Tightly Secure Linearly Homomorphic Signature Schemes for Subspace Under DL Assumption in AGM

Hao Huang[1], Xiaofen Wang[1](✉), Ke Zhang[1], Man Ho Au[2], Sheng Cao[1], Qinglin Zhao[3], and Xiaosong Zhang[1]

[1] University of Electronic Science and Technology of China, Chengdu, China
hh2357@std.uestc.edu.cn,{xfwang,kezhang,caosheng,johnsonzxs}@uestc.edu.cn
[2] The Hong Kong Polytechnic University, Hung Hom, China. mhaau@polyu.edu.hk
[3] Macau University of Science and Technology, Macau, China. qlzhao@must.edu.mo

Abstract. Linearly homomorphic signature (LHS) allows to perform any linear combination on the already signed vectors so that it is widely applied in various scenarios. Unfortunately, the existing LHS schemes do not have tight security proof based on the hardest problem in the cyclic group setting, namely, the discrete logarithm (DL) problem, and they all rely on computationally expensive pairing operations over bilinear groups. Recently, the proposal of the algebraic group model (AGM) enables some ordinary signature schemes to be tightly reduced to the DL problem. However, it remains unknown whether LHS schemes can be proven as secure as the DL problem in the AGM. In this paper, we revisit the security of Boneh's LHS scheme and first give a tight security proof under the DL assumption in the AGM. To improve computational efficiency, we propose a pairing-free linearly homomorphic signature (PF-LHS) scheme and a map-to-point hash function-free linearly homomorphic signature (MF-LHS) scheme, and the latter is given a tight security proof under the DL assumption in the AGM. Finally, the efficiency comparisons show that both our PF-LHS scheme and MF-LHS scheme are computationally efficient.

Keywords: Linearly homomorphic signature · Provable security · Tight reduction · Algebraic group model.

1 Introduction

Linearly homomorphic Signature with Ideal Security. Digital signatures [13] provide authenticity, integrity, and non-repudiation of messages. Linearly homomorphic signature (LHS) is a class of special digital signatures, which allows any entity to perform any linear combination on the signed messages without the signer's private key. More specifically, in a linearly homomorphic signature scheme, given the corresponding m signatures of m vectors $\mathbf{v_1}, \cdots, \mathbf{v}_m$ defined over some field $\mathbb{F}_p^n$, the combine algorithm allows anyone to produce a signature on any vector $\mathbf{v}$ in the linear subspace $V = span\{\mathbf{v}_1, \cdots, \mathbf{v}_m\}$. This property

makes LHS effective in preventing pollution attacks in the network coding routing mechanisms [8, 11], proofs of storage [2], cloud computing area [21], and verifiable computation on outsourced data [1].

LHS for subspace is secure if it is difficult to produce a signature of any vector $\mathbf{v} \notin V$. A more comprehensive security model of LHS is given in Section 2.5. When proving the security of a cryptographic scheme, the security reduction is tight if security loss L (i.e., reduction loss) is a constant or a small number that grows only (as a preferably small function) in the security parameter. Security loss is a crucial factor in determining the security parameter's size in practice. Increasing the size of the security parameter can compensate for the security loss, but degrades the efficiency of the scheme. Moreover, this process is not trivial. For example, considering bilinear groups, its security in source and target groups should be balanced, and computational efficiency is influenced by the choice of curves, pairings, and parameters such as embedding degrees, and some dedicated techniques. The security of a cryptographic scheme also depends on the difficulty of the underlying problem. The harder the underlying problem provides the stronger the security for the scheme becomes. For the cyclic groups setting, it is known that the discrete logarithm (DL) problem is the hardest one because algorithms that solve the DL problem can be used to solve the computational Diffie-Hellman (CDH) problem and its variants [17, 24]. Therefore, LHS schemes with ideal security, i.e., as secure as the DL problem in the security model of LHS for subspace [8], offer the strongest security assurance in the context of cyclic groups.

The Algebraic Group Model. The algebraic group model (AGM) was proposed by Fuchsbauer, Kiltz, and Loss [15], it is a computational model in which all adversaries are modeled as algebraic. A more detailed definition is given in Section 2.2. In the AGM, we can analyze the security of a scheme by giving security reductions from computational hardness assumptions.

Because of the algebraic nature of the adversary's forgeries, the AGM has recently been used to analyze blind (Schnorr) signatures [16], which are notoriously difficult to prove secure in the standard or random oracle model (ROM) [6]. Furthermore, variants of secure signature schemes [4,5,12,18,19,23,25] have been analyzed in the AGM. However, the security of LHS in the AGM has not yet been studied.

1.1 Related work

Boneh et al. [8] first introduced the formal definition and security model of linearly homomorphic signature for subspace, and they constructed an LHS scheme which is secure under the co-computational Diffie-Hellman (co-CDH) assumption in the ROM. In subsequent work, Attrapadung et al. [3] proposed the first LHS scheme over bilinear groups of composite order [10, 20] without random oracles, using the dual system techniques of Waters [30] to carry on the security proof. To improve efficiency, Catalano et al. [11] proposed an LHS scheme for subspace without random oracles that is secure under a strong assumption, namely, the q-Strong Diffie Hellman (q-SDH) assumption [7]. Freeman et al. [14] presented

a generic framework which converts regular signature schemes to linearly homomorphic signature schemes. For this generic construction, the security of the LHS scheme follows the same computational assumptions used to prove the security of the underlying signature scheme which is under the CDH assumption or the q-SDH assumption in the context of cyclic groups. After that, Schabhüser et al. [26] introduced a new LHS scheme based on the CDH assumption, without using a chameleon hash function thereby solving the problem introduced in [14]. Recently, Li et al. [22] designed a structure-preserving linearly homomorphic signature scheme with the designated combiner for subspace, which is based on the construction proposed by Boneh et al. [8]. Unfortunately, it is also based on the CDH assumption and the security reduction is not tight. At present, we find that for the context of cyclic groups, there is no LHS scheme that can be tightly reduced to the DL problem. Furthermore, all the above schemes rely on computationally expensive pairing operations over bilinear groups.

1.2 Our contribution

In this paper, we show that LHS schemes have tight security proof under the DL assumption in the AGM. The contributions are summarized as follows.

- We consider the security of Boneh et al.'s LHS scheme [8] (i.e., the BLS-LHS scheme), which is classically and efficiently constructed based on bilinear pairing. For the first time, we give a tight security proof under the DL assumption in the AGM with a random oracle.
- To improve computational efficiency, we propose a pairing-free linearly homomorphic signature (PF-LHS) scheme and a map-to-point (MTP) hash function-free linearly homomorphic signature (MF-LHS) scheme, and give a tight security proof under the DL assumption in the AGM with a random oracle for the MF-LHS scheme. Efficiency comparison shows that both the PF-LHS scheme and the MF-LHS scheme have lower computational overhead than the BLS-LHS scheme.

1.3 Organization

The rest of this paper is organized as follows. In Section 2, we describe some preliminaries used in this paper. In Section 3, we revisit Boneh's scheme and give a tight security proof under the DL assumption in the AGM with a random oracle. In Section 4, we present our PF-LHS scheme and MF-LHS scheme, the latter is tightly reduced to the DL problem in the AGM. Subsequently, we give the efficiency analysis in Section 5. Finally, conclusions are given in Section 6.

2 Preliminaries

2.1 Bilinear Groups and Complexity Assumption

Definition 1. *Let p be a large prime, $\mathbb{G}_1, \mathbb{G}_T$ be cyclic groups of prime order p, and $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ be a bilinear map. A tuple of $(\mathbb{G}_1, \mathbb{G}_T, p, e)$ is defined as a bilinear group tuple which has the following properties:*

- *Computability:* $\forall\ g, h \in \mathbb{G}_1$, $e(g, h)$ *can be efficiently computed.*
- *Bilinearity:* $\forall\ g, h \in \mathbb{G}_1$, *and* $\forall\ a, b \in \mathbb{Z}_p$, $e(g^a, h^b) = e(g, h)^{ab}$.
- *Non-degeneracy: if* g, h *are generators of* $\mathbb{G}_1$, $e(g, h)$ *is a generator of* $\mathbb{G}_T$.

Definition 2 (Discrete Logarithm Assumption). *Assume* $\mathbb{G}$ *is a cyclic group of order* p*, where* p *is a large prime.* $g \in \mathbb{G}$ *is a generator. Given* (g, g^a) *as input, where* g^a *is an random element in* $\mathbb{G}$*, output* $a \in \mathbb{Z}_p$*. The discrete logarithm (DL) assumption holds in* $\mathbb{G}$ *if for any probabilistic polynomial time (PPT) adversary* $\mathcal{A}$*,*

$$Pr[\mathcal{A}(1^\lambda, g, g^a) \to a] \leq negl(\lambda)$$

holds for arbitrary security parameter λ*, where* $negl(\cdot)$ *is a negligible function.*

2.2 The Algebraic Algorithms

The adversary's computation is considered as algebraic in the AGM. The algebraic algorithm is defined as follows.

