}^{\mathrm{anon}}_{\mathsf{GSdT},\mathbf{A}}(\lambda)$ is negligible in λ, then GSdT is said to be fully anonymous.

However, our scheme of GSdT which will be described in Section IV does not have the above full anonymity. For this reason, we introduce the following "weak anonymity".

GSdT2': Weak anonymity In the following experimental algorithm $\mathsf{Expr}^{\text{w-anon-}b}_{\mathsf{GSdT},\mathbf{A}}$, $\mathbf{A}$ is an algorithm.

$\mathsf{Expr}^{\text{w-anon-}b}_{\mathsf{GSdT},\mathbf{A}}(1^{\lambda}, 1^{l}, 1^{L}, \kappa)$ // $b \in \{0, 1\}$
$(pp, gpk, \boldsymbol{gsk}, \boldsymbol{ok}) \leftarrow \mathsf{GKG}(1^{\lambda}, 1^{l}, 1^{L}, \kappa)$
MS $\leftarrow \emptyset$, CO $\leftarrow \emptyset$, CU $\leftarrow \emptyset$
$d \leftarrow \mathbf{A}(gpk)^{\mathsf{GSignO},\mathsf{OpenO},\mathsf{CrptUO},\mathsf{CrptOO},\mathsf{ChaO}_b}$
If $\mathsf{i}_0 \in$ CU or $\mathsf{i}_1 \in$ CU then return $\perp$
Return d

In the above experiment, after a query to the challenge oracle ChaO_b, it is prohibited for $\mathbf{A}$ to issue queries to the opening oracle OpenO and the corrupting-opener oracle CrptOO that $\mathbf{A}$ wins trivially.

In the definition of the full anonymity, the adversary $\mathbf{A}$ is given all the private secret keys $\boldsymbol{gsk}$. In contrast, in the definition of the weak anonymity, $\mathbf{A}$ is not given the private secret keys $\boldsymbol{gsk}[\mathsf{i}_0]$ and $\boldsymbol{gsk}[\mathsf{i}_1]$ where i_0 and i_1 are group members' IDs queried to the challenge oracle ChaO. That is, $\mathbf{A}$ is not given $\boldsymbol{gsk}$, and moreover, the queries to the corrupting-group-member oracle CrptUO about i_0 and i_1 is prohibited.

Then the advantage of the adversary algorithm $\mathbf{A}$ is defined as

$$\mathbf{Adv}^{\text{w-anon}}_{\mathsf{GSdT},\mathbf{A}}(\lambda) \stackrel{\mathrm{def}}{=} |\Pr[\mathsf{Expr}^{\text{w-anon-0}}_{\mathsf{GSdT},\mathbf{A}}(1^{\lambda}, 1^{l}, 1^{L}, \kappa) = 1] - \Pr[\mathsf{Expr}^{\text{w-anon-1}}_{\mathsf{GSdT},\mathbf{A}}(1^{\lambda}, 1^{l}, 1^{L}, \kappa) = 1]|.$$

Definition 3: If for any PPT $\mathbf{A}$ $\mathbf{Adv}^{\text{w-anon}}_{\mathsf{GSdT},\mathbf{A}}(\lambda)$ is negligible in λ, then GSdT is said to be weakly anonymous.

GSdT3: Traceability In the following experimental algorithm $\mathsf{Expr}^{\mathrm{trace}}_{\mathsf{GSdT},\mathbf{A}}$, $\mathbf{A}$ is an algorithm.

