Homomorphic Trapdoors for Identity-based and Group Signatures

Buvana Ganesh*, Apurva Vangujar, Alia Umrani and Paolo Palmieri

School of Computer Science & IT,
University College Cork, Ireland
b.ganesh@cs.ucc.ie, a.vangujar@cs.ucc.ie,
a.umrani@cs.ucc.ie, p.palmieri@cs.ucc.ie

Abstract. Group signature (GS) schemes are an important primitive in cryptography that provides anonymity and traceability for a group of users. In this paper, we propose a new approach to constructing GS schemes using the homomorphic trapdoor function (HTDF). We focus on constructing an identity-based homomorphic signature (IBHS) scheme using the trapdoor, providing a simpler scheme that has no zero-knowledge proofs. Our scheme allows packing more data into the signatures by elevating the existing homomorphic trapdoor from the SIS assumption to the MSIS assumption to enable packing techniques. Compared to the existing group signature schemes, we provide a straightforward and alternate construction that is efficient and secure under the standard model. Overall, our proposed scheme provides an efficient and secure solution for GS schemes using HTDF.

1 Introduction

Lattice-based cryptography is a promising alternative to conventional cryptography since attacks on it require exponential time for quantum adversaries. Homomorphic encryption is a popular tool that is highly desirable in privacy-sensitive applications such as secure data processing, cloud computing, and secure machine learning because it allows computations to be performed on encrypted data, preserving privacy.

Similarly, homomorphic signatures (HS) guarantee the authenticity of outsourced computation results and are useful for analyzing large amounts of sensitive data held by organizations such as healthcare providers and financial institutions. For instance, after collecting, certifying, and distributing the data to multiple research teams for analysis, there is a possibility that some of these groups may act dishonestly and fabricate results for their advantage, rendering them untrustworthy. With HS, the data can be signed and distributed to multiple research groups for analysis, and the results can be posted publicly with corresponding signatures for anyone to verify without accessing the original data. With identity-based homomorphic signatures (IBHS), multiple signatures can be verified together, allowing for faster and more efficient verification helping create trustless systems .

Group signatures (GS) scheme enables group members to sign on behalf of their group while ensuring authenticity, anonymity, and traceability. The identity of the group

* Buvana Ganesh is supported by a PhD scholarship funded by the Science Foundation Ireland Centre under Grant No. 18/CRT/6223

member who issued the signature remains anonymous. However, if necessary, a trusted entity known as the group manager can use secret information to trace the signature back to the signer, thereby proving the accountability of the group members for their signatures. Due to various advantageous properties, GS schemes have found practical uses in several domains [18], including safeguarding privacy, anonymous online communication, e-commerce systems, and trusted hardware attestation. Specifically, group digital signatures (DS) have a wide range of applications across various industries including contract signing, board resolutions, shareholder meetings, medical records, consent forms, and financial transactions.

In this work, we explore the construction of lattice-based GS schemes primarily on the module variants of the Shortest Integer Solution (MSIS) assumption and the Learning with Errors (MLWE). We find alternate constructions for the GS scheme through the IBHS scheme.

1.1 Our contributions and techniques

We summarize the main contributions from our work and then expand on the techniques used to achieve them and why it was necessary.

- We adapt the homomorphic trapdoor from [16] to the MSIS assumption.
- We construct a novel IBHS scheme using the homomorphic trapdoor function that is secure under the MSIS assumption.
- Then, we create a GS scheme to provide a novel construction through IBHS, excluding the index and attributes in the previous schemes.
- We provide a security proof for both under the standard model and with Unforgeability under Chosen-Message Attacks (UF-CMA) along with the corresponding properties like anonymity and traceability along with security against different forgeries.

To construct a static GS, we require the properties of anonymity and traceability to be satisfied along with the standard unforgeability. Therefore, GS schemes are built on an unforgeable signature and a public key encryption scheme. The majority of GS schemes in the literature for lattices have been constructed following one of two approaches: an access structure and some form of Argument of Knowledge [22,21,29]; or by constructing an attribute based signature (ABS) scheme with different properties and building the GS scheme on top of that [18].

The property of homomorphism also helps in the batch verification of signatures from different parties evaluated together. The connection between signature schemes and identity-based encryption schemes is the trapdoor employed to sample the signature and the identity respectively. If we use the HTDFs as mentioned in Sec. 4.2, we retain the homomorphism in the signature (and the keys of the identities) but not in the messages themselves.

We mainly focus on the work of [18] where the original [16] scheme is perceived as an ABS scheme and construct the GS scheme on top of it. We take a different approach by constructing an IBHS scheme that is simpler in construction and requires lesser discrete Gaussian sampling. Instead of considering an index and the attribute, we consider

only the identity of the user, as that's enough to provide security for GS schemes. We use labeled programs to correlate the identities with the public parameters.

GS schemes constructed with ABS also require the presence of One-Time Signatures (OTS), constructed with Chameleon hashes. The HTDF come with a special property that makes them equivalent to Chameleon hashes because of the usage of trapdoors, inversion, random element, etc in the construction. This makes them the perfect substitute for OTS in our schemes to give the signature strong unforgeability.

Another drawback of previous schemes is that they essentially signed bits in parallel. To pack more data into the signatures, we elevate the existing homomorphic trapdoor from the SIS assumption to the MSIS assumption using the techniques provided in [4], which can be found in Lemma 1 of this paper. This way one can use the Chinese Remainder Theorem (CRT) packing or various other packing methods that allows Single Instruction Multiple Data (SIMD) techniques.

Combining these methods together, we construct the IBHS and the GS schemes. We then compare it to other static multi-signature schemes constructed with some hierarchy to explore efficiency.

Outline We cover the related work in Sec. 2 and the relevant notations and definitions Sec. 3. Then, an overview of trapdoors and signatures is provided in Sec. 4 with the HTDF, the constructions, and security. We construct the novel IBHS scheme in Sec. 5.3 which we use to construct the GS scheme in Sec. 6. The security proofs for both the novel constructions are given in Sec. 7 and conclude with some possible future works in Sec. 9.

2 Related work

We only consider works related to lattices and do not consider classical cryptographic assumptions like Discrete Logarithm or Bilinear pairings to maintain a fair comparison of the complexity. Lattice-based signature schemes, such as Crystals-Dilithium [10] and the GPV framework for signatures [14], have gained increased attention in the research literature due to their basis on quantum-resistant NP-hard problems like the SIS or LWE assumption. These signature schemes typically follow either the Hash and Sign (H/S) paradigm or the Fiat-Shamir with Aborts (FSwA) scheme to prove integrity. As the signature scheme from [14] forms the base for a lot of H/S signatures, we shall refer to the scheme as just the GPV signature from here.

FSwA schemes form the basis for several signature schemes with a challenge, commitment and verification structure. In LBC, such schemes can avoid discrete Gaussians for sampling the random variables and are generally more efficient. But schemes following FSwA like the NIST standardized Crystals-Dilithium [10], contains hash digests integrated within the scheme which can limit their ability to support homomorphic properties. H/S based signatures [14,16,18] include hashing of the message to obtain a fixed size digest and then it is signed by using the private key. The recipient can verify the signature by recomputing the digest from the message.

2.1 Homomorphic Signatures and Multi-Signatures

As our construction uses homomorphism in the IBHS scheme, we look into some multi-signature schemes employing homomorphism and morph them to suit our needs. Hiromasa et al. [17] introduced HS based on polynomials based on the SIS assumption with linear homomorphism. Then, Gorbunov et al. [16] introduced the first leveled fully HS scheme in the standard model that can evaluate arbitrary circuits over signed data under the hardness of SIS assumption. The size of the evaluated signature rises polynomially to the number of users but is independent of data size or circuit size. This scheme also provides amortized verification, context hiding, and composition of different arithmetic over signed data. Fully homomorphic message authenticators [13] use labeled programs again to connect circuits, identities, and messages. They use ciphertext verification than the normal signature approach and can be seen as a symmetric-key version of HS schemes. Boyen et al. [6] provide a way to generate HS by modifying the MP12 trapdoor [26] but is more expensive than the approach of [16] and a little too late.

Bendlin et al. [3] propose a threshold signature scheme based on the GPV signature scheme and honest majority secret sharing of the MP12 trapdoor from [26]. They use labeled programs to connect the list of signers and the message, which we use in our construction as well. Fiore et al. [11] propose multi-key homomorphic signatures (MK-HS) based on the work of [16] by providing evaluation keys. Their authenticators can be reduced to signatures by making the verification a public protocol, but their protocol does not allow for dishonest or malicious signers. Luo et al. [25] proposed an HS scheme that signs messages in a dataset altogether, achieving strong unforgeability. Lai et al. [19] use ZK-SNARKs, i.e., Succinct Non-Interactive Arguments of Knowledge, under lattice assumptions to convert a generic signature scheme into an MK-HS scheme, but this approach relies on non-standard assumptions.