Definition 3 (Algebraic Algorithm). *Suppose the adversary in the security game is algebraic, and it is given* $(\mathbf{X_0}, \mathbf{X_1}, \cdots, \mathbf{X_n}) \in \mathbb{G}^{n+1}$*. Without loss of generality, assume that* $\mathbf{X}_0 = g \in \mathbb{G}$ *is the group generator. Whenever the adversary outputs* $\mathbf{Z} \in \mathbb{G}$*, it attaches a representation vector* $\boldsymbol{\pi} = (\pi_0, \pi_1, \cdots, \pi_n) \in \mathbb{Z}_q^{n+1}$ *such that* $\mathbf{Z} = \prod_{i=0}^{n} \mathbf{X}_i^{\pi_i}$*, which indicates how* $\mathbf{Z}$ *is generated based on the given group elements.*

2.3 The Augmented Basis Vectors

In the LHS scheme [8], a file is viewed as an ordered sequence of n-dimensional vectors $\bar{\mathbf{v}}_1, \cdots, \bar{\mathbf{v}}_m \in \mathbb{F}_p^n$. In order to derive linear combination coefficients from a combined vector and guarantee that these vectors are linearly independent, the signer creates the augmented basis vectors $\mathbf{v}_1, \cdots, \mathbf{v}_m$, where $\mathbf{v}_i = (\bar{\mathbf{v}}_i, \underbrace{0, \cdots, 0}_{i}, \overbrace{1, 0, \cdots, 0}^{m}) \in \mathbb{F}_p^N$ for $i \in \{1, \cdots, m\}$ and $N = n + m$. The augmented vectors $\mathbf{v}_1, \cdots, \mathbf{v}_m$ are viewed as basis vectors of the subspace $V \subset \mathbb{F}_p^N$.

2.4 Definition of Linearly Homomorphic Signature

Definition 4. *A linearly homomorphic signature scheme is defined by the following five algorithms.*

- **Setup**$(1^\lambda, N) \to$ *params*: taking the security parameters 1^λ and an integer N (i.e., dimension of the signed vector) as input, this algorithm outputs the public parameters *params* that contain a prime field $\mathbb{F}_p$ over which vectors and linear combinations are defined.

- **KeyGen**$(params) \rightarrow (pk, sk)$: taking the public parameters $params$ as input, this algorithm outputs a private key sk and the corresponding public key pk.
- **Sign**$(params, sk, id, \mathbf{v}) \rightarrow \sigma$: taking the public parameters $params$, a secret key sk, a subspace identifier $id \in \{0,1\}^\lambda$, and a vector $\mathbf{v} \in \mathbb{F}_p^N$ as input, this algorithm outputs a signature σ. To sign a subspace $V = span\{\mathbf{v}_1, \cdots, \mathbf{v}_m\}$, the signer needs to perform **Sign**$(params, sk, id, \mathbf{v}_i) \rightarrow \sigma_i$ for all $i = 1$ to m.
- **Combine**$(params, pk, id, \{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l) \rightarrow (\mathbf{v}, \sigma)$: taking the public parameters $params$, a public key pk, a subspace identifier id, and a set of tuples $\{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l$ with $\beta_i \in \mathbb{F}_p$ as input, this algorithm outputs a vector/signature pair $(\mathbf{v}, \sigma)$. If σ_i is a valid signature of each vector $\mathbf{v}_i$, σ is a signature of the combined vector $\mathbf{v} = \sum_{i=1}^l \beta_i \mathbf{v}_i$.
- **Verify**$(params, pk, id, \mathbf{v}, \sigma) \rightarrow 0$ or 1: taking the public parameters $params$, a public key pk, an identifier id of subspace V, a vector $\mathbf{v} \in \mathbb{F}_p^N$, and a signature σ as input, this algorithm outputs either 0 (reject) or 1 (accept).

Correctness. The correctness of a linearly homomorphic signature scheme holds iff for each valid key pair (pk, sk) generated from **Setup** and **KeyGen**, the following conditions are satisfied:

- $\forall\ id \in \{0,1\}^\lambda$ and $\forall\ \mathbf{v} \in \mathbb{F}_p^N$, if $\sigma \leftarrow$ **Sign**$(params, sk, id, \mathbf{v})$, then

$$\textbf{Verify}(params, pk, id, \mathbf{v}, \sigma) = 1.$$

- $\forall\, id \in \{0,1\}^\lambda$ and $\{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l$, if $\forall\, i \in \{1, \cdots, l\}$, **Verify**$(params, pk, id, \mathbf{v}_i, \sigma_i) = 1$ holds, then

$$\textbf{Verify}(params, pk, id, \textbf{Combine}(params, pk, id, \{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l)) = 1.$$

2.5 Security Model for LHS

The security model of unforgeability against chosen subspace attacks for LHS schemes is defined between a challenger and an adversary as follows.

- **Initialization**: The challenger runs **Setup** algorithm to generate the public parameters $params$ and **KeyGen** algorithm to generate (sk, pk). Then the challenger initializes the adversary with $(params, pk)$.
- **Sign Query**: The adversary adaptively queries for a signature of any subspace. When making the l-th sign query for a vector subspace $V_l \subset \mathbb{F}_p^N$ described by properly augmented basis vectors $\mathbf{v}_{l,1}, \cdots, \mathbf{v}_{l,m}$, where $N = n + m$, the challenger randomly chooses $id_l \in \{0,1\}^\lambda$, and returns id_l and $\sigma_{l,j} \leftarrow$ **Sign**$(params, sk, id_l, \mathbf{v}_{l,j})$ to the adversary for all $j = 1$ to m.
- **Forgery**: The adversary outputs $id^* \in \{0,1\}^\lambda$, a non-zero vector $\mathbf{y}^*$, and a forged signature σ^*.

The adversary wins the above game if **Verify**$(params, pk, id^*, \mathbf{y}^*, \sigma^*) = 1$, and either (1) $id^* \neq id_l$ for all l (a type 1 forgery) or (2) $id^* = id_l$ for some l but $y^* \notin V_l$ (a type 2 forgery).

Definition 5. *An LHS signature scheme is unforgeable against chosen subspace attacks if any probability polynomial time (PPT) adversary wins the above security game with a negligible advantage.*

3 Tightly Secure BLS-LHS Scheme

In this section, we review Boneh et al. LHS scheme [8], and give a tight security reduction for it in the AGM.

3.1 The BLS-LHS Scheme

The BLS-LHS scheme is based on NCS_1 proposed by Boneh et al. [8], which is modeled on the BLS signature [9]. A precise description of the BLS-LHS scheme is given below.

- **Setup**$(1^\lambda, N)$. Given the security parameter 1^λ and a positive integer N, this algorithm works as follows:
  1. Generate a bilinear group tuple $(\mathbb{G}_1, \mathbb{G}_T, p, e)$ such that $p > 2^\lambda$. Choose generators $g, g_1, \cdots, g_n \in \mathbb{G}_1 \setminus \{1\}$ randomly.
  2. Let $H : \{0,1\}^* \to \mathbb{G}_1$ be a secure hash function.
  3. Output the public parameters $params = (\mathbb{G}_1, \mathbb{G}_T, p, e, g, g_1, \cdots, g_n, H)$.
- **KeyGen**$(params)$. Given the public parameters $params$, the user randomly picks $x \in \mathbb{F}_p^*$ as the private key sk and computes $pk = g^x$ as the corresponding public key.
- **Sign**$(params, sk, id, \mathbf{v})$. Given the public parameters $params$, a user's private key $sk = x$, an identifier $id \in \{0,1\}^\lambda$, and a vector $\mathbf{v} \in \mathbb{F}_p^N$, this algorithm outputs a signature $\sigma = (\prod_{j=1}^n g_j^{v_j} \cdot \prod_{i=1}^m H(id, i)^{v_{n+i}})^x$.
- **Combine**$(params, pk, id, \{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l)$. Given the public parameters $params$, a user's public key pk, an identifier id, and a set of tuples $\{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l$ with $\beta_i \in \mathbb{F}_p$, this algorithm outputs a vector $\mathbf{v} = \sum_{i=1}^l \beta_i \mathbf{v}_i$ and a signature $\sigma = \prod_{i=1}^l \sigma_i^{\beta_i}$.
- **Verify**$(params, pk, id, \mathbf{v}, \sigma)$. Given the public parameters $params$, a user's public key pk, an identifier id, a vector $\mathbf{v} \in \mathbb{F}_p^N$, and a signature σ, if

$$e(\sigma, g) = e(\prod_{j=1}^n g_j^{v_j} \cdot \prod_{i=1}^m H(id, i)^{v_{n+i}}, pk),$$

  this algorithm outputs 1 (accept). Otherwise, it outputs 0 (reject).

3.2 Security Proof

Theorem 1. *If the DL problem is hard, the BLS-LHS scheme is unforgeable against chosen subspace attacks in the AGM with a random oracle with a loss factor of 2.*

Proof. Suppose there exists an algebraic adversary $\mathcal{A}$ who can break the BLS-LHS scheme in the security model defined by Section 2.5 with an advantage ϵ. Then we can construct simulators to solve the DL problem. The hash function H is modeled as a random oracle simulated by simulators, and q_h and q_s are the number of **Hash Queries** and the number of **Sign Queries** respectively. First, we flip a coin $\phi \in \{0,1\}$. If $\phi = 0$, we construct $\mathcal{C}_1$ to run **Simulation-1**. Otherwise, $\phi = 1$, we construct $\mathcal{C}_2$ to run **Simulation-2**. Note that each simulation runs with a probability $\frac{1}{2}$.

Simulation-1. Given a random instance (g, g^x) of the DL problem over the bilinear groups $(\mathbb{G}_1, \mathbb{G}_T, p, e, g)$, $\mathcal{C}_1$ interacts with $\mathcal{A}$ as follows.

Initialization. $\mathcal{C}_1$ randomly chooses $r_1, \cdots, r_n \in \mathbb{F}_p^*$, computes $g_i = g^{r_i}$ for all $i \in \{1, \cdots, n\}$, and initializes $\mathcal{A}$ with $params = (\mathbb{G}_1, \mathbb{G}_T, p, e, g, g_1, \cdots, g_n, H)$ and $pk = g^x$.

