Vol.2021-DPS-186 No.36
Vol.2021-CSEC-92 No.36
2021/3/15

格子を用いた複数鍵線形準同型署名

小﨑 俊二[1,a)]　有田 正剛[1,b)]

概要：本稿では，Boneh と Freeman (Eurocrypt2011) により提案された線形準同型署名に Aranha と Pagnin (Latincrypt2019) の手法を適用して複数ユーザー (鍵) の署名に対して利用可能な線形準同型署名を構成する．Boneh と Freeman の準同型署名は SIS 問題の困難性を安全性の仮定としているが，単一鍵による署名のみに利用可能である．また，Aranha と Pagnin の複数鍵を利用可能な線形準同型署名はペアリングに関する離散対数問題の困難性を安全性の仮定としている．本稿で構成する複数鍵を利用可能な線形準同型署名は SIS 問題の困難性仮定のもとで安全性が証明されることから量子計算機耐性を持つと考えられる．

Multi-key linear homomorphic signature based on lattices

1. はじめに

準同型署名はクラウドのような信頼できない計算資源を利用するときにそれらの計算結果の信頼性を保証するために利用可能な署名方式である．準同型署名では，あらかじめ署名されたデータを入力として計算された結果に対して，署名鍵を持たない第三者がその入力としたデータの署名からその計算結果に対する簡潔な署名を生成できる．

本論文では，SIS 問題の困難性を安全性の根拠とした単一の鍵ではなくそれぞれ異なる複数の鍵で署名された入力データに対する一次式の計算に適用可能な準同型署名の構成を示す．

[3, 4, 9, 10] では単一ユーザー (鍵) によって署名されたデータに対する準同型署名が提案されたが，[5] は複数ユーザー (鍵) に対する準同型署名を定義し，SIS 問題の困難性を安全性の根拠とした深さ固定の回路の計算に対する準同型署名を構成した．その後複数鍵に対する準同型署名として，単一鍵の準同型署名スキームからの複数鍵のスキームへの変換法 [6]，ZK-SNARK のもとでの一般的構成法 [11]，ペアリングを用いた一次式の計算に対する準同型署名の構成 [2, 13] などが提案されている．

本論文の複数鍵準同型署名は [3] の単一鍵準同型署名をもとにして，[2] の手法を適用して単一ユーザー鍵ごとの部分計算結果に対する署名をそれぞれ並べた組を全体の署名とし，署名検証は各単一ユーザー鍵ごとの部分計算結果の署名検証と全ユーザーの署名の総和が計算結果と一致することを確認することでおこなう．[2] ではペアリングを持つ群上の離散対数に関する問題を安全性の根拠としているが，本論文で構成する準同型署名は SIS 問題の困難性を安全性の根拠とするため量子計算機耐性を持つと考えられる．[5] も SIS 問題の困難性を安全性仮定とした複数鍵準同型署名であるが，本論文で構成する準同型署名は計算可能な式を 1 次式に制限するため，その署名長を [5] の署名長の約 1/2 乗以下にできる．また，[5] では計算式をブール回路としているためデータをバイナリ表現であつかうが，本論文の構成ではデータ空間をより大きな有限体とすることが可能である．

2. 準備

正整数全体を $\mathbb{N}$, 整数全体を $\mathbb{Z}$, 実数全体を $\mathbb{R}$ で表し，$k \in \mathbb{N}$ に対して $[k] := \{1, \ldots, k\}$ と書く．$q \in \mathbb{N}$ に対して q を法とする剰余環を $\mathbb{Z}_q$ と書く．行列 $A \in \mathbb{Z}_q^{n \times m}$ に対して A をパリティ検査行列とする m 次元整数格子を $\Lambda_q^{\perp}(A) := \{\mathbf{z} \in \mathbb{Z}^m | A\mathbf{z} = \mathbf{0} \bmod q\}$ で表す．また，$n \in \mathbb{N}$ についての関数 $f(n)$ が任意の正定数 $c \in \mathbb{R}$ に対して $f(n) = O(n^{-c})$ であるとき無視可能関数とよび $f(n) = \mathrm{negl}(n)$ と表す．

[1] 情報セキュリティ大学院大学
Institute of Information Security, Yokohama, Kanagawa 221–0835, Japan
[a)] dgs074103@iisec.ac.jp
[b)] arita@iisec.ac.jp