Security for MK-HS Scheme The MK-HS scheme [9] is UF-CMA secure if, for any probabilistic polynomial-time (PPT) adversary, the likelihood of producing a forging signature on any message of the adversary's choosing is negligible. HS schemes cannot meet the usual unforgeability requirement as the primitive does allow the adversary to come up honestly with new signatures. During a training phase the adversary $\mathcal{A}$ is allowed to see the signatures of messages belonging to different datasets. $\mathcal{A}$ runs the following game with the challenger $\mathcal{C}$. Three conditions to avoid forgery the following unforgeability as in [9]:

1. **Type 1** forgery: $\mathcal{A}$ wins the game if $\mathcal{A}$ can produce either a signature on a message belonging to some previously unseen message list.
2. **Type 2** forgery: For some previously queried messages, $\mathcal{A}$ manages to produce a signature that verifies correctly but the message is not the evaluation of the labeled program given in the signature set.
3. **Type 3** forgery: $\mathcal{A}$ can cheat either by claiming an output on a dataset that was never queried, or an incorrect output of a given program, executed on a collection of messages for queried signatures.

2.2 Group Signature

Before exploring GS schemes, we look into identity-based signatures (IBS) and ABS to follow how GS is constructed. Most IBS schemes are based on the encryption scheme from [15] which we have seen used in the equivocal sense for signatures and extractable sense for encryption schemes, especially identity-based ones. The extraction mode of the trapdoor is modified to accommodate the identity to extract the user secret key from the master secret key. This means that constructing a homomorphic id-based signature without changes to this trapdoor becomes challenging. Even if we consider plain IBS we do not find many in literature considering they have simpler construction than ABS. For example, Pan et al. [28] compare their work to the GPV signatures, we do not find other IBS in the literature.

The initial GS schemes in the literature use trapdoors and provide security under the Shortest Vector Problem assumption. The security was only proven in the RO model due to the unsuitability of lattices with Non-Interactive Zero-Knowledge (NIZK) proofs. Static GS schemes [21,22,23,29] use NIZK for opening the messages to verify the identity and are based on the RO model always. Previous works on GS suggest that either a breakthrough result in lattice-based NIZKs or a different approach than the encrypt-then-prove paradigm is needed to obtain a lattice-based GS in the standard model. Dynamic GS schemes [24,7] use Bonsai trees or Merkle trees that act as a bloom filter to prove whether the particular identity belongs to that particular user. The challenge of constructing lattice-based GS in the standard model without NIZKs was solved by [18] who use ABS and Secret key Encryption (SKE) to construct GS. Proofs in the standard model are preferred as they are less expensive and shorter compared to NIZK.

In the Table 2.2, we list some variety of static GS schemes using different methods to produce group signatures.

Scheme	SM vs RO	ABS vs ZK	Assumption	Trapdoor
Ling et al [22]	RO	ZK - FSwA	RSIS/RLWE	Yes
Libert et al [21]	RO	ZK - FSwA	RSIS	No
Del Pino et al [29]	RO	ZK - FSwA	MSIS/NTRU	Yes
Boschini et al [5]	RO	ZK - FSwA	RSIS	Yes
Katsumata et al [18]	SM	ABS -H/S	SIS	Yes

Table 1. Static GS schemes - Standard Model (SM) vs Random Oracle (RO)

3 Background

Besides resilience to known quantum attacks, strong security proofs are an important feature of lattice-based constructions. Here, we show the basic definitions applied in the design of our group HS scheme. This section is included mainly to fix notations and ideas, and we refer to the original papers like [14,26] for further exposition. We

review some basic properties of lattices as used in previous works. Let λ be the security parameter. We use bold lower-case letters (e.g., $\mathbf{v}$) to denote vectors, and bold upper-case letters (e.g, $\mathbf{A}$) to denote matrices. The message m to be signed is represented in different forms as it can be packed using the CRT encoding or taken as just a bit based on the requirements of the scheme. For q an integer, $\mathbb{Z}_q$ denotes the standard group of integers modulo q.

The gadget matrix is a power-of-2 matrix $\mathbf{G}$ and its inverse is the binary decomposition algorithm $\mathbf{G}^{-1}$ which takes the input, a vector or matrix $\mathbf{x}$ and outputs the vector $\mathsf{BitDecomp} = \mathbf{G}^{-1}(\mathbf{x})$ such that $\mathsf{BitDecomp} \in \{0,1\}^{m\times k}$ and $\mathbf{G} \cdot \mathsf{BitDecomp} = \mathbf{x}$. When multiplying a vector and a matrix, we bit decompose the vector and multiply it with a power-of-2 matrix to match the dimensions as done in [15].

3.1 Lattices

A lattice Λ is a discrete subgroup of $\mathbb{R}^m$ with dimension $n \leq m$. In general, for cryptographic applications, it is restricted to $\mathbb{Z}^m$. It can be represented by a basis comprising n linear independent vectors of $\mathbb{R}^m$. Most of the trapdoor functionalities occur over the dual of the defined lattice $\Lambda^{\perp}$, with vectors orthogonal to the lattice vectors with a syndrome $\mathbf{u}$.

$$\Lambda^{\perp}_{\mathbf{u}}(\mathbf{A}) = \{e \in \mathbb{Z}^m : \mathbf{A}e = \mathbf{u} \bmod q\} \tag{1}$$

The formulation of the Module based SIS and LWE are similar to that of the standard assumptions but here the set of integers $\mathbb{Z}$ is replaced by the ring of algebraic integers $R = \mathbb{Z}[X]/(X^k + 1)$ of a number field K, over the $2k$-th cyclotomic polynomial. This introduces new parameters, like degree n of the number field, the integer d for the module rank, field tensor product, etc.

Discrete Gaussian Distribution Let $\Lambda \subseteq \mathbb{Z}^m$ be a lattice. The discrete Gaussian distribution $D_{\Lambda,s,c}$ is the m- dimensional Gaussian distribution centered at $\mathbf{c}$, but with support restricted to the lattice Λ. The one-dimensional (continuous) Gaussian distribution over $\mathbb{R}$, parameterized by $s \in \mathbb{R}^+$ defined by the density function.

$$\forall x \in \mathbb{R} : D_s(x) = \mathsf{exp}(-\pi(x/s)^2)/s$$

Short Integer Solution Assumption The SIS assumption was first suggested to be hard on average by Ajtai [1] and then formalized by Micciancio and Regev [27]. It is challenging to identify a non-trivial solution to a system of equations over a ring R, according to the SIS assumption. Given a system of equations of the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a matrix over a ring R and $\mathbf{b}$ is a vector over R, it is assumed that it is difficult to find a non-zero vector $\mathbf{x}$ in R^n such that $\mathbf{Ax} = \mathbf{b}$. In this work, we rely on the (MSIS) assumption [20] with super-polynomial β.

Definition 1. *Let n, m, q, β be integer parameters. In the $\mathsf{MSIS}(n, m, q, \beta)$ assumption, the attacker is given a uniformly random matrix $\mathbf{A} \in R_q^{n\times m}$ her goal is to find a vector $\mathbf{u} \in R_q^m$ with $\mathbf{u} \neq 0$ and $||\mathbf{u}||_\infty \leq \beta$ such that $\mathbf{Au} = 0$. For parameters $n = n(\lambda), m = m(\lambda), q = q(\lambda), \beta = \beta(\lambda)$ defined in terms of λ, the $\mathsf{SIS}(n, m, q, \beta)$ states any* PPT *attacker $\mathcal{A}$ we have*

$$\Pr\Big[\mathbf{Au} = 0 \quad \wedge \quad ||\mathbf{u}||_\infty \leq \beta \quad \wedge \quad \mathbf{u} \neq 0 : \mathbf{A} \xleftarrow{\$} R_q^{m\times n}, \mathbf{u} \leftarrow \mathcal{A}(1^\lambda, \mathbf{A})\Big] \leq \mathrm{negl}(\lambda)$$

Learning with Errors Assumption The LWE assumption is the foundation of various cryptographic systems such as the various encryption and signature scheme. Regev [30] showed the hardness of the LWE problem by describing a (quantum) reduction. LWE is defined as the problem of finding a short vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{b} + \mathbf{e} \bmod q$, where $\mathbf{A}$ is an $n \times m$ matrix over $\mathbb{Z}_q$ (integers modulo q), $\mathbf{b}$ is a vector of length n over $\mathbb{Z}_q$, and $\mathbf{e}$ is a random vector of small length. we will need to rely on the Decision MLWE assumption [4] with super-polynomial γ for the proposed scheme.