$\mathsf{Expr}^{\mathrm{trace}}_{\mathsf{GSdT},\mathbf{A}}(1^{\lambda}, 1^{l}, 1^{L}, \kappa)$
$(pp, gpk, \boldsymbol{gsk}, \boldsymbol{ok}) \leftarrow \mathsf{GKG}(1^{\lambda}, 1^{l}, 1^{L}, \kappa)$
CU $\leftarrow \emptyset$
$(m, (Y, \sigma_0)) \leftarrow \mathbf{A}(gpk, \boldsymbol{ok})^{\mathsf{GSignO},\mathsf{CrptUO}}$
$\mathrm{OS}_Y \leftarrow \{\mathsf{o} \in [L] \mid \mathcal{R}^{\kappa}(X_{\mathsf{o}}, Y) = 1$ for $(X_{\mathsf{o}}, \bar{ok}) \leftarrow \boldsymbol{ok}[\mathsf{o}]\}$
If $\mathrm{OS}_Y = \emptyset$ or $\mathsf{GVrfy}(gpk, m, (Y, \sigma_0)) = 0$ then return 0
Find $\mathsf{o} \in [L]$ s.t. $\mathcal{R}^{\kappa}(X_{\mathsf{o}}, Y) = 1$ for $(X_{\mathsf{o}}, \bar{ok}) \leftarrow \boldsymbol{ok}[\mathsf{o}]$
$\mathsf{i} \leftarrow \mathsf{Open}(gpk, \boldsymbol{ok}[\mathsf{o}], m, (Y, \sigma_0))$
If $\mathsf{i} \in$ CU then return 0
If (i, Y, m) was queried to GSignO then return 0
Return 1

Then the advantage of the adversary algorithm $\mathbf{A}$ is defined as

$$\mathbf{Adv}^{\mathrm{trace}}_{\mathsf{GSdT},\mathbf{A}}(\lambda) \stackrel{\mathrm{def}}{=} \Pr[\mathsf{Expr}^{\mathrm{trace}}_{\mathsf{GSdT},\mathbf{A}}(1^{\lambda}, 1^{l}, 1^{L}, \kappa) = 1].$$

Definition 4: If for any PPT $\mathbf{A}$ $\mathbf{Adv}^{\mathrm{trace}}_{\mathsf{GSdT},\mathbf{A}}(\lambda)$ is negligible in λ, then GSdT is said to be traceable.

IV. Our Construction and Security

In this section, we describe our construction of GSdT from symmetric-key primitives. Then we state its security properties.

A. Construction

We construct the four PPT algorithms GKG, GSign, GVrfy and Open from pseudorandom function (PRFs), hash functions (Hs) and commitments (Cmts). Note here that the non-interactive zero-knowledge proof of knowledge NIZKPoK of KKW [16] is constructed from PRF, H and Cmt. In the following, a signature attribute $Y \in \mathbb{Y}^\kappa$ is an all-AND boolean formula, while an opener's attribute $X \in \mathbb{X}^\kappa$ is a subset of the attribute universe $\mathcal{U} = [K]$. We denote by $[Y]$ the set of attributes that appear in Y.

• $\mathsf{GKG}(1^\lambda, 1^l, 1^L, \kappa) \to (pp, gpk, \boldsymbol{gsk}, \boldsymbol{ok})$. Our GKG takes as input 1^λ, 1^l, 1^L and κ. GKG computes the values that determines the circuits of $\mathsf{PRF}_\cdot(\cdot)$, $\mathsf{H}(\cdot)$ and NIZKPoK. It also computes the values of public parameters of the Merkle tree. It sets the data of the public parameters as *pp*. Then for each group member $\mathsf{i} \in [l]$ and for every attribute $\mathsf{j} \in [K]$, it generates two strings $k^0_{\mathsf{i},\mathsf{j}}, k^1_{\mathsf{i},\mathsf{j}} \in_R \{0,1\}^\lambda$ uniformly and independently at random. It sets a private secret key for i as

$$\boldsymbol{gsk}[\mathsf{i}] := (k^0_{\mathsf{i},\mathsf{j}}, k^1_{\mathsf{i},\mathsf{j}})_{\mathsf{j}\in[K]}, \quad \mathsf{i} = 1, \ldots, l. \tag{1}$$