Hash Query. $\mathcal{C}_1$ keeps a list L_H which is initially empty. L_H is utilized to record H hash queries. When $\mathcal{A}$ requests the value of $H(id_l, i)$, where $i \in \{1, 2, \cdots, m\}$, $\mathcal{C}_1$ does the following steps:

- If (id_l, i) exists in L_H, return the corresponding $g^{h_{l,i}} = H(id_l, i)$ to $\mathcal{A}$.
- If (id_l, i) does not exist in L_H, randomly select $h_{l,i} \in \mathbb{F}_p^*$, return $g^{h_{l,i}}$ to $\mathcal{A}$, and add $(id_l, i, g^{h_{l,i}})$ to L_H.

Sign Query. When $\mathcal{A}$ requests a signature of a vector subspace $V_l \subset \mathbb{F}_p^N$ described by properly augmented basis vectors $\mathbf{v}_{l,1}, \cdots, \mathbf{v}_{l,m} \in \mathbb{F}_p^N$, where $\mathbf{v}_{l,i} = (v_{l,i,1}, \cdots, v_{l,i,n}, \overbrace{\underbrace{0, \cdots, 0, 1}_{i}, 0, \cdots, 0}^{m})$ for $i = 1$ to m, $\mathcal{C}_1$ chooses a random $id_l \in \{0,1\}^\lambda$ and does the following steps for all $i = 1$ to m:

1. If $H(id_l, i)$ has not been queried then make a **Hash Query** for it.
2. Compute a signature $\sigma_{l,i} = (\prod_{j=1}^n g_j^{v_{l,i,j}} \cdot H(id_l, i))^x = (g^x)^{\sum_{j=1}^n r_j v_{l,i,j} + h_{l,i}}$.

$\mathcal{C}_1$ outputs id_l and $\sigma_l = (\sigma_{l,1}, \cdots, \sigma_{l,m})$.

Forgery. $\mathcal{A}$ outputs $id^* \in \{0,1\}^\lambda$, a non-zero vector $\mathbf{y}^* \in \mathbb{F}_p^N$, and a forged signature $\sigma^* = (\prod_{j=1}^n g_j^{y_j^*} \cdot \prod_{i=1}^m H(id^*, i)^{y_{n+i}^*})^x$, where $H(id^*, i) = g^{h_i^*}$. It also attaches together with some representation vectors $[\boldsymbol{a}], [\boldsymbol{b}], [\boldsymbol{c}]$ in $\mathbb{F}_p$, such that

$$[\boldsymbol{a}] = (a_0, a_1, a_{2,1}, \cdots, a_{2,n}), \quad [\boldsymbol{c}] = (c_1, \cdots, c_{q_h}),$$

$$[\boldsymbol{b}] = (b_{1,1}, b_{1,2}, \cdots, b_{1,m}, \cdots, b_{q_s,1}, b_{q_s,2}, \cdots, b_{q_s,m}),$$

which indicate how the forgery is algebraically computed, i.e.,

$$\sigma^* = g^{a_0}(g^x)^{a_1} \prod_{j=1}^{n} g_j^{a_{2,j}} \prod_{j=1}^{q_s} \prod_{i=1}^{m} \sigma_{j,i}^{b_{j,i}} \prod_{j=1}^{q_h} H_j^{c_j}.$$

In order to simplify the above equation, let $\sigma_{j,i} = g^{xe_{j,i}}$, $e_{j,i} = \sum_{k=1}^{n} r_k v_{j,i,k} + h_{j,i}$, $H_i = g^{h_i}$. Based on the above equations, it is easy to see

$$x(\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^*) = a_0 + a_1 x + \sum_{j=1}^{n} r_j a_{2,j} + x \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i} + \sum_{j=1}^{q_h} c_j h_j,$$

$$x(\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^* - a_1 - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i}) = a_0 + \sum_{j=1}^{n} r_j a_{2,j} + \sum_{j=1}^{q_h} c_j h_j.$$

Depending on whether $(\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^* - a_1 - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i}) = 0$, we consider the following two cases in this simulation.

In **Case 1**, $(\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^* - a_1 - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i}) \neq 0$. We assume that $\mathcal{A}$ outputs a forgery in **Case 1** with an advantage ϵ_1. No matter $\mathcal{A}$ outputs a type 1 forgery or a type 2 forgery, $\mathcal{C}_1$ can solve the DL problem by computing

$$x = \frac{a_0 + \sum_{j=1}^{n} r_j a_{2,j} + \sum_{j=1}^{q_h} c_j h_j}{\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^* - a_1 - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i}}.$$

In **Case 2**, $(\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^* - a_1 - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i}) = 0$. We assume that $\mathcal{A}$ outputs a forgery in **Case 2** with an advantage ϵ_2. In this case, $\mathcal{C}_1$ aborts. Obviously, $\epsilon_1 + \epsilon_2 = \epsilon$.

There is no abort in the simulation as $\mathcal{C}_1$ manages to respond to every query. The probability of successful simulation and useful attack is ϵ_1 in **Simulation-1**.

Simulation-2. Given a random instance (g, g^x) of the DL problem over the bilinear groups $(\mathbb{G}_1, \mathbb{G}_T, p, e, g)$, $\mathcal{C}_2$ interacts with $\mathcal{A}$ as follows.

Initialization. $\mathcal{C}_2$ randomly chooses $\eta_1, \cdots, \eta_n, \mu_1, \cdots, \mu_n, z \in \mathbb{F}_p^*$, computes $g_i = (g^x)^{\eta_i} \cdot g^{\mu_i}$ for all $i \in \{1, \cdots, n\}$ and $pk = g^z$, and initializes $\mathcal{A}$ with $params = (\mathbb{G}_1, \mathbb{G}_T, p, e, g, g_1, \cdots, g_n, H)$ and pk.

Hash Query. $\mathcal{C}_2$ keeps a list L_H which is initially empty. L_H is utilized to record H hash queries. When $\mathcal{A}$ requests the value of $H(id_l, i)$, where $i \in \{1, \cdots, m\}$, $\mathcal{C}_2$ does the following steps:

- If (id_l, i) exists in L_H, return the corresponding $H(id_l, i) = (g^x)^{s_{l,i}} \cdot g^{t_{l,i}}$ to $\mathcal{A}$.
- If (id_l, i) does not exist in L_H, randomly select $s_{l,i}, t_{l,i} \in \mathbb{F}_p^*$, return $(g^x)^{s_{l,i}} \cdot g^{t_{l,i}}$ to $\mathcal{A}$, and add $(id_l, i, (g^x)^{s_{l,i}} \cdot g^{t_{l,i}})$ to L_H.

Sign Query. When $\mathcal{A}$ requests a signature of a vector subspace $V_l \subset \mathbb{F}_p^N$ described by properly augmented basis vectors $\mathbf{v}_{l,1}, \cdots, \mathbf{v}_{l,m} \in \mathbb{F}_p^N$, where $\mathbf{v}_{l,i} = (v_{l,i,1}, \cdots, v_{l,i,n}, \overbrace{\underbrace{0, \cdots, 0, 1}_{i}, 0, \cdots, 0}^{m})$ for $i = 1$ to m, $\mathcal{C}_2$ chooses a random $id_l \in \{0,1\}^\lambda$ and does the following steps for all $i \in \{1, \cdots, m\}$:

1. If $H(id_l, i)$ has not been queried, then make a **Hash Query** for it.

2. Compute a signature $\sigma_{l,i} = (\prod_{j=1}^{n} g_j^{v_{l,i,j}} \cdot H(id_l, i))^z$.

$\mathcal{C}_2$ outputs id_l and $\sigma_l = (\sigma_{l,1}, \cdots, \sigma_{l,m})$.

Forgery. $\mathcal{A}$ outputs $id^* \in \{0,1\}^\lambda$, a non-zero vector $\mathbf{y}^* \in \mathbb{F}_p^N$, and a forged signature $\sigma^* = (\prod_{j=1}^{n} g_j^{y_j^*} \cdot \prod_{i=1}^{m} H(id^*, i)^{y_{n+i}^*})^z$, where $H(id^*, i) = g^{h_i^*} = g^{xs_i^* + t_i^*}$. It also attaches some representation vectors $[\boldsymbol{a}], [\boldsymbol{b}], [\boldsymbol{c}]$ in $\mathbb{F}_p$ such that

$$[\boldsymbol{a}] = (a_0, a_1, a_{2,1}, \cdots, a_{2,n}),\ [\boldsymbol{c}] = (c_1, \cdots, c_{q_h}),$$

$$[\boldsymbol{b}] = (b_{1,1}, b_{1,2}, \cdots, b_{1,m}, \cdots, b_{q_s,1}, b_{q_s,2}, \cdots, b_{q_s,m}),$$

which indicates how the forgery is algebraically computed, i.e.,

$$\sigma^* = g^{a_0} g^{z a_1} \prod_{j=1}^{n} g_j^{a_{2,j}} \prod_{j=1}^{q_s} \prod_{i=1}^{m} \sigma_{j,i}^{b_{j,i}} \prod_{j=1}^{q_h} H_j^{c_j}.$$

Depending on whether $(\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^* - a_1 - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i}) = 0$ where $r_k = x\eta_k + \mu_k,\ e_{j,i} = \sum_{k=1}^{n} r_k v_{j,i,k} + h_{j,i},\ h_{j,i} = xs_{j,i} + t_{j,i}$, we consider two cases that are the same as those of **Simulation-1**.

In **Case 1**, $(\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^* - a_1 - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i}) \neq 0$. In this case, $\mathcal{C}_2$ aborts.