Definition 2. *Let n, m, q, γ be integer parameters. In the* Decision $\mathsf{MLWE}(n, m, q, \gamma)$ *assumption, the attacker is given a uniformly random matrix $\mathbf{A} \in R_q^{m \times n}$ and her goal is to find the vector $\mathbf{b}$ where $\mathbf{b} = \mathbf{Ax} + \mathbf{e} \bmod q$, where $\mathbf{x} \leftarrow R_q^n$ and $\mathbf{e} \leftarrow R_q^m$, distinguish the distribution of $(\mathbf{A}, \mathbf{b})$ from the uniform distribution over $R_q^{m \times n} \times R_q^m$.*

4 Lattice Trapdoors and Signatures

In the lattice setting, we only have so-called preimage sampleable trapdoor functions (and no trapdoor permutation). The quality of the trapdoor is defined to be the spectral norm or the singular values s_1. The quality of a trapdoor roughly corresponds to the Euclidean lengths of its vectors, i.e., the shorter the trapdoor, the better. In this section, we deliver the trapdoor and the signature schemes derived from it. Here, we level up the original SIS trapdoor to MSIS for the improved performance of our scheme.

Parameters We define the set of parameters $\mathsf{param} = \{n, m, q, \beta_{\mathsf{MSIS}}, \beta_{\mathsf{max}}, \beta_{\mathsf{init}}\}$ in terms of the parameters required by the trapdoor algorithm in Lemma 1 used in our scheme, where $n = poly(\lambda)$, $q = O(2^{poly(\lambda)})$, $m^* = O(n \log q)$, $\beta_{\mathsf{Sam}} = O(n\sqrt{\log q})$; $\beta_{max} = 2^{\omega(\log \lambda)d}$, where $d = poly(\lambda)$ is the bound on the depth of the circuits supported by our scheme. Let $\beta_{\mathsf{MSIS}} = 2^{\omega(\log \lambda)} \beta_{\mathsf{max}}\, m = \mathsf{max}\{m^*, n \log q + \omega(\log(\lambda))\} = poly(\lambda)$ and, finally, $\beta_{\mathsf{init}} = \beta_{\mathsf{Sam}} = poly(\lambda)$. Let q be a small prime so that we have $\mathsf{MSIS}(n, m', q, \beta_{\mathsf{MSIS}})$ assumption hold for all $m' = poly(\lambda)$. The following Lemma 1 is obtained by combining concepts from [26,14,8,16,4] to construct the extended HDTF.

Lemma 1. *There exist efficient algorithms* $\mathsf{TrapGen}, \mathsf{SamPre}, \mathsf{Sam}$ *such that the following holds. Given integers $n \geq 1, q \geq 2$ there exists some degree d, $k = \log_q$, $m = d(k+2)$, $w = dk$, C_0 is constant we have:*

- *For any $s \geq \omega(\sqrt{\log n})$ the algorithm* $\mathsf{TrapGen}\,(1^n, 1^m, s, q)$ *outputs matrices $\mathbf{A} \in R_q^{n \times m}$ and its trapdoor $\mathsf{td} \in R^{(m-w) \times w}$ such that $\mathbf{A}$ is statistically close to uniform,* td *has entries sampled from $D_{R,s}$ and $s_1(\mathsf{td}) \leq s \cdot C_0$. $(\sqrt{m-w} + \sqrt{w})$.*
- *For $\mathbf{A} \in R_q^{n \times m}$ with trapdoor* td*, syndrome $\mathbf{u} \in R_q^n$ and any $s \geq C_1 \cdot \sqrt{s_1(\mathsf{td})^2 + 1} \cdot \omega(\sqrt{\log n})$, the following distribution is statistically close to $D_{\Lambda_{\mathbf{u}}^{\perp}(\mathbf{A}), s}$:*

$$\{\mathbf{U} \mid \mathbf{U} \leftarrow \mathsf{Sam}(\mathbf{A}, \mathsf{td}, \mathbf{u}, s)\}$$

$\mathbf{U} \leftarrow \mathsf{Sam}(m, n, k, q)$ samples always a matrix $\mathbf{U} \in R_q^{m \times k}$ which satisfies $||\mathbf{U}||_\infty \leq \beta_{\mathsf{Sam}}$.

– *For any matrix* $\mathbf{A} \in R_q^{n\times m}$ *with trapdoor* td*, any matrix* $\mathbf{A}' \in R_q^{n\times w}$ *and any* $s \geq C_1 \cdot \sqrt{s_1(\mathsf{td})^2 + 1} \cdot \omega(\sqrt{\log n})$*,*

$$\mathsf{DelTrap}\left([\mathbf{A} \mid \mathbf{A}'], \mathsf{td}, s\right) \to \mathsf{td}' \in R_q^{m\times w}$$

where $[\mathbf{A} \mid \mathbf{A}']$ *with distribution independent of* td *and* $s_1\left(\mathsf{td}'\right) \leq s \cdot C_0 \cdot (\sqrt{m} + \sqrt{w})$*. Further, for* $\tilde{s} \geq \omega(\sqrt{\log n})$ *and under the same conditions, the following distributions are statistically close and*

$$\left\{\left(\mathbf{A}, \mathbf{A}', \mathsf{td}'\right) \mid (\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{GenTrap}\left(1^n, 1^m, \tilde{s}\right), \mathsf{td}' \leftarrow D_{R,s}^{m\times w}, \mathbf{A}' := \mathbf{A}\mathsf{td}' + \mathbf{G}\right\}$$

4.1 Homomorphic Trapdoors

Using the MSIS assumption, we design the HTDF to support the construction of our scheme using Lemma 1. HTDF is constructed as the following polynomial-time algorithms ($\mathsf{HTDF.KeyGen}$, f, Inv, $\mathsf{HTDF.Eval}^{in}$, $\mathsf{HTDF.Eval}^{out}$) with syntax:

- $\mathsf{HTDF.KeyGen}(1^\lambda)$: Select $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{Sam}(1^m, 1^m, q)$. Set $pk := \mathbf{A} \in R_q^{n\times m}$ and $sk = \mathsf{td}$.
- $f_{pk,x}$: Define a function from $f : \mathcal{U} \to \mathcal{V} : f_{pk,x} = \mathbf{A}\mathbf{U} + x\mathbf{G}$. f is well-defined on $R_q^{n\times m}$, but the domain of f is the subset $\mathcal{U} \subset R_q^{m\times m}$ to ensure short vectors.
- $\mathsf{Inv}_{sk,x}$: Define $\mathbf{U} \leftarrow \mathsf{Inv}_{sk,x}(\mathbf{V})$ to output $\mathbf{U} \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{V} - x\mathbf{G}, \mathsf{td})$.
- We define homomorphic evaluation algorithms used to prove the equivalence of homomorphism in the signature inputs and outputs.
  - $\mathsf{HTDF.Eval}^{in}(g, (x_1, \mathbf{U}_1), \ldots, (x_N, \mathbf{U}_N)) = \mathbf{U}^*$
  - $\mathsf{HTDF.Eval}^{out}(g, \mathbf{V}_1, \ldots, \mathbf{V}_N) = \mathbf{V}^*$

Security HTDFs possess the property of standard Chameleon hashes thereby lifting the security of the scheme from selective to adaptive security for single or MKHS. Perhaps the most natural security requirement would be *one-wayness*, meaning that for a random $v \leftarrow \mathcal{V}$ and any $x \in \mathcal{X}$ it should be hard to find a pre-image $u \in \mathcal{U}$ such that $f_{pk,x(u)} = v$. In particular, it should be difficult to find $u, u' \in \mathcal{U}$ and $x \neq x' \in \mathcal{X}$ such that $f_{pk,x}(u) = f_{pk,x'}(u')$ by the MSIS assumption. This is given the statistical indistinguishability requirements:

$$\mathbf{A} \approx \mathbf{A}' \quad (\mathbf{A}, \mathsf{td}, \mathbf{U}, \mathbf{V}) \approx (\mathbf{A}, \mathsf{td}, \mathbf{U}', \mathbf{V}')$$

where $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}(1^n, 1^m, q)$, $\mathbf{A}' \leftarrow R_q^{n\times m}$ and $\mathbf{U} \leftarrow \mathsf{Sam}(1^m, 1^k, q)$, $\mathbf{V} := \mathbf{A}\mathbf{U}$, $\mathbf{V}' \leftarrow R_q^{n\times k}$, $\mathbf{U}' \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{V}', \mathsf{td})$. The statistical distance between the random and calculated value is negligible in λ. Moreover, any $\mathbf{U}' \in \mathsf{SamPre}(\mathbf{A}, \mathbf{V}', \mathsf{td})$ always satisfies $\mathbf{A}\mathbf{U}' = \mathbf{V}'$ and $\|\mathbf{U}'\|_\infty \leq \beta_{\mathsf{Sam}}$.

4.2 Homomorphic Signatures Scheme

The H/S type of signature scheme fixes the public verification key as a trapdoor f and the signing key as the function's inverse. To sign a message m, a hash function $\mathbb{H}$ is used to compute $y = \mathbb{H}(m)$ in the range of the trapdoor f, and output the signature

$\sigma = f^{-1}(y)$. To verify (m, σ), one checks if $f(\sigma) = \mathbb{H}(m)$. Bellare et al. [2] show that such a scheme is existentially UF-CMA when f is a trapdoor permutation and $\mathbb{H}$ is a hash designed after a RO model.