Then, for each opener $\mathsf{o} \in [L]$, it sets an opening key as

$$\boldsymbol{ok}[\mathsf{o}] := \left(X_\mathsf{o}, (k^0_{\mathsf{i},\mathsf{j}})_{\mathsf{j}\in X_\mathsf{o}}\right)_{\mathsf{i}\in[l]}, \quad \mathsf{o} = 1, \ldots, L. \tag{2}$$

Then, for each $\mathsf{i} \in [l]$ and for every $\mathsf{j} \in [K]$, it computes values of the pseudorandom function as

$$y^0_{\mathsf{i},\mathsf{j}} \leftarrow \mathsf{PRF}_{k^0_{\mathsf{i},\mathsf{j}}}(0^\lambda), \ y^1_{\mathsf{i},\mathsf{j}} \leftarrow \mathsf{PRF}_{k^1_{\mathsf{i},\mathsf{j}}}(0^\lambda). \tag{3}$$

Using these values, it sets the group public key as

$$y_{\mathsf{i},\mathsf{j}} := (y^0_{\mathsf{i},\mathsf{j}}, y^1_{\mathsf{i},\mathsf{j}}), \quad \mathsf{i} = 1, \ldots, l, \ \mathsf{j} = 1, \ldots, K, \tag{4}$$

$$gpk := (y_{\mathsf{i},\mathsf{j}})_{\mathsf{i}\in[l],\mathsf{j}\in[K]}. \tag{5}$$

It returns $(pp, gpk, \boldsymbol{gsk}, \boldsymbol{ok})$.

GM maintains *pp* and *gpk*. On the other hand, GM distributes $\boldsymbol{gsk}[\mathsf{i}]$ to each group member i ($\mathsf{i} = 1, \ldots, l$), and $\boldsymbol{ok}[\mathsf{o}]$ to each opener o ($\mathsf{o} = 1, \ldots, L$).

• $\mathsf{GSign}(gpk, \boldsymbol{gsk}[\mathsf{i}], Y, m) \to (Y, \sigma_0)$. Our GSign takes as input gpk, $\boldsymbol{gsk}[i]$, Y and m. First, for each $\mathsf{i} \in [l]$, it computes for each $\mathsf{j} \in [Y]$ the hash value of $y_{\mathsf{i},\mathsf{j}}$ as

$$h_{\mathsf{i},\mathsf{j}} \leftarrow \mathsf{H}(y_{\mathsf{i},\mathsf{j}}), \quad \mathsf{j} \in [Y], \ \mathsf{i} \in [l]. \tag{6}$$

Then it generates a Merkle tree T from the data $(h_{\mathsf{i},\mathsf{j}})_{\mathsf{j}\in[Y],\mathsf{i}\in[l]}$, as follows (see Fig. 2 for an instance case $(l, |[Y]|) = (2, 2)$).

1) For each $\mathsf{i} \in [l]$: Construct a binary tree from the data $(h_{\mathsf{i},\mathsf{j}})_{\mathsf{j}\in[Y]}$, and set the root hash as h^*_i.
2) Construct a binary tree from the data $(h^*_\mathsf{i})_{\mathsf{i}\in[l]}$, and set the root hash as h^*.
3) Concatenate the root nodes of 1) with leaf nodes of 2) by identifying the nodes with the same label h^*_i, and set the resulting tree as T.

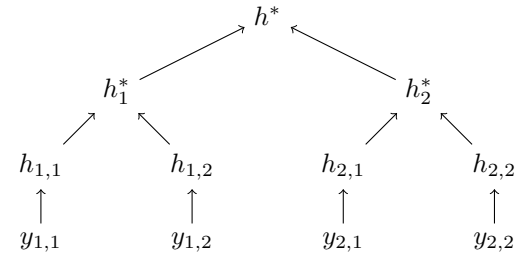

Fig. 2. Merkle tree T of our GSdT. Case of $(l, |[Y]|) = (2, 2)$.