In **Case 2**, $(\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^* - a_1 - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i}) = 0$. It happens with an advantage ϵ_2. Since $\mathcal{C}_2$ embeds the DL problem instance g^x to each **Hash Query** and g_j $(1 \leq j \leq n)$, the following equation holds

$$x(\sum_{k=1}^{n} \eta_k y_k^* + \sum_{i=1}^{m} s_i^* y_{n+i}^* - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} s_{j,i} - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} \sum_{k=1}^{n} \eta_k v_{j,i,k}) = a_1 - \sum_{k=1}^{n} \mu_k y_k^* -$$

$$\sum_{i=1}^{m} t_i^* y_{n+i}^* + \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} (t_{j,i} + \sum_{k=1}^{n} \mu_k v_{j,i,k}).$$

If $\mathcal{A}$ outputs a type 1 forgery such that $h_i^* \neq h_{j,i}$ (i.e., $s_i^* \neq s_{j,i}$) for $\forall i \in [1, m]$ and $\forall j \in [1, q_s]$, $\mathcal{C}_2$ can solve the DL problem by computing

$$x = \frac{a_1 - \sum_{k=1}^{n} \mu_k y_k^* - \sum_{i=1}^{m} t_i^* y_{n+i}^* + \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} (t_{j,i} + \sum_{k=1}^{n} \mu_k v_{j,i,k})}{\sum_{k=1}^{n} \eta_k y_k^* + \sum_{i=1}^{m} s_i^* y_{n+i}^* - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} s_{j,i} - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} \sum_{k=1}^{n} \eta_k v_{j,i,k}}.$$

If $\mathcal{A}$ outputs a type 2 forgery such that $h_i^* = h_{j',i}$ (i.e., $s_i^* = s_{j',i}$) for some j' and $1 \leq i \leq m$, $\mathcal{C}_2$ computes $\mathbf{v}^* = \mathbf{y}^* - \sum_{i=1}^{m} y_{n+i}^* \mathbf{v}_{j',i}$ $(\mathbf{v}^* \neq 0^{n+m})$. From the above equations, $\mathcal{C}_2$ can solve the DL problem by computing

$$\sum_{k=1}^{n} r_k y_k^* + \sum_{i=1}^{m} h_i^* y_{n+i}^* - a_1 - \sum_{j=1}^{q_s} \sum_{i=1}^{m} b_{j,i} e_{j,i} - \sum_{i=1}^{m} y_{n+i}^* e_{j',i} + \sum_{i=1}^{m} y_{n+i}^* e_{j',i} = 0,$$

$$\sum_{k=1}^{n} r_k v_k^* - a_1 - \sum_{j=1}^{q_s}\sum_{i=1}^{m} b_{j,i} \sum_{k=1}^{n} r_k v_{j,i,k} - \sum_{j=1}^{q_s}\sum_{i=1}^{m} b_{j,i} h_{j,i} + \sum_{i=1}^{m} y_{n+i}^* e_{j',i} = 0,$$

$$x = \frac{a_1 - \sum_{k=1}^{n} \mu_k v_k^* + \sum_{j=1}^{q_s}\sum_{i=1}^{m} b_{j,i}(\sum_{k=1}^{n} \mu_k v_{j,i,k} + t_{j,i}) - \sum_{i=1}^{m} y_{n+i}^*(\sum_{k=1}^{n} \mu_k v_{j',i,k} + t_{j',i})}{\sum_{k=1}^{n} \eta_k(v_k^* - \sum_{j=1}^{q_s}\sum_{i=1}^{m} b_{j,i} v_{j,i,k} + \sum_{i=1}^{m} y_{n+i}^* v_{j',i,k}) - \sum_{j=1}^{q_s}\sum_{i=1}^{m} b_{j,i} s_{j,i} + \sum_{i=1}^{m} y_{n+i}^* s_{j',i}}.$$

Suppose $s_k^* \neq s_{j,i}$ for $\forall k, i \in [1, m]$ and $\forall j \in [1, q_s]$, we next analyze the probability of solving the DL problem in **Simulation-2**. Since the variables $\eta_1, \cdots, \eta_n, s_1^*, \cdots, s_m^*, s_{1,1}, \ldots, s_{1,m}, \cdots, s_{q_s,1}, \cdots, s_{q_s,m}$ are each independently uniform in $\mathbb{F}_p^*$ even conditioned on $\mathcal{A}$'s view, the probability that either $\sum_{k=1}^{n} \eta_k y_k^* + \sum_{i=1}^{m} s_i^* y_{n+i}^* - \sum_{j=1}^{q_s}\sum_{i=1}^{m} b_{j,i} s_{j,i} - \sum_{j=1}^{q_s}\sum_{i=1}^{m} b_{j,i} \sum_{k=1}^{n} \eta_k v_{j,i,k} = 0$ or $\sum_{k=1}^{n} \eta_k(v_k^* - \sum_{j=1}^{q_s}\sum_{i=1}^{m} b_{j,i} v_{j,i,k} + \sum_{i=1}^{m} y_{n+i}^* v_{j',i,k}) - \sum_{j=1}^{q_s}\sum_{i=1}^{m} b_{j,i} s_{j,i} + \sum_{i=1}^{m} y_{n+i}^* s_{j',i} = 0$ is negligible. There is no abort in the simulation as $\mathcal{C}_2$ manages to respond to every query. Therefore, the probability of successful simulation and useful attack is ϵ_2 in **Simulation-2**.

Indistinguishable Simulations. According to **Simulation-1** and **Simulation-2**, simulators set all elements with perfectly hidden randomness. The public key pk, the public parameters $g_1, \cdots, g_n$, and hash value of $H(id_l, i)$ are simulated as follows

$$\begin{cases} pk = g^x, & g_j = g^{r_j}, \quad H(id_l, i) = g^{h_{l,i}}, & \mathcal{C}_1 \\ pk = g^z, & g_j = g^{x\eta_j + \mu_j}, \quad H(id_l, i) = g^{x s_{l,i} + t_{l,i}}, & \mathcal{C}_2 \end{cases}$$

where $x, r_j, h_{l,i}, z, \eta_j, \mu_j, s_{l,i}, t_{l,i}$ are all randomly chosen. Therefore, all elements are random and independent on $\mathcal{A}$'s view.

Reduction Loss. Since the two simulations are indistinguishable, we run one of the simulations based on $\phi \in \{0, 1\}$. The success probability of solving the DL problem is shown as follows,

$$\begin{aligned} Pr[Success] &= Pr[Success|\mathcal{C}_1]Pr[\phi = 0] + Pr[Success|\mathcal{C}_2]Pr[\phi = 1] \\ &= \epsilon_1 \cdot \frac{1}{2} + \epsilon_2 \cdot \frac{1}{2} = \frac{\epsilon}{2}. \end{aligned}$$

Therefore, the success probability of solving the DL problem is at least $\frac{\epsilon}{2}$, which concludes that the reduction loss of the BLS-LHS scheme is 2. This completes the proof.

4 Tightly Secure LHS Schemes

In this section, to improve computational efficiency we propose two LHS schemes, i.e. the PF-LHS scheme and the MF-LHS scheme, and give a tight security reduction for the MF-LHS scheme in the AGM.

4.1 The PF-LHS scheme

To reduce the computational cost introduced by bilinear pairing, we propose a concrete PF-LHS scheme, which is modeled on the Schnorr signature scheme [27, 28]. Below we give a precise description of the PF-LHS scheme.

- **Setup**$(1^\lambda, N)$. Given security parameter 1^λ and a positive integer N, this algorithm works as follows:
  1. Generate a cyclic group $\mathbb{G}$ of prime order p such that $p > 2^\lambda$. Choose a generator $g \in \mathbb{G} \setminus \{1\}$ randomly.
  2. Let $H_1 : \{0,1\}^* \to \mathbb{F}_p^*$ be a secure hash function.
  3. Output the public parameters $params = (\mathbb{G}, p, g, H_1)$.
- **KeyGen**$(params)$. Given the public parameters $params$, the user randomly picks $x_0, x_1, \cdots, x_N \in \mathbb{F}_p^{N+1}$, computes $g_j = g^{x_j} \in \mathbb{G}$ for all $j = 0$ to N, and sets the private key $sk = (x_0, x_1, \cdots, x_N)$ and the corresponding public key $pk = (g_0, g_1, \cdots, g_N)$.
- **Sign**$(params, sk, id, \mathbf{v})$. Given the public parameters $params$, a user's private key sk, an identifier id which is randomly chosen in $\{0,1\}^\lambda$, and a vector $\mathbf{v} \in \mathbb{F}_p^N$, this algorithm proceeds as follows.
  1. Pick a random $w \in \mathbb{F}_p^*$ and compute $W = g^w$.
  2. Compute $\delta = \frac{\sum_{j=1}^N v_j x_j}{x_0 H_1(W,id)+w}$.

  Finally, this algorithm outputs a signature $\sigma = (\delta, W)$. Note that to sign the vectors with the same id we use the same R.
- **Combine**$(params, pk, id, \{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l)$. Given the public parameters $params$, a user's public key pk, an identifier id, and a set of tuples $\{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l$ with $\beta_i \in \mathbb{F}_p$, this algorithm outputs a vector $\mathbf{v} = \sum_{i=1}^l \beta_i \mathbf{v}_i$ and a signature $\sigma = (\delta, W)$, where $\delta = \sum_{i=1}^l \beta_i \delta_i$ and $W = W_1 = \cdots = W_l$.
- **Verify**$(params, pk, id, \mathbf{v}, \sigma)$. Given the public parameters $params$, a user's public key pk, an identifier id, a vector $\mathbf{v} \in \mathbb{F}_p^N$, and a signature σ, if

$$(g_0^{H_1(W,id)} \cdot W)^\delta = \prod_{j=1}^N g_j^{v_j},$$

  this algorithm outputs 1 (accept). Otherwise, it outputs 0 (reject).

Correctness. $\forall\ id \in \{0,1\}^\lambda,\ \forall\ \mathbf{v} \in \mathbb{F}_p^N$, if $\sigma \leftarrow$ **Sign**$(params, sk, id, \mathbf{v})$, then

$$(g_0^{H_1(W,id)} \cdot W)^\delta = g^{\delta(x_0 H_1(W,id)+w)} = g^{\sum_{j=1}^N x_j v_j} = \prod_{j=1}^N g_j^{v_j}.$$

It is obvious that the combined vector and signature can be verified similarly.