ⓒ 2021 Information Processing Society of Japan

2.1 SIS 問題

提案スキームの安全性の根拠となる SIS 問題 [1] を定義する．

定義 1 ($\mathrm{SIS}_{n,q,\beta}$ 問題)．$n, q, m \in \mathbb{N}$, $\beta \in \mathbb{R}_{>0}$ とするとき，一様ランダムに選ばれた $A \in \mathbb{Z}_q^{n\times m}$ に対して $\|\mathbf{z}\| \leq \beta$ かつ $A\mathbf{z} = \mathbf{0} \bmod q$ を満足する $\mathbf{z} \in \mathbb{Z}^m \setminus \{\mathbf{0}\}$ を求める問題を $\mathrm{SIS}_{n,q,\beta}$ 問題と呼ぶ．

$\mathrm{SIS}_{n,q,\beta}$ 問題は $m \geq 2n\log q$, $\beta \geq \sqrt{m}$ ならばほとんどの A に対して解が存在する．さらに $\beta = \mathrm{poly}(n)$, $q \geq \beta \cdot \omega\left(\sqrt{n\log n}\right)$ ならば $\mathrm{SIS}_{n,q,\beta}$ 問題を解くことは，任意の n 次元格子上の近似 Shortest Independent Vector Problem (SIVP) を解くことと少なくとも同程度の困難さであることが知られている [8, Proposition 5.7]．ここで，SIVP の近似係数は $\beta \cdot \tilde{O}(\sqrt{n})$ である．現在までこの近似 SIVP を量子計算機で効率的に解くアルゴリズムは知られていないことから，$\mathrm{SIS}_{n,q,\beta}$ 問題も量子計算機で効率的に解くことは難しいと考えられている．

2.2 トラップドア付格子生成およびガウス分布サンプリングアルゴリズム

提案スキーム中で利用するトラップドア付格子を生成する $\mathsf{TrapGen}$ アルゴリズムと，その生成された格子に関するガウス分布に従った短いベクトルをサンプリングする $\mathsf{SamplePre}$ アルゴリズムについて述べる．

[12, Theorem 5.1] より $n \in \mathbb{N}$, $q \geq 2$, $m \geq 2n\log q$ を入力として，ほぼ一様ランダムな行列 $A \in \mathbb{Z}_q^{n\times m}$ とその格子 $\Lambda_q^{\perp}(A)$ のトラップドア R を出力する $\mathsf{TrapGen}(n,q,m) \mapsto (A,R)$ が得られる．さらに，この A, R, n 次元ベクトル $\mathbf{u} \in \mathbb{Z}_q^n$ と十分な大きさのパラメータ $s = O\left(\sqrt{n\log q}\right)$ を入力として，$A\mathbf{t} \equiv \mathbf{u} \pmod q$ となる $\mathbf{t} \in \mathbb{Z}^m$ の剰余類の元 $\mathbf{s} \in \mathbf{t} + \Lambda_q^{\perp}(A)$ を出力する $\mathsf{SamplePre}(A,R,\mathbf{u},s) \mapsto \mathbf{s}$ が得られる．この $\mathsf{SamplePre}$ の出力は離散ガウス分布 $D_{\mathbf{t}+\Lambda_q^{\perp}(A),s}$ とほぼ同一であり，したがって圧倒的な確率で $\|\mathbf{s}\| < s\sqrt{m}$ となる [12, Lemma 2.6]．また十分に大きな $s = O\left(\sqrt{n\log q}\right)$ と任意に固定した n 次元ベクトル $\mathbf{u}$ に対して $\mathsf{SamplePre}$ が同一の $\mathbf{s}$ を出力する確率は $\mathrm{negl}(n)$ である [12, Lemma 2.7]．

2.3 複数鍵準同型署名

準同型署名では計算結果の正当性を示す署名を生成するためにどのような計算を行ったのかを記述する必要がある．このため次で定義されるラベル付きプログラム [7] を利用する．