Second, by using the index '0' part of the private secret key $\boldsymbol{gsk}[\mathsf{i}] = (k^0_{\mathsf{i},\mathsf{j}}, k^1_{\mathsf{i},\mathsf{j}})_{\mathsf{j}\in[K]}$ (i.e. the former component), GSign computes the following hash value.

$$\begin{aligned} &\text{Parsing } [Y] \text{ as } (\mathsf{j}_1, \ldots, \mathsf{j}_k), \ \mathsf{j}_1 < \cdots < \mathsf{j}_k, \\ &h' \leftarrow \overline{\mathsf{H}}(k^0_{\mathsf{i},\mathsf{j}_1}, \ldots, k^0_{\mathsf{i},\mathsf{j}_k}), \end{aligned} \tag{7}$$

where $\overline{\mathsf{H}}$ means applying the hash function H successively (for neighboring two entries) to form a binary-tree structure extending the Merkle tree.

Then, using h' as the key, GSign computes the hash value x of m and the pseudorandom function value of x.

$$x \leftarrow \mathsf{H}(m), \tag{8}$$

$$y \leftarrow \mathsf{PRF}_{h'}(x). \tag{9}$$

Third, GSign generates a boolean circuit $C_{x,y,T}$ so that it satisfies the following conditions. $C_{x,y,T}$ should be hardcoded with the values x, y and $(h_{\mathsf{i},\mathsf{j}})_{\mathsf{j}\in[Y],\mathsf{i}\in[l]}$, where each $h_{\mathsf{i},\mathsf{j}}$ is the value labeled at each leaf node of T. $C_{x,y,T}$ should take as input $(k^0_{\mathsf{i},\mathsf{j}}, k^1_{\mathsf{i},\mathsf{j}})_{\mathsf{j}\in[Y]}$, i and a Merkle path Path. Here, when the tree T is truncated just under the node N' which has the label h^*_i(so that N' turns into a leaf node), Path is the Merkle path at the node N' to reconstruct h^*. $C_{x,y,T}$ should be a circuit to compute $y^0_{\mathsf{i},\mathsf{j}} \leftarrow \mathsf{PRF}_{k^0_{\mathsf{i},\mathsf{j}}}(0^\lambda)$ and $y^1_{\mathsf{i},\mathsf{j}} \leftarrow \mathsf{PRF}_{k^1_{\mathsf{i},\mathsf{j}}}(0^\lambda)$ for every $\mathsf{j} \in [Y]$. $C_{x,y,T}$ should output a boolean 1 if and only if the following three conditions hold.

1) $y = \mathsf{PRF}_{h'}(x)$.
2) Path is certainly the Merkle path at the node N' with respect to the truncated tree of T.
3) The remaining subtree (generated by truncating T just above N') is the Merkle tree to reconstruct the root hash h^*_i from the values $(k^0_{\mathsf{i},\mathsf{j}}, k^1_{\mathsf{i},\mathsf{j}})_{\mathsf{j}\in[Y]}$ through PRF and H.

Fourth, GSign generates a proof π of knowledge of a witness $w := ((k^0_{\mathsf{i},\mathsf{j}}, k^1_{\mathsf{i},\mathsf{j}})_{\mathsf{j}\in[Y]}, \mathsf{i}, \mathsf{Path})$ that makes the circuit $C_{x,y,T}$ output 1 by executing the prover algorithm $\mathsf{P}(C_{x,y,T}, w)$ of the KKW NIZKPoK.

Finally, setting $\sigma_0 := (y, \pi)$, GSign returns a group signature (Y, σ_0).