Following the H/S paradigm, Gorbunov et al. [16] construct a HS scheme using trapdoor function $\mathcal{F}$ and message space $\mathcal{X}$ using Lemma 1 based on SIS assumption as follows:

$$\mathsf{S} = (\mathsf{PrmsGen}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{Process}, \mathsf{SignEval})$$

1. $\mathsf{PrmsGen}(1^\lambda, 1^N)$: Choose $\mathbf{V}_1, \ldots, \mathbf{V}_N$ by sampling $\mathbf{V}_i \xleftarrow{\$} \mathcal{V}$ and output $\mathsf{prms} = (\mathbf{V}_1, \ldots, \mathbf{V}_N)$.
2. $\mathsf{KeyGen}(\mathsf{prms}, 1^\lambda)$: Choose $(pk', sk') \leftarrow \mathsf{HTDF.KeyGen}(\lambda)$ and set $pk = pk', sk = (\mathsf{prms}, sk')$.
3. $\mathsf{Sign}_{sk}(x_1, \ldots, x_N)$: Sample $\sigma_i \leftarrow \mathsf{Inv}_{sk',x_i}(\mathbf{V}_i)$ and output $(\sigma_1, \ldots, \sigma_N)$.
4. Eval consists of two algorithms: $\mathsf{SignEval}_{pk}(g, (x_1, \sigma_1), \ldots, (x_N, \sigma_N))$, where we run $\mathsf{HTDF.Eval}^{in}_{pk'}$ on the signatures for function g and output σ. $\mathsf{Process}_{\mathsf{prms}}(g)$ where we pre-compute $\alpha_g = \mathsf{HTDF.Eval}^{out}_{pk'}(g, \mathbf{V}_1, \ldots, \mathbf{V}_N)$.
5. $\mathsf{Verify}_{pk}(\alpha_g, y, \sigma)$: For $y = g(x_1, \ldots, x_N)$, if $f_{pk',y}(\sigma) = \alpha(g)$ accept, else reject.

5 Construction of the Identity based Homomorphic Signature

We desire homomorphism as it helps in verifying the output of analysis on encrypted and signed data, so we make use of H/S-based signature schemes like [16] to create the IBHS scheme. The majority of constructions use the ABS approach to create the GS [7,18], which causes a substantial amount of computational overhead. We simply introduce IBHS to prior approaches in the GS scheme, which ensures tractability and anonymity.

We follow the notion of labeled data and programs just as [12] to indicate which of the data, user, and functions are to be signed/evaluated. We consider labeled programs $\mathcal{P}$, where each input bit of the program has an associated label indicating the data to be evaluated. A dataset is identified by an arbitrary string Δ, tags $\tau = \{\tau_i\}_{i \in \mathrm{ID}}$, labels $\mathcal{L} = \mathrm{ID} \times \tau$. This concept is necessary in order to not allow forgeries on any message using constant functions in the HS scheme that produces a signature σ_f that links the output of the computation $f(\mu)$ with the computation f itself. In other words, the signature should bind the output to the function used to compute it, rather than allowing arbitrary function evaluations.

Introducing identities, identity-based encryption schemes, and in some IBS was performed in the key extraction phase initially where the whole SamPre algorithm was used to extract the id key. While SamPre is performed at the signing step for us, we also use a specific trapdoor to ensure homomorphism. This becomes a challenging task. We retain the structure of Fiore et al. [11] and introduce the extract step to the process through the process of key delegation for our proposed scheme.

5.1 Definition

Definition 3. *An Identity-based Homomorphic Signature Scheme (*IBHS*) is defined as a tuple of PPT algorithms* IBS = (Setup, KeyExt, Sig, Eval, Verify)*, where*

1. $\mathsf{Setup}(1^\lambda)$*: takes as input the security parameter* 1^λ *and outputs a master public key* mpk *and a master secret key* msk*. We assume that* mpk *implicitly defines a message space* $\mathcal{M} = \mathcal{M}_{\mathsf{mpk}}$ *and an identity space* $\mathcal{ID} = \mathcal{ID}_{\mathsf{mpk}}$*.*
2. $\mathsf{KeyExt}(\mathsf{msk}, \mathsf{id})$*: takes as input a master secret key* msk *and an identity* $\mathsf{id} \in \mathcal{ID}$ *and outputs a secret key* $\mathsf{sk}_{\mathsf{id}}$*, we assume that* $\mathsf{sk}_{\mathsf{id}}$ *implicitly contains* id*.*
3. $\mathsf{Sig}(\mathsf{sk}_{\mathsf{id}}, \mathsf{m})$*: takes as input a secret key* $\mathsf{sk}_{\mathsf{id}}$ *and a message* $\mathsf{m} \in \mathcal{M}$ *and outputs a signature* σ*.*
4. Eval*: Taking as input* mpk*, message-signature pairs and an arithmetic circuit and outputs the evaluation of the pairs on the circuit* σ^**.*
5. $\mathsf{Verify}(\mathsf{mpk}, \mathsf{id}, \mathsf{m}, \sigma)$*: is deterministic, takes as input a master public key* mpk*, identity* $\mathsf{id} \in \mathcal{ID}$*, message* $\mathsf{m} \in \mathcal{M}$ *and signature* σ *and outputs a bit* $b \in \{0, 1\}$*.*

5.2 Security Model

The IBHS scheme consists of a Trusted Authority (TA) that distributes the user key pairs and possesses the master key pairs. We do not consider this authority to be malicious though we may allow attackers to monitor the activities of the trusted authority. The adversary can try to impersonate the signatures of the existing identities or try to join the group with maliciously constructed keys. *Correctness* demands that a signature generated by an honest and active user is always accepted by algorithm Verify, and that algorithm Trace can always identify that user.

5.3 Identity-based Homomorphic Signature Construction

Identity-based Homomorphic Signature (IBHS) scheme consists of the following four polynomial-time algorithms (IBHS.KeyGen, IBHS.KeyExt, IBHS.Sign, IBHS.Eval, IBHS.Verify). We follow the same notations as [11,16] in order to improve readability and construct the scheme based on MSIS assumption.

IBHS.KeyGen $\left(1^\lambda, 1^N\right)$. For the public parameter $\mathsf{pp} = \{\mathsf{param}, \mathcal{U}, \mathcal{V}, \mathcal{M}, \mathcal{T}, \mathcal{ID}\}$ as given below, we assume that pp as an implicit input for all subsequent algorithms. The labeled program $\mathcal{P} = (f, \ell_1, \ldots, \ell_t)$ can be generated, where f is any arithmetic circuit and different labels ℓ_i. The KeyGen algorithm takes as input the security parameter λ and generates the public parameters pp which includes the following:

- Class $\mathcal{F}$ of boolean circuits.
- Parameters param as defined above from Section 4.
- Preimage/signature space $\mathcal{U} = \left\{\mathbf{U} : \mathbf{U} \in R_q^{m \times m} : \|\mathbf{U}\|_\infty \leq \beta_{\max}\right\}$.
- Range space $\mathcal{V} = \left\{\mathbf{V} : \mathbf{V} \in R_q^{n \times m}\right\}$,
- Message space over $\mathcal{M} = \{0, 1\}$
- Tags $\mathcal{T} = [\mathsf{T}]$, $\mathsf{T} = poly(\lambda)$, $\mathsf{T} \in \mathbb{N}$.
- Identities $\mathcal{ID} = [N]$, $N = 2^\lambda$.
- Label space $\mathcal{L} = \mathcal{ID} \times \mathcal{T}$ and label $\ell = (id, \tau)$.

The key generation algorithm takes as input the public parameters pp and generates a pair $(\mathsf{mpk}, \mathsf{msk})$ where $\mathsf{mpk} = \mathbf{A}$ and $\mathsf{msk} = \mathsf{td}$. Using Lemma 1, $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}\,(1^n, 1^m, q)$ to generate a matrix $\mathbf{A} \in R_q^{n \times m}$ along with its trapdoor td.

IBHS.KeyExt(msk, id). Define a function $\mathbf{H}$ such that $\mathbf{H}(\mathsf{mpk}, id) = \mathbf{H}_{id} = \mathbf{A}\mathbf{R}_{id} + \mathbf{G}$ where $\mathbf{R}_{id}$ is randomly sampled from $R_q^{n \times n \log q}$. Then, use the trapdoor delegation algorithm to extend the trapdoor from td to $\mathsf{td}_{id} \leftarrow \mathsf{DelTrap}([\mathbf{A} \mid \mathbf{H}_{id}], \mathsf{td}, s)$, where s is a bound for the trapdoor delegation using the Lemma 1. It returns $\mathsf{upk}_{id} = [\mathbf{A} \mid \mathbf{H}_{id}]$, $\mathsf{usk}_{id} = \mathsf{td}_{id}$.