In the AGM with a random oracle, we can achieve tight security proof for type 1 forgery of the PF-LHS scheme under the DL assumption. But type 2 forgery of the PF-LHS scheme cannot be tightly reduced to the DL problem[4].

[4] Suppose $H_1(W_i, id_i)$ is from the i-th sign query and $H_1(W^*, id^*)$ is from type 2 forgery. If $H_1(W^*, id^*) = H_1(W_i, id_i)$ (i.e., $W^* = W_i$ and $id^* = id_i$), the simulator cannot extract the solution of the DL problem in a simulation.

To consider a computationally efficient LHS scheme that can be tightly reduced to DL problem against both type 1 fogery and type 2 forgery, we further design the MF-LHS scheme.

Due to space limitations and the proof for type 1 forgery of the PF-LHS scheme is similar to that of the MF-LHS scheme, refer to Theorem 2 for details.

4.2 The MF-LHS scheme

To reduce the number of public keys in the PF-LHS scheme and avoid MTP hash function, which is a special type of probabilistic hash function resulting in low efficiency, we propose a concrete MF-LHS scheme. Below we give a precise description of the MF-LHS scheme.

- **Setup**$(1^\lambda, N)$. Given the security parameter 1^λ and a positive integer N, this algorithm works as follows:
  1. Generate a bilinear group tuple $(\mathbb{G}_1, \mathbb{G}_T, p, e)$ such that $p > 2^\lambda$. Choose generators $g, g_1, \cdots, g_N \in \mathbb{G}_1 \setminus \{1\}$ randomly.
  2. Let $H_1 : \{0,1\}^* \to \mathbb{F}_p^*$ be a secure hash function.
  3. Output the public parameters $params = (\mathbb{G}_1, \mathbb{G}_T, p, e, g, g_1, \cdots, g_N, H_1)$.
- **KeyGen**$(params)$. Given the public parameters $params$, the user randomly picks $x \in \mathbb{F}_p^*$, computes $X = g^x$, and sets the private key $sk = x$ and public key $pk = X$.
- **Sign**$(params, sk, id, \mathbf{v})$. Given the public parameters $params$, a user's private key sk, a random identifier $id \in \{0,1\}^\lambda$, and a vector $\mathbf{v} \in \mathbb{F}_p^N$, this algorithm picks a random $r \in \mathbb{F}_p^*$ and computes $R = g^r,\ \delta = (\prod_{j=1}^{N} g_j^{v_j})^{xH_1(id,R)+r}$. Finally, this algorithm outputs a signature $\sigma = (R, \delta)$. Note that to sign the vectors with the same id, the same R is used.
- **Combine**$(params, pk, id, \{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l)$. Given the public parameters $params$, a user's public key pk, an identifier id, and a set of tuples $\{(\beta_i, \mathbf{v}_i, \sigma_i)\}_{i=1}^l$, where $\sigma_i = (R_i, \delta_i)$ and $\beta_i \in \mathbb{F}_p$, this algorithm outputs a vector $\mathbf{v} = \sum_{i=1}^l \beta_i \mathbf{v}_i$ and a signature $\sigma = (R, \delta)$, where $R = R_1 = \cdots = R_l$ and $\delta = \prod_{i=1}^l \delta_i^{\beta_i}$.
- **Verify**$(params, pk, id, \mathbf{v}, \sigma)$. Given the public parameters $params$, a user's public key pk, an identifier id, a vector $\mathbf{v} \in \mathbb{F}_p^N$, and a signature $\sigma = (R, \delta)$, if $e(\delta, g) = e(\prod_{j=1}^N g_j^{v_j}, X^{H_1(id,R)} \cdot R)$, this algorithm outputs 1 (accept). Otherwise, it outputs 0 (reject).

Correctness. $\forall\ id \in \{0,1\}^\lambda,\ \forall\ \mathbf{v} \in \mathbb{F}_p^N$, if $\sigma \leftarrow$ **Sign**$(params, sk, id, \mathbf{v})$, then

$$e(\delta, g) = e((\prod_{j=1}^{N} g_j^{v_j})^{xH_1(id,R)+r}, g) = e(\prod_{j=1}^{N} g_j^{v_j}, X^{H_1(id,R)} \cdot R).$$

It is obvious that the combined vector and signature can be verified similarly.

Security Proof. In the AGM with a random oracle, we give a tight security proof for the MF-LHS scheme under the DL assumption.

Theorem 2. *If the DL problem is hard, the MF-LHS scheme is unforgeable against chosen subspace attacks in the AGM with a random oracle with a loss factor of* 3.

Proof. Suppose there exists an algebraic adversary $\mathcal{A}$ who can break the MF-LHS scheme in the security model defined by Section 2.5 with an advantage ϵ, and we can construct simulators $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ to solve the DL problem. $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ run **Simulation-1**, **Simulation-2**, and **Simulation-3** respectively, where the hash function H_1 is modeled as a random oracle. Note that each simulation is chosen at random with a probability of $\frac{1}{3}$.

Let q_h and q_s be the number of **Hash Queries** and the number of **Sign Queries** respectively, and $(id^*, \mathbf{y}^*, \sigma^*)$ be the forgery output by the adversary $\mathcal{A}$, where $\sigma^* = (R^*, \delta^*)$. In this security proof, if $id^* = id_{j'}$ and $R^* = R_{j'}$ for some $j' \in [1, q_s]$, we view the output of $\mathcal{A}$ as a type 2 forgery with an advantage ϵ_2. Otherwise, we view the output of $\mathcal{A}$ as a type 1 forgery with an advantage ϵ_1[5]. Obviously, $\epsilon = \epsilon_1 + \epsilon_2$.

Simulation-1. Given a random instance (g, g^x) of the DL problem over the bilinear groups $(\mathbb{G}_1, \mathbb{G}_T, p, e, g)$, $\mathcal{C}_1$ interacts with $\mathcal{A}$ as follows.

Initialization. $\mathcal{C}_1$ randomly chooses $\alpha_1, \cdots, \alpha_N \in \mathbb{F}_p^*$, computes $g_i = g^{\alpha_i}$ for all $i \in \{1, \cdots, N\}$, and initializes $\mathcal{A}$ with $params = (\mathbb{G}_1, \mathbb{G}_T, p, e, g, g_1, \cdots, g_N)$ and $pk = g^x$.

Hash Query. $\mathcal{C}_1$ keeps a list L_{H_1} which is initially empty. L_{H_1} is utilized to record H_1 queries. When $\mathcal{A}$ requests the value of $H_1(id_l, R_l)$, it provides a representation vector of R_l. $\mathcal{C}_1$ does the following steps:

- If (id_l, R_l) exists in L_{H_1}, return the corresponding $h_l = H_1(id_l, R_l)$ to $\mathcal{A}$.
- If (id_l, R_l) does not exist in L_{H_1}, pick a random $h_l \in \mathbb{F}_p^*$, return it to $\mathcal{A}$, and add (id_l, R_l, h_l) to L_{H_1}.

Sign Query. When $\mathcal{A}$ requests a signature of a vector subspace $V_l \subset \mathbb{F}_p^N$ described by properly augmented basis vectors $\mathbf{v}_{l,1}, \cdots, \mathbf{v}_{l,m} \in \mathbb{F}_p^N$, $\mathcal{C}_1$ randomly chooses $id_l \in \{0,1\}^\lambda$, h_l, $k_l \in \mathbb{F}_p^*$ and does the following steps:

1. Compute $R_l = g^{k_l} \cdot (g^x)^{-h_l}$. If (id_l, R_l) exists in L_{H_1}, abort. Otherwise, add (id_l, R_l, h_l) to L_{H_1}.
2. Compute $\delta_{l,i} = (\prod_{j=1}^N g_j^{v_{l,i,j}})^{k_l}$ and set $\sigma_{l,i} = (R_l, \delta_{l,i})$ for all $i = 1$ to m.

$\mathcal{C}_1$ outputs id_l and $\sigma_{l,1}, \cdots, \sigma_{l,m}$.

Forgery. $\mathcal{A}$ outputs $id^* \in \{0,1\}^\lambda$, a non-zero vector $\mathbf{y}^* \in \mathbb{F}_p^N$, and a forged signature $\sigma^* = (R^*, \delta^*)$, where $R = g^{r^*}$, $h^* = H_1(id^*, R^*)$, and $\delta^* = (\prod_{j=1}^N g_j^{y_j^*})^{xh^*+r^*}$.

[5] If $id^* = id_{j'}$ and $R^* \neq R_{j'}$, the adversary $\mathcal{A}$ still needs to obtain a new $h^* = H_1(id^*, R^*)$ from the **Hash Query** such that $h^* \neq H_1(id_j, R_j)$ $(1 \leq j \leq q_s)$. The proof for this case is the same as type 1 forgery.