• $\mathsf{GVrfy}(gpk, m, (Y, \sigma_0)) \rightarrow 1/0$. Our GVrfy takes as input gpk, m and (Y, σ_0). It computes the hash value $x \leftarrow \mathsf{H}(m)$ of the message m, and parses $\sigma_0 = (y, \pi)$ to obtain y, and generates the Merkle tree T from the data $h_{\mathsf{i},\mathsf{j}} \leftarrow \mathsf{H}(y_{\mathsf{i},\mathsf{j}})$, $\mathsf{j} \in [Y]$, $\mathsf{i} \in [l]$. Then it executes the verifier algorithm $\mathsf{V}(C_{x,y,T}, \pi)$ of the KKW $\mathsf{NIZKPoK}$, and obtains a boolean decision b. GVrfy returns b.

• $\mathsf{Open}(gpk, \boldsymbol{ok}[\mathsf{o}], m, (Y, \sigma_0)) \rightarrow \mathsf{i}'$. Our Open takes as input gpk, $\boldsymbol{ok}[\mathsf{o}]$, m and (Y, σ_0). It firstly checks whether $\mathcal{R}(X_{\mathsf{o}}, Y) = 1$ or not. Note that $\mathcal{R}(X_{\mathsf{o}}, Y) = 1$ holds if and only if $X_{\mathsf{o}} (\subset [K])$ satisfies that $X_{\mathsf{o}} \supset [Y]$. If $\mathcal{R}$ returns 0, then Open returns $\perp$. Else if $\mathcal{R}$ returns 1, then Open computes $x \leftarrow \mathsf{H}(m)$, and parses $\sigma_0 = (y, \pi)$ to obtain y. Then it checks whether $\mathsf{GVrfy}(gpk, m, (Y, \sigma_0)) = 1$ holds or not. If GVrfy returns 0, then Open sets $\mathsf{i}' := \perp$, and returns i'. Else if GVrfy returns 1, then, by using $\boldsymbol{ok}[\mathsf{o}] = \left((k^0_{\mathsf{i},\mathsf{j}})_{\mathsf{j} \in X_{\mathsf{o}}}\right)_{\mathsf{i} \in [l]}$, Open does the following exhaustive search. (Note that $[Y] = \{\mathsf{j}_1, \ldots, \mathsf{j}_k\}$.)

For $\mathsf{i} \in [l]$:
$h' \leftarrow \overline{\mathsf{H}}(k^0_{\mathsf{i},\mathsf{j}_1}, \ldots, k^0_{\mathsf{i},\mathsf{j}_k})$
If $y = \mathsf{PRF}_{h'}(x)$ then return $\mathsf{i}' := \mathsf{i}$
Return $\mathsf{i}' := \perp$

Open returns i'.

B. Security

In the following, (PRF), (HF1), (HF2), (Com1), (Com2) and (Com3) denote the properties of PRF, H and Cmt summarized in Section II.

Theorem 1 (Correctness): If PRF is a pseudorandom function, H is a hash function and Cmt is correct (Com1), then our GSdT is correct (GSdT1).

Proof(*sketch*). In the experiment $\mathsf{Expr}^{\mathrm{trace}}_{\mathsf{GSdT},\mathbf{A}}(1^\lambda, 1^l, 1^L, \kappa)$, $\mathsf{GVrfy}(gpk, m, (Y, \sigma_0))$ returns 1 when $\mathrm{OS}_Y \neq \emptyset$ due to the construction of our GSdT. Further, i' returned by $\mathsf{Open}(gpk, \boldsymbol{ok}[\mathsf{o}], m, (Y, \sigma_0))$ satisfies $\mathsf{i}' = \mathsf{i}$. These holds for any unbounded $\mathbf{A}$. □

Theorem 2 (Weak anonymity): If PRF is pseudorandom (PRF), H is preimage resistant (HF1) and collision resistant (HF2), Cmt is correct (Com1) and computationally hiding (Com2), then our GSdT is weakly anonymous (GSdT2') in the random oracle model.