IBHS.Sign $(\mathsf{usk}_{id}, \ell, m)$. The signing algorithm takes as input a user secret key usk_{id}, a label to indicate $\mathbf{V}$'s and id of the signer $\ell = (id, \tau)$, a message m. The signature is of the form $\forall\ id \in \mathbf{I}$, where $\mathbf{I} \subseteq \mathcal{ID}$ and their respective signatures $\mathbf{U}_{id} \in \mathcal{U}$ is represented as

$$\sigma_i = (m_i, \mathbf{I}, \{\mathbf{U}_{id}\}_{id \in \mathbf{I}}) \tag{2}$$

where $\mathbf{U}_{id} \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{V}_\ell - m\mathbf{G}, \mathsf{td}_{id})$ using Lemma 1.

Because of the evaluation phase, the signature gets updated from $\mathbf{I} = \{id\}$ to different signers to maintain the full history of the signatures.

IBHS.Eval$(g, \{\sigma_i\}_{i \in [t]})$. The evaluation algorithm takes $g : \mathcal{M}^t \longrightarrow \mathcal{M}$ for f an arithmetic circuit over R_q with additions and multiplications, a set of signatures $\{\sigma_i\}_{i \in [t]}$ where $t \leq \mathsf{T} \cdot N$ with a set of evaluation keys implicitly for further calculations. From [16,11], we follow how to evaluate additions or multiplications with two inputs. This can be extended to as many inputs as desired.

Let f be a such an arithmetic gate with inputs $\sigma_1 = (m_1, \mathbf{I}_1, \mathbf{U}_1)$ and $\sigma_2 = (m_2, \mathbf{I}_2, \mathbf{U}_2)$. To generate the evaluated signature σ, first set $\mathbf{I} = \mathbf{I}_1 \cup \mathbf{I}_2$. To ensure that both signatures to be evaluated are the same size, check if every id in $\mathbf{I}$ is in $\mathbf{I}_1$ or $\mathbf{I}_2$ and set $\hat{\mathbf{U}}_i^{id}$ to $\mathbf{U}_i^{id}$ if present and to $\mathbf{0}_m$ if in the other set to form $\hat{\mathbf{U}}_1$ and $\hat{\mathbf{U}}_2$. This is to make sure that both signatures can be combined over the same number of parties, $|\mathbf{I}|$. For $i = \{1, 2\}$, let the noise be $\beta_i = \|\mathbf{U}_i\|_\infty$ for each signature.

- If g is addition, compute $m = m_1 + m_2$

$$\mathbf{U} = \{\hat{\mathbf{U}}_1^{id} + \hat{\mathbf{U}}_2^{id}\}_{id \in \mathbf{I}}; \tag{3}$$

- If g is multiplication, compute $m = m_1 \cdot m_2$ and $\mathbf{V_1} = \sum_{id \in \mathbf{I}_1} \mathbf{A}_{id}\mathbf{U}_{id} + m_1\mathbf{G}$.

$$\mathbf{U} = \{\mathbf{U}_{id}\}_{id \in 1} = \left\{m_2\hat{\mathbf{U}}_1^{id} + \hat{\mathbf{U}}_2^{id} \cdot \mathbf{G}^{-1}(\mathbf{V}_1)\right\}_{id \in \mathbf{I}} \tag{4}$$

Scalar multiplication can be performed similarly.

IBHS.Verify$(\mathcal{P}, \{\mathsf{upk}_{id}\}_{id \in \mathcal{P}}, m, \sigma)$. Here, the verification algorithm takes as input a labeled program $\mathcal{P}$, the set of the verification keys $\{\mathsf{mpk}_{id}\}_{id \in \mathcal{P}}$ of users involved in the program $\mathcal{P}$, the message m and it's signature σ. From the circuit g and the values $\{\mathbf{V}_{\ell_i}\}_{i \in [t]}$, compute $\mathbf{V}^*$ for the signatures as done in $\mathsf{IBHS.Eval}$ or HTDF_{out} in Section 4.2. $\mathbf{V}^*$ can be pre-computed and re-used every time to verify for the same $\mathcal{P}$. It then performs the following checks and rejects if at least one check fails, otherwise it accepts the signature.

1. The list of identities in σ should match with the labels of $\mathcal{P}$; $\mathbf{I} = \{id : id \in \mathcal{P}\}$.
2. The verification algorithm parses $\mathbf{U} = \{\mathbf{U}_{id}\}_{id \in \mathbf{I}}$ and checks if
   - The noise is bounded such that $\|\mathbf{U}\|_\infty \leq \beta_{\mathsf{max}}$
   - $\sum_{id \in \mathbf{I}} \mathbf{A}_{id}\mathbf{U}_{id} + m\mathbf{G} = \mathbf{V}^*$

Correctness The correctness of the IBHS.Verify can be checked easily using Lemma 1 and the use of labeled programs. The bounded noise growth is an important factor to be checked. The straightforward construction of $\mathbf{V}^*$ ensures that the input and the output signatures match. This can be checked manually to show and this is done in the Equation 2. The correctness of the Eval can be verified using the same methods as [11] because we do not change the process past key extraction.

6 Construction of the Group Signature Scheme

As a consequence of the construction of the IBHS scheme, we construct a GS scheme as per Def. 4. We developed the IBHS scheme first to demonstrate key extraction and the updations and then leveled it up into the GS scheme by making minor alterations to the construction. The signature is homomorphic only when the signature corresponds to the same person. We demonstrate the construction for just one group, as the extension to multiple groups and hierarchies is straightforward.

We make changes only in the key extraction step to elevate the security of the scheme. The labeled programs can be indirectly accessed by using another hash $\mathsf{VGen} : \{0,1\}^d \rightarrow \mathcal{V}$ which generates the values $\mathbf{V}_i$ as required. This way we do not reveal identity in the public parameters of the user. This helps produce shorter keys and obfuscate the identity of the signer.

6.1 Definition

A GS scheme consists of a group manager GM who chooses who can join the group, a tracing manager TM who can open signatures, and a group of possible group members. We can collapse the two managers to the same trusted authority as well. Users may join or leave the group at GM's discretion.

Definition 4. *A Group Signature (*GS*) scheme consists of the following seven polynomial-time algorithms:*

$$\mathsf{GS}_{\mathsf{MSIS}} = (\mathsf{GS.KeyGen}, \mathsf{GS.KeyExt}, \mathsf{GS.Sign}, \mathsf{GS.Verify}, \mathsf{GS.Trace})$$

- GS.KeyGen : *On input security parameter λ and the number of users N, along with the public parameters* pp*, it generates as output* $(\mathsf{gpk}, \{\mathsf{gsk}_i\}_{i \in N})$ *for* GM *and for* TM *it generates* gok*.*
- GS.KeyExt : *Using the* $\{\mathsf{gsk}_i\}_{i \in N}$*, extract the user secret keys whenever a new user joins the group. Taking as input $id \in \mathcal{ID}$, it outputs a user secret key* usk_{id} *of a user.*
- GS.Sign : *Taking as input* gpk*, user secret key* usk_{id} *of the sender, and a plaintext message $m \in \mathcal{M}$, it outputs a signature σ_{id}.*
- GS.Verify : *Taking as input* $(\mathsf{gpk}, m, \sigma^*)$*, this algorithm returns* 1 *if is valid relative to* gpk*.*
- GS.Trace : *Using the opening key* gok *and* $(\mathsf{gpk}, m, \sigma^*)$*, the* TM *traces which of the users in the group signed the message and outputs in id otherwise $\perp$.*

6.2 Security Model

A GS scheme consists of a group manager GM who chooses who can join the group, a tracing manager TM who can open signatures, and a group of possible group members. Users may join or leave the group at GM's discretion. We can collapse the TM and GM to be the same authority that is trusted. The adversary model follows from the IBHS. We require the following properties in addition to the unforgeability to provide security for GS.

Anonymity requires that it is infeasible for any PPT adversary to distinguish signatures generated by two active users of its choice at the chosen epoch, even if it can corrupt any user, can choose the key of GM, and is given access to the Trace oracle.

Traceability ensures that the adversary cannot produce a valid signature that cannot be traced to an active user at the chosen epoch, even if it can corrupt any user and can choose the key of TM. *Tracing Soundness* requires that it is infeasible to produce a valid signature that traces to two different users, even if all group users and both managers are fully controlled by the adversary.

6.3 Secret Key Encryption

The introduction of an SKE scheme is essential to maintain the anonymity of the user who signs on behalf of the group. The decryption of this circuit is used in the Trace function in order to identify who signed the GS. The decryption key is only possessed by the trusted authority who generates it, like the GM or TM. The SKE consists of three polynomial time algorithms $\mathsf{SKE} = (\mathsf{SKE.KeyGen}, \mathsf{SKE.Enc}, \mathsf{SKE.Dec})$. The use of SKE is common to all GS schemes in lattices proposed so far and therefore is not an overhead.

We follow the GSW construction [15] for encryption as the structure of the schemes matches very effectively. While this is only Chosen Plaintext Attack (CPA) secure because of the homomorphism, it can be upgraded to Chosen Ciphertext Attack (CCA) security by working in conjunction with a MAC as suggested in [18]. We do not consider the CCA notion of security as the signatures under a single key are homomorphic already, therefore exhibiting only CPA-like properties. This SKE would be secure under the MLWE problem by upgrading to appropriate parameters.