It also attaches some representation vectors $[\boldsymbol{a}], [\boldsymbol{b}]$ in $\mathbb{F}_p$, such that

$$[\boldsymbol{a}] = (a_0, a_1, a_{2,1}, \cdots, a_{2,N}, a_{3,1}, \cdots, a_{3,q_s}, a_{4,1,1}, \cdots, a_{4,1,m}, \cdots, a_{4,q_s,1}, \cdots, a_{4,q_s,m}),$$
$$[\boldsymbol{b}] = (b_0, b_1, b_{2,1}, \cdots, b_{2,N}, b_{3,1}, \cdots, b_{3,q_s}, b_{4,1,1}, \cdots, b_{4,1,m}, \cdots, b_{4,q_s,1}, \cdots, b_{4,q_s,m}),$$

which indicate how the forgery is algebraically computed, i.e.,

$$R^* = g^{a_0}(pk)^{a_1} \prod_{j=1}^{N} g_j^{a_{2,j}} \prod_{j=1}^{q_s} R_j^{a_{3,j}} \prod_{j=1}^{q_s} \prod_{i=1}^{m} \delta_{j,i}^{a_{4,j,i}},$$

$$\delta^* = g^{b_0}(pk)^{b_1} \prod_{j=1}^{N} g_j^{b_{2,j}} \prod_{j=1}^{q_s} R_j^{b_{3,j}} \prod_{j=1}^{q_s} \prod_{i=1}^{m} \delta_{j,i}^{b_{4,j,i}}.$$

In order to simplify the above equations, let $e_{j,i} = \sum_{u=1}^{N} \alpha_u v_{j,i,u}$, $R_j = g^{r_j}$, $h_j = H_1(id_j, R_j)$, $k_j = xh_j + r_j$, and

$$S_0 = b_0 + \sum_{j=1}^{N} b_{2,j}\alpha_j + \sum_{j=1}^{q_s} b_{3,j}k_j + \sum_{j=1}^{q_s} k_j \sum_{j=1}^{m} b_{4,j,i}e_{j,i} - \sum_{u=1}^{N} \alpha_u y_u^* (a_0 + \sum_{j=1}^{N} a_{2,j}\alpha_j +$$

$$\sum_{j=1}^{q_s} a_{3,j}k_j + \sum_{j=1}^{q_s} k_j \sum_{i=1}^{m} a_{4,j,i}e_{j,i}).$$

Based on the above equations, it is easy to see

$$r^* = a_0 + xa_1 + \sum_{j=1}^{N} a_{2,j}\alpha_j + \sum_{j=1}^{q_s} a_{3,j}r_j + \sum_{j=1}^{q_s} k_j \sum_{i=1}^{m} a_{4,j,i}e_{j,i}, \tag{1}$$

$$(xh^* + r^*)(\sum_{i=1}^{N} \alpha_i y_i^*) = b_0 + xb_1 + \sum_{j=1}^{N} b_{2,j}\alpha_j + \sum_{j=1}^{q_s} b_{3,j}r_j + \sum_{j=1}^{q_s} k_j \sum_{i=1}^{m} b_{4,j,i}e_{j,i}, \tag{2}$$

$$x((h^* + a_1 - \sum_{j=1}^{q_s} h_j a_{3,j})(\sum_{i=1}^{N} \alpha_i y_i^*) - b_1 + \sum_{j=1}^{q_s} h_j b_{3,j}) = S_0. \tag{3}$$

If $\mathcal{A}$ outputs a type 2 forgery, $\mathcal{C}_1$ aborts. Otherwise, we consider the following two cases in **Simulation-1**.

In **Case 1-1**, $(h^* + a_1 - \sum_{j=1}^{q_s} h_j a_{3,j})(\sum_{i=1}^{N} \alpha_i y_i^*) - b_1 + \sum_{j=1}^{q_s} h_j b_{3,j} \neq 0$. We assume that $\mathcal{A}$ outputs a forgery in this case with an advantage ϵ_{11}. If $\mathcal{A}$ outputs a type 1 forgery, $\mathcal{C}_1$ can solve the DL problem by computing $x = S_0 \cdot ((h^* + a_1 - \sum_{j=1}^{q_s} h_j a_{3,j})(\sum_{i=1}^{N} \alpha_i y_i^*) - b_1 + \sum_{j=1}^{q_s} h_j b_{3,j})^{-1}$.

In **Case 1-2**, $(h^* + a_1 - \sum_{j=1}^{q_s} h_j a_{3,j})(\sum_{i=1}^{N} \alpha_i y_i^*) - b_1 + \sum_{j=1}^{q_s} h_j b_{3,j} = 0$. We assume that $\mathcal{A}$ outputs a forgery in this case with an advantage ϵ_{12}. In this case, $\mathcal{C}_1$ aborts. Obviously, $\epsilon_{11} + \epsilon_{12} = \epsilon_1$.

We next analyze the probability that $\mathcal{C}_1$ aborts in the simulation. When $\mathcal{C}_1$ responds to a **Sign Query** by choosing an identifier id and generating R,

but $\mathcal{A}$ has already requested the value of $H_1(id, R)$, the simulation aborts. The probability of this event is at most $\frac{q_s(q_h+q_s)}{4^\lambda} < \frac{1}{2^\lambda}$, which is a negligible probability. Therefore, the success probability of solving the DL problem is ϵ_{11} in **Simulation-1**.

Simulation-2. Given a random instance (g, g^x) of the DL problem over the bilinear groups $(\mathbb{G}_1, \mathbb{G}_T, p, e, g)$, $\mathcal{C}_2$ interacts with $\mathcal{A}$ as follows.

Initialization. $\mathcal{C}_2$ randomly chooses $\alpha_1, \cdots, \alpha_N, z \in \mathbb{F}_p^*$, computes $g_i = g^{\alpha_i}$ for all $i \in \{1, \cdots, N\}$, $X = g^z$, and initializes $\mathcal{A}$ with $params = (\mathbb{G}_1, \mathbb{G}_T, p, e, g, g_1, \cdots, g_N)$ and $pk = X$.

Hash Query. This query is the same as that of **Simulation-1**.

Sign Query. When $\mathcal{A}$ requests a signature of a vector subspace $V_l \subset \mathbb{F}_p^N$ described by properly augmented basis vectors $\mathbf{v}_{l,1}, \cdots, \mathbf{v}_{l,m} \in \mathbb{F}_p^N$, $\mathcal{C}_2$ randomly chooses $id_l \in \{0,1\}^\lambda, r_{l,1}, r_{l,2} \in \mathbb{F}_p^*$, computes $R_l = (g^x)^{r_{l,1}} g^{r_{l,2}}, \sigma_{l,i} = (\prod_{j=1}^N g_j^{v_{l,i,j}})^{zh_l + r_l} = (\prod_{j=1}^N g_j^{v_{l,i,j}})^{zh_l} ((g^x)^{r_{l,1}} g^{r_{l,2}})^{\sum_{j=1}^N \alpha_j v_{l,i,j}}$ for all $i = 1$ to m, where $h_l = H_1(id_l, R_l)$ is obtained from **Hash Query**, and sets $\sigma_{l,i} = (R_l, \delta_{l,i})$. Then $\mathcal{C}_1$ outputs id_l and $\sigma_{l,1}, \cdots, \sigma_{l,m}$.

Forgery. $\mathcal{A}$ outputs $id^* \in \{0,1\}^\lambda$, a non-zero vector $\mathbf{y}^* \in \mathbb{F}_p^N$, and a forged signature $\sigma^* = (R^*, \delta^*)$, where $R = g^{r^*}$, $h^* = H_1(id^*, R^*)$, and $\delta^* = (\prod_{j=1}^N g_j^{y_j^*})^{zh^* + r^*}$. It also attaches some representation vectors $[\boldsymbol{a}], [\boldsymbol{b}]$, which are the same as those of **Simulation-1**. If $\mathcal{A}$ outputs a type 1 forgery, $\mathcal{C}_2$ aborts. Otherwise, suppose $id^* = id_{j'}$ and $r^* = r_{j'}$ for some $j' \in [1, q_s]$, $r_j = r_{j,1}x + r_{j,2}$ for all $j \in [1, q_s]$, and $S_1 = b_0 + zb_1 + \sum_{j=1}^N b_{2,j}\alpha_j + \sum_{j=1}^{q_s} b_{3,j} r_{j,2} + \sum_{j=1}^{q_s} (zh_j + r_{j,2}) \sum_{i=1}^m b_{4,j,i} e_{j,i} - (zh^* + r_{j',2})(\sum_{i=1}^N \alpha_i y_i^*)$, according to equation (2), we can see $(r_{j',1} \sum_{i=1}^N \alpha_i y_i^* - \sum_{j=1}^{q_s} b_{3,j} r_{j,1} - \sum_{j=1}^{q_s} r_{j,1} \sum_{i=1}^m b_{4,j,i} e_{j,i}) x = S_1$. Then we consider the following two cases in this simulation.

In **Case 2-1**, $r_{j',1} \sum_{i=1}^N \alpha_i y_i^* - \sum_{j=1}^{q_s} b_{3,j} r_{j,1} - \sum_{j=1}^{q_s} r_{j,1} \sum_{i=1}^m b_{4,j,i} e_{j,i} \neq 0$. We assume that $\mathcal{A}$ outputs a forgery in this case with an advantage ϵ_{21}. If $\mathcal{A}$ outputs a type 2 forgery, $\mathcal{C}_2$ can solve the DL problem by computing $x = S_1 \cdot (r_{j',1} \sum_{i=1}^N \alpha_i y_i^* - \sum_{j=1}^{q_s} b_{3,j} r_{j,1} - \sum_{j=1}^{q_s} r_{j,1} \sum_{i=1}^m b_{4,j,i} e_{j,i})^{-1}$.

In **Case 2-2**, $r_{j',1} \sum_{i=1}^N \alpha_i y_i^* - \sum_{j=1}^{q_s} b_{3,j} r_{j,1} - \sum_{j=1}^{q_s} r_{j,1} \sum_{i=1}^m b_{4,j,i} e_{j,i} = 0$. We assume that $\mathcal{A}$ outputs a forgery in this case with an advantage ϵ_{22}. In this case, $\mathcal{C}_2$ aborts. Obviously, $\epsilon_{21} + \epsilon_{22} = \epsilon_2$.

From the above, we know the probability of successful simulation and useful attack is ϵ_{21} in **Simulation-2**.

Simulation-3. Given a random instance (g, g^x) of the DL problem over the bilinear groups $(\mathbb{G}_1, \mathbb{G}_T, p, e, g)$, $\mathcal{C}_3$ interacts with $\mathcal{A}$ as follows.