Proof(*sketch*). The property holds due to the computational zero-knowledge of $\mathsf{NIZKPoK}$ and our construction of the Merkle tree T. More precisely, for any given PPT adversary $\mathbf{A}$ that is in accordance with the experiment $\mathsf{Expr}^{\text{w-anon-}b}_{\mathsf{GSdT},\mathbf{A}}(1^\lambda, 1^l, 1^L, \kappa)$, we construct a PPT distinguisher $\mathbf{D}$ on $\mathsf{NIZKPoK}$, assuming all the hash functions are the random oracles. The details are omitted. □

Theorem 3 (Traceability): If PRF is pseudorandom (PRF), H is preimage resistant (HF1) and collision resistant (HF2), Cmt is correct (Com1) and computationally binding (Com3), then our GSdT is traceable (GSdT3) in the random oracle model.

Proof(*sketch*). The proof goes in the same way as in the traceability proof of the group signature scheme proposed by KKW [16], except treatment of the predicate $\mathcal{R}(X, Y)$ and the related keys $\boldsymbol{gsk}$ and $\boldsymbol{ok}$ because our construction of the Merkle tree T is similar to that of the Merkle tree of KKW [16]. More precisely, for any given PPT adversary $\mathbf{A}$ that is in accordance with the experiment $\mathsf{Expr}^{\mathrm{trace}}_{\mathsf{GSdT},\mathbf{A}}(1^\lambda, 1^l, 1^L, \kappa)$, we construct a polynomial-time reduction to the computationally binding (Com3) by using the PPT knowledge-extractor $\mathbf{B}$ on $\mathsf{NIZKPoK}$, assuming all the hash functions are the random oracles. The details are omitted. □

V. Performance Estimation

In this section, we estimate expected performance of our GSdT described in Section IV. The estimation is based on the implementation and estimation of KKW [16].

As a concrete instantiation that is expected to attain security against quantum computers, the block cipher LowMC [14] (as PRF) and the cryptographic hash function SHAKE [15] (as H and Cmt) are employed in KKW [13], [16]. LowMC is suitably designed for multi-party computation because it has less AND-gates and less AND-depth [14]. By employing LowMC in the design of PRF, the computational amount to generate a signature and the length of a generated signature of the group signature scheme GS that was proposed in KKW [16] are expected to be less than the ones which does not employ LowMC. The computational amount and the length are dominated by the circuit C that appears in the signing algorithm of GS. Here we note that the structure of wires and gates are determined by the Merkle tree T. Things are similar in the case of our GSdT because the construction of our Merkle tree has similar structure to that of GS [16]. Nonetheless, we pay attention that the number of leaf nodes of GS [16] is equal to the number of group members l, while the number of leaf nodes of our GSdT is equal to lk, where $k := |[Y]|$ is the number of attributes that appears in the all-AND boolean formula Y associated to the group signature. Thus, in short, when our GSdT is instantiated with the block cipher LowMC [14], its performance is estimated to be the one where the number l is substituted with lk.

Table I shows the estimation based on the substitution. We remark that the note (*1) in Table I shows citation from the results with 128 bit quantum security. Here $l = 2^7 = 128$ and $k = 2^3 = 8$, so $lk = 2^{10}$, and hence we read ' 2^{10} ' in Table 4 of KKW [16].

VI. Conclusion

In this paper, based on the previous work of Katz-Kolesnikov-Wang [16] (KKW), we have proposed a group signature scheme GSdT that is constructed from only symmetric-key primitives, in which a signer can designate openers with an all-AND formula over openers' attributes. Here the symmetric-key primitives are pseudorandom functions, hash functions and commitments. The proposed GSdT is expected to be secure against computational power of quantum computers. The performance of our GSdT is also estimated. The signing time is less than 3.0 sec and the signature size is less than 0.5 MB in the situation that the number of group members l is

TABLE I
COMPARISON OF EFFICIENCY AND SECURITY IN THE RANDOM ORACLE MODEL. (*1) MEANS $(l, k) = (2^7, 2^3)$.