6.4 GS Construction

GS.KeyGen $\left(1^{\lambda}, 1^{N}\right)$. The KeyGen algorithm takes as input the security parameter λ and the parameter from IBHS such as $\mathcal{F}, \mathsf{param}, \mathcal{V}, \mathcal{M}, \mathcal{T}, \mathcal{L}$ with group identities $\mathcal{GID} = [N]$, $N = 2^{\lambda}$, $gid \in \mathcal{GID}$. It gives the output as the public parameters $\mathsf{pp} = (\mathsf{param}, \mathcal{U}, \mathcal{V}, \mathcal{M}, \mathcal{T}, \mathcal{GID})$. It outputs the keys $(\mathsf{gpk}, \mathsf{gsk}, \mathsf{gok})$ where $\mathsf{gpk} = \mathbf{A}$, $\mathsf{gsk} = \mathsf{td}$, and $\mathsf{gok} = \{\mathbf{s}, e\}$. Using a random $\mathbf{s}$ and error e chosen according to MLWE parameters, we calculate the encryption key for the SKE as $\mathbf{B} = \left[\frac{\mathbf{A}}{\mathbf{sA}+e}\right]$. Using Lemma 1, $(\mathbf{A}, \mathsf{td}) \leftarrow \mathsf{TrapGen}\left(1^{n}, 1^{m}, q\right)$ to generate a matrix $\mathbf{A} \in R_q^{n \times m}$ along with its trapdoor td.

GS.KeyExt(gsk, gid). For every group user gid, encrypt the identity as $\mathbf{H}_{gid}$ such that $\mathbf{H}_{gid} = \mathsf{SKE.Enc}(\mathsf{gsk}, gid)$ where $\mathbf{R}$ is randomly sampled from $R_q^{n \times n \log q}$. In the following construction, in order to match the number of rows in the $\mathsf{DelTrap}$ phase, we can add a zero vector to $\mathbf{A}$ to generate the extended basis. Then, use the trapdoor delegation algorithm to extend the trapdoor from td to $\mathsf{td}_{gid} \leftarrow \mathsf{DelTrap}([\mathbf{A} \mid \mathbf{H}_{gid}], \mathsf{td}, s)$, where s is a bound for the trapdoor delegation using the Lemma 1. For the label space of the user, perform $\mathsf{VGen}(\mathsf{seed}) = \{\mathbf{V}_i\}_{i \in [T]}$. It returns group user public key $\mathsf{gevk}_{gid} = \{\mathbf{H}_{gid}, \mathsf{seed}\}$, the group user secret key $\mathsf{gusk}_{gid} = \mathsf{td}_{gid}$, .

GS.Sign $(\mathsf{gusk}_{gid}, \mathsf{seed}, m)$. The signing algorithm takes as input group user secret key gusk_{gid}, a label to indicate $\mathbf{V}$'s and gid of the signer $\ell = (gid, \tau)$, a message m. The signature is given by $\mathbf{U}_{gid} \leftarrow \mathsf{SamPre}(\mathbf{A}, \mathbf{V}_\ell - m\mathbf{G}, \mathsf{td}_{gid})$ using Lemma 1. Therefore, the signature $\mathbf{U}_{gid} \in \mathcal{U}$ is given similarly as Equation 2 and for group setting it gives output as signature $\sigma_i = (m_i, \mathsf{gevk}_{gid}, \mathbf{U}_{gid})$.

GS.Verify$(\mathsf{gpk}, m, \sigma)$. Here, the verification algorithm takes as input a labeled program $\mathcal{P}$, the set of the verification keys gpk_{gid} of users involved in the program $\mathcal{P}$, the message m and it's signature $\sigma = (m, \mathsf{gevk}, \mathbf{U})$. Compute from the circuit f and the values $\{\mathbf{V}_{\ell_1}, \ldots, \mathbf{V}_{\ell_t}\}$, computes $\mathbf{V}^*$ for the general circuit on the signatures as in $\mathsf{IBHS.Eval}$ in Section 4.2. The verification algorithm parses $\mathbf{U}$ and checks:

1. $\|\mathbf{U}\|_\infty \leq \beta_{\mathsf{max}}$
2. $[\mathbf{A} \mid \mathbf{H}] \cdot \mathbf{U} + m \cdot \mathbf{G} = \mathbf{V}^*$

GS.Trace$(\mathsf{gpk}, \mathsf{gok}, m, \sigma)$. It first runs $\mathsf{Verify}(\mathcal{P}, \mathsf{gpk}, m, \sigma)$ and returns $\perp$ if the verification result is $\perp$. Otherwise, it parses $\sigma = (m, \mathbf{H}, \mathbf{U})$. It then computes $d = \mathsf{SKE.Dec}(gok, \sigma)$ which returns the gid of the user. As the SKE is secure under MLWE, this cannot be opened by anyone else.

Correctness The correctness of the GS scheme follows directly from the correctness of the IBHS and the SKE. We can easily prove that a signature that was correctly generated passes the verification. As the signing and verification are taken directly from the IBHS, we have $\mathsf{IBHS.Verify}$ valid by the correctness of IBHS. Then, by the correctness of SKE, we have the Trace as also valid. Therefore, the $\mathsf{GS.Verify}$ step is valid.

7 Security Proof

Any HS scheme fails to fulfill the typical condition for unforgeability since the attacker is permitted to generate new signatures honestly. In a multi-signer setting, the security of a cryptographic scheme depends not only on the properties of the scheme itself but also on the trustworthiness of the signers. The adversary may attempt to corrupt some signers or generate key pairs with malicious intent to compromise the security of the scheme.

Therefore, in such scenarios, it is crucial to thoroughly analyze the unforgeability of the scheme against insider attacks. Such analysis ensures that even if some of the signers are compromised, an attacker cannot generate valid signatures that would allow them to impersonate legitimate signers or forge new messages. Keeping this in mind, we now prove the security of the IBHS and the GS in the standard model under the MSIS and MLWE assumption.

7.1 IBHS security

We consider a trusted authority to distribute the usk to the users during the key generation and extraction phases. We do not consider the trusted authority to be malicious in our scheme. The adversary can be an external party that has corrupted one of the users or is trying to impersonate them. If we consider insider corruption, which mandates that a group of corrupt signers cannot produce valid signatures outside the queries of the adversarial model, NIZK can be used as a viable solution and can sometimes be combined with Chameleon hashes to achieve adaptive security.

Theorem 1. *The scheme* $\mathsf{IBHS} = (\mathsf{KeyGen}, \mathsf{KeyExt}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ *described in Sec. 5.3 is secure under* $\mathrm{UF} - \mathrm{CMA}$*, assuming that the* MSIS *assumption is hard for parameters* β*,* s *as in Sec. 5.3, if for every PPT adversary* $\mathcal{A}$ *the advantage that the Game 1 succeeds is negligible in* λ*.*

Proof. We first consider the security of MKHS and adapt to the IBHS scheme. The HTDF immediately provide selective security which can be transformed to adaptive security of the HS schemes using the different key pairs for signing. The reason behind this is that the trapdoors are pre-image sampleable only, not trapdoor permutations. There is no known construction of trapdoor permutations in lattices. According to Catalano et al. [9], the scheme we have already considered from [11] for MKHS satisfies semi-adaptive security, which can be extended to strong adaptive security by combining an OR gate based HS scheme derived from the HS to the original scheme. In order to thwart statistical learning attacks on trapdoor sampling, it is crucial to use a discrete Gaussian distribution for randomness, despite its high cost.

Since this scheme works for a single dataset, Type 1 forgeries mentioned in Def. 2.1 cannot occur. However we do use multiple messages and tags, we do have to consider the other two forgeries. The Type 2 forgery equates to breaking MSIS in the function f defined in 4.2. While Type 2 forgeries focus on a tuple of original messages, Type 3 forgeries focus on the labeled program and group identities. They can be easily related to the way key extraction is performed, to thwart identity impersonation and reuse of labeled programs.

Algorithm 1 Game UF-CMA

$(\mathcal{L}_{id}, \mathcal{L}_m) \leftarrow \mathcal{A}(1^\lambda)$
$(\mathsf{mpk}, \mathsf{msk}) \leftarrow Setup(1^\lambda)$
for $id \in \mathcal{L}_{id}$:
  $\mathsf{usk}_{id} \leftarrow \mathsf{KeyExt}(\mathsf{msk}, id)$
  $\mathcal{L}_{\mathsf{usk}} := \mathcal{L}_{\mathsf{usk}} \cup \mathsf{usk}_{id}$
for $(id, m) \in \mathcal{L}_m$:
  $\sigma \leftarrow \mathsf{Sign}(\mathsf{usk}_{id}, m)$
  $\mathcal{L}_{\mathsf{Sign}} := \mathcal{L}_{\mathsf{Sign}} \cup \sigma$
$(\mathcal{P}^*, m^*, \mathbf{I}^*, \sigma^*) \leftarrow \mathcal{A}_{\mathbf{H}}(\mathsf{mpk}, \mathcal{L}_{\mathsf{usk}}, \mathcal{L}_{\mathsf{Sign}})$
if $\mathbf{I}^* \in \mathcal{L}_{id}$: return 0
if $(\mathcal{P}^*, m^*) \in \mathcal{L}_{\mathcal{P}}$: **return** 0
return $\mathsf{Verify}(\mathcal{P}^*, \mathsf{mpk}, \mathbf{I}^*, m^*, \sigma^*)$

We assume that the adversary has the list of all queries for messages signed, identities used, and signatures. The challenger has only the master secret key which is updated after the key extract in addition to what the adversary possesses. With all the information like the signature and message list, the adversary produces a signature such that for an identity and message pair not in the list, the probability that the verification algorithm returns 1 is negligible.