Initialization. $\mathcal{C}_3$ randomly chooses $\eta_1, \cdots, \eta_N, \mu_1, \cdots, \mu_N, z \in \mathbb{F}_p^*$, computes $g_i = (g^x)^{\eta_i} \cdot g^{\mu_i}$ for all $i \in \{1, \cdots, N\}$, $X = g^z$, and initializes $\mathcal{A}$ with $params = (\mathbb{G}_1, \mathbb{G}_T, p, g, e, g_1, \cdots, g_N)$ and $pk = X$.

Hash Query. This query is the same as that of **Simulation-1**.

Sign Query. When $\mathcal{A}$ requests a signature of a vector subspace $V_l \subset \mathbb{F}_p^N$ described by properly augmented basis vectors $\mathbf{v}_{l,1}, \cdots, \mathbf{v}_{l,m} \in \mathbb{F}_p^N$, $\mathcal{C}_3$ randomly chooses $id_l \in \{0,1\}^\lambda, r_l \in \mathbb{F}_p^*$, computes $R_l = g^{r_l}, \sigma_{l,i} = (\prod_{j=1}^N g_j^{v_{l,i,j}})^{zh_l + r_l}$ for all $i = 1$ to m, where $h_l = H_1(id_l, R_l)$ is obtained from **Hash Query**, and sets $\sigma_{l,i} = (R_l, \delta_{l,i})$. Then $\mathcal{C}_1$ outputs id_l and $\sigma_{l,1}, \cdots, \sigma_{l,m}$.

Forgery. $\mathcal{A}$ outputs $id^* \in \{0,1\}^\lambda$, a non-zero vector $\mathbf{y}^* \in \mathbb{F}_p^N$, and a forged signature $\sigma^* = (R^*, \delta^*)$, where $R = g^{r^*}$, $h^* = H_1(id^*, R^*)$, and $\delta^* = (\prod_{j=1}^N g_j^{y_j^*})^{zh^* + r^*}$. It also attaches some representation vectors $[\boldsymbol{a}], [\boldsymbol{b}]$, which are the same as those of **Simulation-1**. If $\mathcal{A}$ outputs the forgery in **Case 1-1** or **Case 2-1**, $\mathcal{C}_3$ aborts. Otherwise, let $\alpha_i = \eta_i x + \mu_i$ for all $i \in [1, N]$, we consider two cases that are **Case 1-2** and **Case 2-2** respectively.

In **Case 1-2**, $(h^* + a_1 - \sum_{j=1}^{q_s} h_j a_{3,j})(\sum_{i=1}^N \alpha_i y_i^*) - b_1 + \sum_{j=1}^{q_s} h_j b_{3,j} = 0$, it happens with an advantage ϵ_{12}. If $\mathcal{A}$ outputs a type 1 forgery, $\mathcal{C}_3$ can solve the DL problem by computing

$$x = \frac{b_1 - \sum_{j=1}^{q_s} h_j b_{3,j} - (h^* + a_1 - \sum_{j=1}^{q_s} h_j a_{3,j}) \sum_{i=1}^N \mu_i y_i^*}{(h^* + a_1 - \sum_{j=1}^{q_s} h_j a_{3,j}) \cdot \sum_{i=1}^N \eta_i y_i^*}.$$

In **Case 2-2**, $r_{j',1} \sum_{i=1}^N \alpha_i y_i^* - \sum_{j=1}^{q_s} b_{3,j} r_{j,1} - \sum_{j=1}^{q_s} r_{j,1} \sum_{i=1}^m b_{4,j,i} e_{j,i} = 0$ (i.e., $r_{j',1}(\sum_{i=1}^N \alpha_i y_i^* - b_{3,j'} - b_{4,j',i} e_{j',i}) - \sum_{j=1, j \neq j'}^{q_s} r_{j,1}(b_{3,j} + \sum_{i=1}^m b_{4,j,i} e_{j,i}) = 0$), it happens with an advantage ϵ_{22}. Since $r_{j,1}, \cdots, r_{q_s,1}$ are each independently uniform in $\mathbb{F}_p^*$ even conditioned on the view of $\mathcal{A}$, $\sum_{i=1}^N \alpha_i y_i^* - b_{3,j'} - b_{4,j',i} e_{j',i} = 0$ holds. If $\mathcal{A}$ outputs a type 2 forgery, $\mathcal{C}_3$ can solve the DL problem by computing $\mathbf{v}^* = \mathbf{y}^* - \sum_{i=1}^m y_{n+i} \mathbf{v}_{j',i}$ ($\mathbf{v}^* \neq 0^{n+m}$), $\sum_{i=1}^N \alpha_i y_i^* - b_{3,j'} - b_{4,j',i} e_{j',i} - \sum_{i=1}^m y_{n+i} \mathbf{v}_{j',i} + \sum_{i=1}^m y_{n+i} \mathbf{v}_{j',i} = 0$, and

$$x = \frac{b_{3,j'} - \sum_{l=1}^n \mu_l (v_l^* - \sum_{i=1}^m b_{4,j',i} v_{j',i,l} + \sum_{i=1}^m y_{n+i}^* v_{j',i,l}) - \sum_{i=1}^m \mu_{n+i}(y_{n+i}^* - b_{4,j',i})}{\sum_{l=1}^n \eta_l (v_l^* - \sum_{i=1}^m b_{4,j',i} v_{j',i,l} + \sum_{i=1}^m y_{n+i}^* v_{j',i,l}) + \sum_{i=1}^m \eta_{n+i}(y_{n+i}^* - b_{4,j',i})}.$$

For **Case 1-2**, since h^* is chosen at random after $\mathcal{A}$ chooses $[\boldsymbol{a}]$, the probability that $h^* + a_1 - \sum_{j=1}^{q_s} h_j a_{3,j} = 0$ is negligible. Meanwhile, as $\eta_1, \eta_2, \cdots, \eta_N$ are each independently uniform in $\mathbb{F}_p$ even conditioned on the view of $\mathcal{A}$, the probability that $\sum_{i=1}^N \eta_i y_i^* = 0$ or $\sum_{l=1}^n \eta_l (v_l^* - \sum_{i=1}^m b_{4,j',i} v_{j',i,l} + \sum_{i=1}^m y_{n+i}^* v_{j',i,l}) + \sum_{i=1}^m \eta_{n+i}(y_{n+i}^* - b_{4,j',i}) = 0$ is negligible. Therefore, the probability of successful simulation and useful attack is $\epsilon_{12} + \epsilon_{22}$ in **Simulation-3**.

It is obvious that the above three simulations are indistinguishable. We describe the success probability of solving the DL problem as follows,

$$\begin{aligned} Pr[Success] &= Pr[Success|\mathcal{C}_1]Pr[\mathcal{C}_1] + Pr[Success|\mathcal{C}_2]Pr[\mathcal{C}_2] + Pr[Success|\mathcal{C}_3]Pr[\mathcal{C}_3] \\ &= \epsilon_{11} \cdot \frac{1}{3} + \epsilon_{21} \cdot \frac{1}{3} + (\epsilon_{12} + \epsilon_{22}) \cdot \frac{1}{3} = \frac{\epsilon_1 + \epsilon_2}{3} = \frac{\epsilon}{3}. \end{aligned}$$

Therefore, the success probability of solving the DL problem is at least $\frac{\epsilon}{3}$, which indicates that the reduction loss of the MF-LHS scheme is 3. This completes the proof.

5 Efficiency Analysis

Since the BLS-LHS scheme is classic and effective, we compare the efficiency of the BLS-LHS scheme, the PF-LHS scheme, and the MF-LHS scheme.

Communication Cost. Let $|p|$, $|\mathbb{G}_1|$, and $|\mathbb{G}|$ be the size of elements in $\mathbb{F}_p$, $\mathbb{G}_1$, and $\mathbb{G}$ respectively. The communication cost comparisons are presented in Table 1, from which we find that the size of public keys in the PF-LHS scheme linearly increases with the dimension of the augmented vector, but the size of public keys in the BLS-LHS scheme and the MF-LHS scheme is constant. The signature size of the MF-LHS scheme is slightly larger than that of the BLS-LHS scheme.

Table 1. Comparisons of communication cost

Scheme	Public Key	Signature
BLS-LHS	$\|\mathbb{G}_1\|$	$\|\mathbb{G}_1\|$
PF-LHS	$(N+1)\|\mathbb{G}\|$	$\|p\| + \|\mathbb{G}\|$
MF-LHS	$\|\mathbb{G}_1\|$	$2\|\mathbb{G}_1\|$

Computation Cost. Using a personal computer (PC) with the Ubuntu 22.04.1 (4 GB memory, 4 processor cores) VMware 17.0.0 in a laptop running Windows 11 with 12th Gen Intel(R) Core(TM) i5-12500H @ 2.50 GHz and 16.0 GB RAM, we evaluate cryptographic operations by using C/C++ programming language with MIRACL library [29]. For the BLS-LHS scheme and the MF-LHS scheme, we use the Tate pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ to achieve the security level of 80 bits, where $\mathbb{G}_1$ is an additive group generated by a point $\hat{P}$ with the order $\hat{p}$ on the super singular elliptic curve $\hat{E} : y^2 = x^3 + x \mod \hat{q}$, $\hat{q}$ is a 512-bit prime number, and $\hat{p}$ is a 160-bit Solinas prime number. For the PF-LHS scheme, we use an additive group $\mathbb{G}$ of order $\bar{p}$ generated by a point P on a non-singular elliptic curve $E : y^2 = x^3 + ax + b \mod \bar{q}$ to achieve the security level of 80 bits, where $\bar{q}$ and $\bar{p}$ are 160-bit prime numbers, and $a, b \in \mathbb{Z}_{\bar{p}}^*$.