Scheme	Len.σ	Comp.σ	Security Assump.	Quant. Resist.	Len.σ	Comp.σ(*1)
KKW [16] GS	$O(\log(l))$	$O(\log(l))$	(PRF),(HF1,2),(COM1-3)	Yes	285KB (*1)	2.0sec (*1)
Our GSdT	$O(\log(lk))$	$O(\log(lk))$	(PRF),(HF1,2),(COM1-3)	Yes	418KB (*1)	3.0sec (*1)

2^7, the number of designated attributes k is 2^3 and the target security is 128 bit quantum security.

In the presented construction, only all-AND formulas over attributes can be designated as access structures by a group member in the signing phase. In future work, the bound of designation should be generalized to any boolean formulas towards attaining fine grained access control of the opening function. Besides, since our GSdT is in the static group model, to give GSdT schemes from symmetric-key primitives in the models of the partially dynamic group [11] and the fully dynamic group [23] is a natural direction. Further, implementing our GSdT and optimizing the parameters such as (M, n, τ) of NIZKPoK would be useful for real usage. For the purpose, the "Picnic 3" post-quantum signature algorithm [13] should be analyzed.

ACKNOWLEDGMENTS

This work was supported by JSPS KAKENHI Grant Numbers JP23K11106, JP23K11105 and JP22K12023.

REFERENCES

[1] D. Chaum and E. van Heyst, "Group signatures," in *Advances in Cryptology - EUROCRYPT '91, Workshop on the Theory and Application of of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, Proceedings*, 1991, pp. 257–265. [Online]. Available: http://dx.doi.org/10.1007/3-540-46416-6_22

[2] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions," in *Advances in Cryptology - EUROCRYPT 2003, International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4-8, 2003, Proceedings*, 2003, pp. 614–629. [Online]. Available: http://dx.doi.org/10.1007/3-540-39200-9_38

[3] S. Xu and M. Yung, "Accountable ring signatures: A smart card approach," in *Smart Card Research and Advanced Applications VI, IFIP 18th World Computer Congress, TC8/WG8.8 & TC11/WG11.2 Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS), 22-27 August 2004, Toulouse, France*, ser. IFIP, J. Quisquater, P. Paradinas, Y. Deswarte, and A. A. E. Kalam, Eds., vol. 153. Kluwer/Springer, 2004, pp. 271–286. [Online]. Available: https://doi.org/10.1007/1-4020-8147-2_18

[4] Y. Sakai, K. Emura, G. Hanaoka, Y. Kawai, T. Matsuda, and K. Omote, "Group signatures with message-dependent opening," in *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, ser. Lecture Notes in Computer Science, M. Abdalla and T. Lange, Eds., vol. 7708. Springer, 2012, pp. 270–294. [Online]. Available: https://doi.org/10.1007/978-3-642-36334-4_18

[5] B. Libert, K. Nguyen, T. Peters, and M. Yung, "Bifurcated signatures: Folding the accountability vs. anonymity dilemma into a single private signing scheme," in *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part III*, ser. Lecture Notes in Computer Science, A. Canteaut and F. Standaert, Eds., vol. 12698. Springer, 2021, pp. 521–552. [Online]. Available: https://doi.org/10.1007/978-3-030-77883-5_18

[6] K. Nguyen, F. Guo, W. Susilo, and G. Yang, "Multimodal private signatures," in *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part II*, ser. Lecture Notes in Computer Science, Y. Dodis and T. Shrimpton, Eds., vol. 13508. Springer, 2022, pp. 792–822. [Online]. Available: https://doi.org/10.1007/978-3-031-15979-4_27

[7] H. Anada, M. Fukumitsu, and S. Hasegawa, "Group signatures with designated traceability," in *Proc. Ninth International Symposium on Computing and Networking, CANDAR 2021, Matsue, Japan, November 23-26, 2021*, 2021.