If the adversary $\mathcal{A}$ wins the game, i.e., $\mathsf{IBHS.Verify}(\mathcal{P}^*, \mathsf{mpk}, m^*, \sigma^*) = 1$, then the adversary has produced the signature σ^* for the messages evaluated under the function f^* in $\mathcal{P}^*$ for at least one identity that is not present in the list of $\mathcal{ID}$s already, i.e., then if σ is obtained by honestly evaluating the $\mathcal{P}^*$ on the messages, then $\sum_{id \in \mathbf{I}} \mathbf{A}_{id}\mathbf{U}_{id} + m \cdot \mathbf{G} = \sum_{id \in \mathbf{I}^*} \mathbf{A}_{id}\mathbf{U}^*_{id} + m^* \cdot \mathbf{G}$.

For the difference $\hat{\mathbf{U}} = \mathbf{U}^*_{id} - \mathbf{U}_{id}$ computed through $\mathsf{IBHS.Eval}$, and $\hat{m} = m^* - m$, then $\mathbf{A}\hat{\mathbf{U}} + \hat{m} \cdot \mathbf{G}$, for every id in the labeled program. By assuming a certain random $\mathbf{r}$, $s = \mathbf{A}r$ and $\mathbf{r}' = \mathbf{G}^{-1}(\hat{m}^{-1}s)$, we can go on to prove that such a signature can be produced only when $\mathbf{A}(\hat{\mathbf{U}}\mathbf{r}' - \mathbf{r}) = 0$, by substitutions. This can only be possible when the MSIS problem is not hard. Also, by choosing a small enough $\mathbf{r}$, the condition that the vector $(\hat{\mathbf{U}}\mathbf{r}' - \mathbf{r})$ be small is also satisfied.

The statistical closeness of the $\mathbf{V}_i$ for those constructed using the function f described in Sec. 4.2, i.e., $\mathbf{V} = \mathbf{V}\mathbf{U} + \mathbf{G}$ for the key extraction process makes it secure to protect the user secret key from $\mathcal{A}$ The main difference between our scheme and the multi-signature scheme described in [11] is the way that the keys are extracted, enabling the formation of a GS. The trapdoor delegation is secure as long as the msk is under a trusted authority that distributes the user secret keys. The value of $\mathbf{H}_{gid}$ is considered secure under MSIS as similar proofs can be seen for hash functions under the Ring SIS assumption in [18]. Therefore, our IBHS scheme is secure under the MSIS assumption.

7.2 GS security

We do not consider cache-based attacks formally because even though the identity is encrypted, the adversary can find out that the same identity has been signed multi-

ple times because of the presence of the labeled programs in the Verification algorithm which is a part of the public parameters.

Lemma 2. *The GS scheme described in Sec. 6.4 preserves the anonymity and the traceability of the scheme under the* MLWE *assumption*

Proof. We prove the anonymity aspect of the construction, i.e., nobody can trace which identity signed the group signature. For this we follow the sequence of games defined similarly to [18] as that remains valid. We add another game to the sequence as the signatures allow homomorphism for the same identity. Let E be the event of $\mathcal{A}$ succeeding with negligible probability.
Game 0: We define this as an experiment between the challenger and the adversary $\mathcal{A}$. Here, we have $\Pr[\mathrm{E}_0] = \epsilon$.
Game 1: Here, $\mathcal{A}$ tries to use the signatures already queried and matches the $\mathbf{H}$ values and creates a new signature by running IBHS.Eval on the signatures. In this case, the signature passes GS.Verify as the values match and the evaluation algorithm is correct. Therefore $\mathcal{A}$ cannot break find the identity or the gusk to impersonate the signer as that would imply breaking the IBHS scheme which in turn breaks MSIS.

The encryption step in GS.KeyExt generates $\mathbf{H}_{gid}$ for every gid and this step is run by the trusted authority as it involves the msk and then the extracted user signing key is distributed to the users. The DelTrap algorithm cannot be replicated without the master secret key. The $\mathbf{H}_{gid}$ value is given along with the signature to add it to the verification step. Though $\mathcal{A}$ can notice that two signatures are signed by the same party because of the matching $\mathbf{H}_{gid}$, they cannot extract the identity as it is equivalent to breaking the MLWE assumption. Therefore, $\Pr[E_1] \geq \Pr[E_0] - \mathrm{negl}(\lambda)$.
Game 2: In this game, the way $\{\mathbf{V}^{(i)}_{i\in[T]}$ are chosen is changed. At the beginning of the game, the challenger receives $1^N, \{i\}_{i\in[N]}$, and a non empty $\mathcal{S} \subseteq \mathbf{GID}$ from $\mathcal{A}$. Then, the challenger samples $\mathbf{R}^{(i)}$ for $i \in [N]$ and sets the matrices as

$$\mathbf{V}^{(i)} = \begin{cases} \mathbf{A}\mathbf{R}^{(i)} + i\mathbf{G} & \text{if } i \in \mathcal{S} \\ \mathbf{A}\mathbf{R}^{(i)} + i_{min}\mathbf{G} & \text{if } i \notin \mathcal{S} \end{cases} \quad \text{where} \quad i_{min} := \min\{i \mid i \in \mathcal{S}\}.$$

Then the challenger gives $\left\{\mathsf{gusk}_i := \left(\mathbf{R}^{(i)}\right)\right\}_{i\in\mathcal{S}}$ to $\mathcal{A}$. By Lemma 1, the distribution of $\left(\mathbf{A}, \left\{\mathbf{V}^{(i)}\right\}_{i\in[T]}, \left\{\mathbf{R}^{(i)}\right\}_{i\in\mathcal{S}}\right)$ in both games are statistically close. Also, the DelTrap in the lemma follows the statistical closeness as well. Therefore, we have $\Pr[\mathrm{E}_2] \geq \Pr[\mathrm{E}_1] - \mathrm{negl}(\lambda)$.

The identity may appear redundantly in the labeled program where the function and the identity are intertwined. But this proves that it is negligible to succeed with the sampling also because of the hashing algorithm for the seed. This method is also recommended in [16] for shorter keys and in our case means that the values can be cached but the identity is not revealed.
Game 3: In this game, we change the way $\mathbf{A}$ is sampled. Namely, the challenger samples only $\mathbf{A}$ instead of sampling it with the trapdoor using TrapGen and DelTrap. By the Lemma, the distribution of $\mathbf{A}$ in this game is statistically close to that of the previous game. Therefore, we have $\Pr[\mathrm{E}_3] \geq \Pr[\mathrm{E}_2] - \mathrm{negl}(\lambda)$.

Secondly, we prove the traceability of the scheme by making sure that only the trusted authority with the opening key will be able to trace the identity of the signer. This is straightforward as the TM is the only entity with the opening key, which is the decryption key of the SKE. It is easy to see when GS.Open is invalid then the $i \notin \mathcal{GID}$. Therefore, the trusted authority can find the identity of the signer by decrypting $\mathbf{H}_{gid}$.

Theorem 2. *The GS scheme described in Sec. 6.4 is secure against unforgeability under the* MSIS *assumption and provides anonymity and traceability under the* MLWE *assumption.*

Proof. Assume that the challenger always aborts if the message and signature pair is from a list of queries already performed as the homomorphism allows new signatures to be obtained in this manner. The labeled program is very useful here in order to $\mathcal{C}$ only consider a signature only if it is either on a message or an identity that has already not been queried. The unforgeability follows from the construction of the IBHS scheme and the game described in 1. The proof for the anonymity and trace follows from Lemma 2. Altogether the GS is secure under both the MSIS and MLWE assumptions.

8 Improvements

We do not implement the scheme as implementations are not widely available for GS schemes under the standard model based on the MSIS problem [7,18]. Therefore, we compare the performance to other static GS schemes under the Standard model with parameters like key size and signature size.

Succinctness The derived signature should be short, with length independent of the size of the data to avoid further attacks. If a signature of lattice-based signature schemes consists of a single lattice vector, i.e., it increases at a rate of order $O(1)$, the size of the signature is called short. In our scheme, because the signature $\mathbf{U} \leq \beta_{max}$, the signatures even after evaluation are bounded.