As the computational cost to perform the normal hash function and addition/multiplication in $\mathbb{Z}_{\hat{p}}$ are very small, we ignore these trifling operations in the following evaluation. For convenience, we use the following notations to denote the execution time of cryptographic operations, each of which is the average value obtained by running the operation 1000 times.

- BP=3.839 ms: the execution time of a symmetric bilinear pairing.
- $SM_{\mathbb{G}_1}$=1.634 ms: the execution time of a scalar multiplication in $\mathbb{G}_1$.
- $PA_{\mathbb{G}_1}$=0.015 ms: the execution time of a point addition in $\mathbb{G}_1$.

- T_{mtp}=4.310 ms: the execution time of a MTP hash that maps $\{0,1\}^*$ to $\mathbb{G}_1$.
- $SM_{\mathbb{G}}$=0.481 ms: the execution time of a scalar multiplication in $\mathbb{G}$.
- $PA_{\mathbb{G}}$=0.005 ms: the execution time of a point addition in $\mathbb{G}$.
- $A_{\bar{p}}$=0.0005 ms: the execution time of an addition in $\mathbb{Z}_{\bar{p}}$.
- $M_{\bar{p}}$=0.0004 ms: the execution time of a multiplication in $\mathbb{Z}_{\bar{p}}$.

To facilitate computational cost comparisons, we assume that the **Combine** algorithm involves m vectors. From Table 2, we see that the PF-LHS scheme has significantly less computation cost in terms of signature generation, verification, and combination than the BLS-LHS scheme and the MF-LHS scheme, and the MF-LHS scheme requires less computation cost of signature generation and verification than the BLS-LHS scheme.

Table 2. Comparisons of computation cost

Scheme	Sign	Verify	Combine
BLS-LHS	$mT_{mtp} + N \cdot SM_{\mathbb{G}_1}$ $+(N-1)PA_{\mathbb{G}_1}$ $\approx 5.959m + 1.649n - 0.015$	$2BP + mT_{mtp} + N \cdot SM_{\mathbb{G}_1}$ $+(N-1)PA_{\mathbb{G}_1}$ $\approx 5.959m + 1.649n + 7.663$	$mSM_{\mathbb{G}_1} + (m-1)PA_{\mathbb{G}_1}$ $\approx 1.649m - 0.015$
PF-LHS	$N \cdot A_{\bar{p}} + (N+2)M_{\bar{p}}$ $\approx 0.0009(m+n) + 0.0008$	$N \cdot PA_{\mathbb{G}} + (N+2)SM_{\mathbb{G}}$ $\approx 0.486(m+n) + 0.962$	$mM_{\bar{p}} + (m-1)A_{\bar{p}}$ $\approx 0.0009m - 0.0005$
MF-LHS	$N \cdot SM_{\mathbb{G}_1} + (N-1)PA_{\mathbb{G}_1}$ $\approx 1.649(m+n) - 0.015$	$2BP + (N+1) \cdot SM_{\mathbb{G}_1}$ $+N \cdot PA_{\mathbb{G}_1}$ $\approx 1.649(m+n) + 9.312$	$mSM_{\mathbb{G}_1} + (m-1)PA_{\mathbb{G}_1}$ $\approx 1.649m - 0.015$

$N = n + m$: the dimension of the augmented vector.
n: the dimension of the original vector.
m: the number of basis vectors.

6 Conclusion

In this work, we give tight security reduction for LHS schemes under the DL assumption. At first, we prove the security of the BLS-LHS scheme proposed by Boneh et al. [8] under the DL assumption in the AGM with a random oracle. Our first security proof involves two simulations that can capture both types of forgeries, resulting in a reduction loss of two. To improve computational efficiency, we propose a PF-LHS scheme and an MF-LHS scheme, and the latter can also be tightly reduced to the DL problem in the AGM with a random oracle. Moreover, the efficiency analysis shows that the PF-LHS scheme and the MF-LHS scheme require less computational cost compared with the BLS-LHS scheme.

Further research could consider the possibility of the PF-LHS scheme that provides ideal security. The PF-LHS scheme is the most computationally efficient, however, we identified that type 2 forgery of the PF-LHS scheme cannot be tightly reduced to the DL problem even in the AGM as it is unable to both respond to sign queries and extract the problem solution from type 2 forgery with the same hash value as that of some sign query in a simulation.

References

1. Ahn, J.H., Boneh, D., Camenisch, J., Hohenberger, S., Shelat, A., Waters, B.: Computing on authenticated data. Journal of Cryptology **28**(2), 351–395 (2015)
2. Ateniese, G., Kamara, S., Katz, J.: Proofs of storage from homomorphic identification protocols. In: Advances in Cryptology–ASIACRYPT 2009: 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings 15. pp. 319–333. Springer (2009)
3. Attrapadung, N., Libert, B.: Homomorphic network coding signatures in the standard model. In: Public Key Cryptography–PKC 2011: 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings 14. pp. 17–34. Springer (2011)
4. Bacho, R., Loss, J.: On the adaptive security of the threshold bls signature scheme. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. pp. 193–207 (2022)
5. Bellare, M., Dai, W.: Chain reductions for multi-signatures and the hbms scheme. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 650–678. Springer (2021)
6. Bellare, M., Rogaway, P.: Random oracles are practical: A paradigm for designing efficient protocols. In: Proceedings of the 1st ACM Conference on Computer and Communications Security. pp. 62–73 (1993)
7. Boneh, D., Boyen, X.: Short signatures without random oracles and the sdh assumption in bilinear groups. Journal of cryptology **21**(2), 149–177 (2008)
8. Boneh, D., Freeman, D., Katz, J., Waters, B.: Signing a linear subspace: Signature schemes for network coding. In: Public Key Cryptography–PKC 2009: 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings 12. pp. 68–87. Springer (2009)
9. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22. pp. 416–432. Springer (2003)
10. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Theory of Cryptography: Second Theory of Cryptography Conference, TCC 2005, Cambridge, MA, USA, February 10-12, 2005. Proceedings 2. pp. 325–341. Springer (2005)
11. Catalano, D., Fiore, D., Warinschi, B.: Efficient network coding signatures in the standard model. In: Public Key Cryptography–PKC 2012: 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings 15. pp. 680–696. Springer (2012)
12. Chen, Y., Zhao, Y.: Half-aggregation of schnorr signatures with tight reductions. In: European Symposium on Research in Computer Security. pp. 385–404. Springer (2022)
13. Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory (1976)
14. Freeman, D.M.: Improved security for linearly homomorphic signatures: A generic framework. In: Public Key Cryptography–PKC 2012: 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, May 21-23, 2012. Proceedings 15. pp. 697–714. Springer (2012)

15. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Advances in Cryptology–CRYPTO 2018: 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2018, Proceedings, Part II 38. pp. 33–62. Springer (2018)
16. Fuchsbauer, G., Plouviez, A., Seurin, Y.: Blind schnorr signatures and signed elgamal encryption in the algebraic group model. In: Advances in Cryptology–EUROCRYPT 2020: 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10–14, 2020, Proceedings, Part II 30. pp. 63–95. Springer (2020)
17. Goh, E.J., Jarecki, S., Katz, J., Wang, N.: Efficient signature schemes with tight reductions to the diffie-hellman problems. Journal of Cryptology **20**, 493–514 (2007)
18. Kastner, J., Loss, J., Xu, J.: On pairing-free blind signature schemes in the algebraic group model. In: IACR International Conference on Public-Key Cryptography. pp. 468–497. Springer (2022)
19. Kılınç Alper, H., Burdges, J.: Two-round trip schnorr multi-signatures via delinearized witnesses. In: Advances in Cryptology–CRYPTO 2021: 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16–20, 2021, Proceedings, Part I 41. pp. 157–188. Springer (2021)
20. Lewko, A., Waters, B.: New techniques for dual system encryption and fully secure hibe with short ciphertexts. In: Theory of Cryptography Conference. pp. 455–479. Springer (2010)
21. Li, J., Chen, X., Huang, X., Tang, S., Xiang, Y., Hassan, M.M., Alelaiwi, A.: Secure distributed deduplication systems with improved reliability. IEEE Transactions on Computers **64**(12), 3569–3579 (2015)
22. Li, Y., Zhang, M., Zhang, F.: Structure-preserving linearly homomorphic signature with designated combiner for subspace. In: Australasian Conference on Information Security and Privacy. pp. 229–243. Springer (2022)
23. Loh, J.C., Guo, F., Susilo, W., Yang, G.: A tightly secure id-based signature scheme under dl assumption in agm. In: Australasian Conference on Information Security and Privacy. pp. 199–219. Springer (2023)
24. Maurer, U.M., Wolf, S.: The relationship between breaking the diffie–hellman protocol and computing discrete logarithms. SIAM Journal on Computing **28**(5), 1689–1721 (1999)
25. Nick, J., Ruffing, T., Seurin, Y.: Musig2: simple two-round schnorr multi-signatures. In: Annual International Cryptology Conference. pp. 189–221. Springer (2021)
26. Schabhüser, L., Buchmann, J., Struck, P.: A linearly homomorphic signature scheme from weaker assumptions. In: Cryptography and Coding: 16th IMA International Conference, IMACC 2017, Oxford, UK, December 12-14, 2017, Proceedings 16. pp. 261–279. Springer (2017)
27. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Advances in Cryptology—CRYPTO'89 Proceedings 9. pp. 239–252. Springer (1990)
28. Schnorr, C.P.: Efficient signature generation by smart cards. Journal of cryptology **4**, 161–174 (1991)
29. SDK, M.C.: Miracl cryptographic sdk: Multiprecision integer and rational arithmetic cryptographic library (2022)
30. Waters, B.: Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In: Annual International Cryptology Conference. pp. 619–636. Springer (2009)