[8] ——, "Group signatures with designated traceability over openers' attributes," *International Journal of Networking and Computing*, vol. 12, no. 2, pp. 493–508, July 2022. [Online]. Available: http://ijnc.org/index.php/ijnc/article/view/294

[9] ——, "Group signatures with designated traceability over openers' attributes in bilinear groups," in *The 23rd World Conference on Information Security Applications (WISA 2022)*, I. You and T.-Y. Youn, Eds. Cham: Springer Nature Switzerland, August 2022, pp. 29–43.

[10] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*. IEEE Computer Society, 1994, pp. 124–134. [Online]. Available: https://doi.org/10.1109/SFCS.1994.365700

[11] M. Bellare, H. Shi, and C. Zhang, "Foundations of group signatures: The case of dynamic groups," in *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, ser. Lecture Notes in Computer Science, A. Menezes, Ed., vol. 3376. Springer, 2005, pp. 136–153. [Online]. Available: https://doi.org/10.1007/978-3-540-30574-3_11

[12] C. S. R. CENTER, "Post-quantum cryptography PQC, round 3 submissions," 4 2024.

[13] G. Zaverucha, M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, J. Katz, X. Wang, V. Kolesnikov, and D. Kales, "Picnic a family of post-quantum secure digital signature algorithms," accessed: 2024-04-26.

[14] M. R. Albrecht, C. Rechberger, T. Schneider, T. Tiessen, and M. Zohner, "Ciphers for MPC and FHE," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, ser. Lecture Notes in Computer Science, E. Oswald and M. Fischlin, Eds., vol. 9056. Springer, 2015, pp. 430–454. [Online]. Available: https://doi.org/10.1007/978-3-662-46800-5_17

[15] Internet Engineering Task Force, "Use of the SHAKE one-way hash functions in the cryptographic message syntax (cms)," https://www.rfc-editor.org/rfc/rfc8702.html, January 2020.

[16] J. Katz, V. Kolesnikov, and X. Wang, "Improved non-interactive zero knowledge with applications to post-quantum signatures," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, D. Lie, M. Mannan, M. Backes, and X. Wang, Eds. ACM, 2018, pp. 525–537. [Online]. Available: https://doi.org/10.1145/3243734.3243805

[17] H. Anada, M. Fukumitsu, and S. Hasegawa, "Accountable ring signatures from symmetric-key primitives," in *2023 Eleventh International Symposium on Computing and Networking (CANDAR)*. Los Alamitos, CA, USA: IEEE Computer Society, dec 2023, pp. 86–92. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CANDAR60563.2023.00018

[18] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology - CRYPTO '87, A Conference on the Theory and Applications of Cryptographic*

Techniques, Santa Barbara, California, USA, August 16-20, 1987, Proceedings, ser. Lecture Notes in Computer Science, C. Pomerance, Ed., vol. 293. Springer, 1987, pp. 369–378. [Online]. Available: https://doi.org/10.1007/3-540-48184-2_32

[19] O. Goldreich, *The Foundations of Cryptography - Volume 1, Basic Techniques*. Cambridge University Press, 2001.

[20] P. Rogaway and T. Shrimpton, "Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance," in *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, 2004, pp. 371–388.

[21] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-knowledge from secure multiparty computation," in *Proceedings of the 39th Annual ACM Symposium on Theory of Computing, San Diego, California, USA, June 11-13, 2007*, 2007, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/1250790.1250794

[22] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, 1986, pp. 186–194. [Online]. Available: http://dx.doi.org/10.1007/3-540-47721-7_12

[23] J. Bootle, A. Cerulli, P. Chaidos, E. Ghadafi, and J. Groth, "Foundations of fully dynamic group signatures," in *Applied Cryptography and Network Security - 14th International Conference, ACNS 2016, Guildford, UK, June 19-22, 2016. Proceedings*, ser. Lecture Notes in Computer Science, M. Manulis, A. Sadeghi, and S. A. Schneider, Eds., vol. 9696. Springer, 2016, pp. 117–136. [Online]. Available: https://doi.org/10.1007/978-3-319-39555-5_7