Therefore for the IBHS scheme, the signature size depends linearly on the number of participants while the size of each signature remains the same. This is made possible because of the homomorphism of the underlying scheme. In the GS scheme, as there is only one signature representing the group, the signature size is succinct and bounded.

Packing SIMD techniques are used to perform operations on multiple elements of data simultaneously. One of the popular SIMD methods used in lattice-based cryptography is the Chinese Remainder Theorem (CRT) packing, which splits a Ring of integers into several smaller rings, each of which can be represented using a smaller modulus. If $q = q_1, q_2, \cdots, q_k$ be a product of small primes, where q_i are chosen such that q is larger than all the coefficients in the secret and error vectors. This enables us to represent the vector using several smaller integers instead of a single larger integer. CRT packing is particularly useful for lattice-based encryption schemes that use ideal lattices. The use of CRT packing in ideal lattices enables efficient SIMD operations on ciphertexts. This enables efficient SIMD operations on ciphertexts, such as addition and multiplication.

9 Conclusion and Future works

In this paper, we propose a new approach to constructing group signature schemes using HTDFs. Group signature schemes are an important primitive in cryptography that provides anonymity and traceability for a group of users. Our approach focuses on constructing an identity-based homomorphic signature scheme that is simpler in construction and requires less discrete Gaussian sampling and no NIZK proofs.

Our proposed scheme allows packing more data into the signatures by elevating the existing homomorphic trapdoor from the SIS assumption to the MSIS assumption to enable packing techniques. We compare our scheme with other static multi-signature schemes constructed with some hierarchy to explore efficiency. Our results show that our proposed scheme outperforms previous schemes in terms of efficiency and provides strong unforgeability.

We also provide some ways to make our scheme more functional.

- Dynamic setting: The scheme presented in this paper is proposed in a static setting and can be made dynamic by introducing an access structure like Merkle Trees or Bloom Filters and then combined with NIZKs and SKE to generate signatures that can be verified based on time like the work of [7]. Also, the ZK proof provides a way to avoid insider corruption.
- Context hiding: The property of homomorphism can reveal which signatures have been modified to output the current signatures. The homomorphism can be removed and rectified by introducing a hiding signature using [8] or by using NIZKs in the signature.
- Increasing efficiency: Using Fast Fourier Transforms can speed up computation. Discrete Gaussian sampling can now be efficiently done in lattice structures like rings due to recent advancements, and this has been used by FALCON to speed up their GPV-based signature scheme. The extension requires a full-rank CRT encoding for value aggregation during sampling. Our scheme can also be made more succinct by introducing SNARKs to the signature protocol.
- Parametrization: Providing concrete parameters can aid with the implementation of the scheme in the future.

We believe that our proposed scheme provides an efficient and secure solution for GS schemes using HTDF. This can have important implications for various applications in cryptography, where GS schemes are required to provide anonymity and traceability in a secure manner.

References

1. Ajtai, M.: Generating hard instances of the short basis problem. In: Automata, Languages and Programming: 26th International Colloquium, ICALP'99 Prague, Czech Republic, July 11–15, 1999 Proceedings 26. pp. 1–9. Springer (1999)
2. Bellare, M., Micali, S.: How to sign given any trapdoor permutation. J. ACM **39**(1), 214–233 (1992)

3. Bendlin, R., Krehbiel, S., Peikert, C.: How to share a lattice trapdoor: Threshold protocols for signatures and (H)IBE. In: Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 7954, pp. 218–236. Springer (2013)
4. Bert, P., Eberhart, G., Prabel, L., Roux-Langlois, A., Sabt, M.: Implementation of lattice trapdoors on modules and applications. In: Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021,Proceedings. Lecture Notes in Computer Science, vol. 12841, pp. 195–214. Springer (2021)
5. Boschini, C., Camenisch, J., Neven, G.: Floppy-sized group signatures from lattices. In: Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10892, pp. 163–182. Springer (2018)
6. Boyen, X., Fan, X., Shi, E.: Adaptively secure fully homomorphic signatures based on lattices. Cryptology ePrint Archive (2014)
7. Canard, S., Georgescu, A., Kaim, G., Roux-Langlois, A., Traoré, J.: Constant-size lattice-based group signature with forward security in the standard model. In: Provable and Practical Security: 14th International Conference, ProvSec 2020, Singapore, November 29–December 1, 2020, Proceedings 14. pp. 24–44. Springer (2020)
8. Cash, D., Hofheinz, D., Kiltz, E., Peikert, C.: Bonsai trees, or how to delegate a lattice basis. J. Cryptol. **25**(4), 601–639 (2012)
9. Catalano, D., Fiore, D., Nizzardo, L.: On the security notions for homomorphic signatures. In: Applied Cryptography and Network Security - 16th International Conference, ACNS 2018, Leuven, Belgium, July 2-4, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10892, pp. 183–201. Springer (2018)
10. Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., Stehlé, D.: CRYSTALS - dilithium: Digital signatures from module lattices. IACR Cryptol. ePrint Arch. p. 633 (2017)
11. Fiore, D., Mitrokotsa, A., Nizzardo, L., Pagnin, E.: Multi-key homomorphic authenticators. IET Inf. Secur. **13**(6), 618–638 (2019)
12. Freeman, D.M.: Improved security for linearly homomorphic signatures: A generic framework. In: Public Key Cryptography - PKC 2012 Proceedings. Lecture Notes in Computer Science, vol. 7293, pp. 697–714. Springer (2012)
13. Gennaro, R., Wichs, D.: Fully homomorphic message authenticators. In: Advances in Cryptology-ASIACRYPT 2013: Proceedings, Part II 19. pp. 301–320. Springer (2013)
14. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008. pp. 197–206. ACM (2008)
15. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In: Advances in Cryptology - CRYPTO 2013 Proceedings, Part I. Lecture Notes in Computer Science, vol. 8042, pp. 75–92. Springer (2013)
16. Gorbunov, S., Vaikuntanathan, V., Wichs, D.: Leveled fully homomorphic signatures from standard lattices. In: Proceedings of the forty-seventh annual ACM symposium on Theory of computing. pp. 469–477 (2015)
17. Hiromasa, R., Manabe, Y., Okamoto, T.: Homomorphic signatures for polynomial functions with shorter signatures. In: The 30th symposium on cryptography and information security, Kyoto (2013)
18. Katsumata, S., Yamada, S.: Group signatures without nizk: from lattices in the standard model. In: Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38. pp. 312–344. Springer (2019)

19. Lai, R.W., Tai, R.K., Wong, H.W., Chow, S.S.: Multi-key homomorphic signatures unforgeable under insider corruption. In: Advances in Cryptology–ASIACRYPT 2018: Proceedings, Part II. pp. 465–492. Springer (2018)
20. Langlois, A., Stehlé, D.: Worst-case to average-case reductions for module lattices. Designs, Codes and Cryptography **75**(3), 565–599 (2015)
21. Libert, B., Ling, S., Nguyen, K., Wang, H.: Zero-knowledge arguments for lattice-based accumulators: Logarithmic-size ring signatures and group signatures without trapdoors. In: Advances in Cryptology - EUROCRYPT 2016 - Proceedings, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 1–31. Springer (2016)
22. Ling, S., Nguyen, K., Wang, H.: Group signatures from lattices: Simpler, tighter, shorter, ring-based. In: Public-Key Cryptography - PKC 2015, Proceedings. Lecture Notes in Computer Science, vol. 9020, pp. 427–449. Springer (2015)
23. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Constant-size group signatures from lattices. In: Public-Key Cryptography - PKC 2018 - 21st IACR International Conference on Practice and Theory of Public-Key Cryptography, Rio de Janeiro, Brazil, March 25-29, 2018, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10770, pp. 58–88. Springer (2018)
24. Ling, S., Nguyen, K., Wang, H., Xu, Y.: Lattice-based group signatures: Achieving full dynamicity (and deniability) with ease. Theor. Comput. Sci. **783**, 71–94 (2019)
25. Luo, F., Wang, F., Wang, K., Chen, K.: A more efficient leveled strongly-unforgeable fully homomorphic signature scheme. Information Sciences **480**, 70–89 (2019)
26. Micciancio, D., Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller. In: Advances in Cryptology - EUROCRYPT 2012 Proceedings. Lecture Notes in Computer Science, vol. 7237, pp. 700–718. Springer (2012)
27. Micciancio, D., Regev, O.: Worst-case to average-case reductions based on gaussian measures. SIAM Journal on Computing **37**(1), 267–302 (2007)
28. Pan, J., Wagner, B.: Short identity-based signatures with tight security from lattices. In: Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Proceedings. Lecture Notes in Computer Science, vol. 12841, pp. 360–379. Springer (2021)
29. del Pino, R., Lyubashevsky, V., Seiler, G.: Lattice-based group signatures and zero-knowledge proofs of automorphism stability. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018. pp. 574–591. ACM (2018)
30. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM) **56**(6), 1–40 (2009)