定義 2 (ラベル付きプログラム)．計算を行う対象のメッセージ空間を $\mathcal{M}$，それらのメッセージを識別するためにつけるラベル空間を $\mathcal{L}$，また準同型署名の計算が可能な関数集合を $\mathcal{F}$ とするとき，ラベル付きプログラム $\mathcal{P}$ は $\mathcal{F}$ の計算式を表す関数 $f : \mathcal{M}^k \to \mathcal{M}$ とその入力値のラベル $\ell_1, \ldots, \ell_k \in \mathcal{L}$ の組 $\mathcal{P} = (f, \ell_1, \ldots, \ell_k)$ である．ここで，$\ell_1, \ldots, \ell_k$ の並び順は f の k 個の変数の順序とする．

g を $\mathcal{M}$ 上の k 変数関数，$(\mathcal{P}_i)_{i\in[k]}$ を k 個のラベル付きプログラムとするとき，ラベル付きプログラムの合成を $\mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$ とあらわす．ここで，$\mathcal{P}$ に含まれるラベルはすべての $(\mathcal{P}_i)_{i\in[k]}$ に含まれるラベルにわたって重複を除き，異なる $\mathcal{P}_i$ に含まれる同じラベルに対応する変数を一つに置き換えて関数の合成を行うものとする．特に $\mathcal{M}$ 上の恒等関数 I に対してラベル付きプログラムを $\mathcal{I}_i := (I, \ell_i)), i \in [k]$ とすると，$\mathcal{P} = g(\mathcal{I}_1, \ldots, \mathcal{I}_k) = (g, \ell_1, \ldots, \ell_k)$ である．ただし，一般のラベル付きプログラムの合成 $\mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$ の場合には $\mathcal{P}$ の関数の記述は g の各変数に各 $\mathcal{P}_i, i \in [k]$ の関数を合成したものとは異なることに注意されたい．

複数ユーザー (鍵) に対応した準同型署名では上の定義で述べたラベル空間としてそれぞれのユーザーを識別するための識別子空間 $\mathcal{N}$ と各ユーザーがメッセージにつけるタグ空間 $\mathcal{T}$ の直積 $\mathcal{L} := \mathcal{N} \times \mathcal{T}$ を使う．

定義 3 (複数鍵準同型署名 (MKHS) スキーム)．複数鍵準同型署名 (MKHS) は次の 5 つのアルゴリズム $\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify}$ で構成される．

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$
  セキュリティパラメータ λ を入力として公開パラメータ pp を設定する．公開パラメータにはメッセージ空間 $\mathcal{M}$, ユーザ識別子空間 $\mathcal{N}$, タグ空間 $\mathcal{T}$, 準同形署名を計算可能な関数集合 $\mathcal{F}$ が含まれる．以降のアルゴリズムでは特に明記しなくとも pp は入力に含まれるものとする．
- $\mathsf{KeyGen}(\mathsf{id}) \to (\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}})$
  ユーザ識別子 id を入力として，その id に対する秘密鍵 $\mathsf{sk}_{\mathsf{id}}$, 公開鍵 $\mathsf{pk}_{\mathsf{id}}$ を出力する．
- $\mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{m}, \ell = (\mathsf{id}, \tau)) \to \sigma$
  メッセージ m に対してそのユーザ識別子 id とタグ τ からなるメッセージ識別子 ℓ をラベルとして，$\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}$ を用いて m の署名 σ を出力する．
- $\mathsf{Eval}\left(g, \left\{\left(\Sigma_i, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\Sigma_i}\right)\right\}_{i\in[k]}\right) \to \Sigma$
  関数 $g : \mathcal{M}^k \to \mathcal{M}$ とその変数に対応する署名 $(\Sigma_1, \ldots, \Sigma_k)$ から，計算結果 $\mathsf{m} = g(\mathsf{m}_1, \ldots, \mathsf{m}_k)$ に対する署名 Σ を出力する．ここで，$\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\Sigma_i}$ は署名 Σ_i に含まれるユーザの公開鍵の集合である．
- $\mathsf{Verify}\left(\mathcal{P} = (f, \ell_1, \ldots, \ell_k), \mathsf{m}, \Sigma, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}\right) \to 1 \text{ or } 0$
  ラベル付きプログラム $\mathcal{P}$ と計算結果のメッセージ m と署名 Σ，および $\mathcal{P}$ に含まれるすべてのユーザの公開鍵集合 $\{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}$ を入力として，m が $\mathcal{P}$ の正しい計算結果すなわち $\mathsf{m} = f(\mathsf{m}_1, \ldots, \mathsf{m}_k)$ であるならば 1 を，そうでなければ 0 を出力する．ここで，$(\mathsf{m}_1, \ldots, \mathsf{m}_k)$ は $\mathcal{P}$ のラベル $(\ell_1, \ldots, \ell_k)$ に対応するメッセージであるが，これらのメッセージは入力として与えられてい

ないことに注意されたい．

この MKHS スキームは次の 2 つの正当性を満たす必要がある．

- Authentication 正当性:
  任意のユーザ識別子 $\mathsf{id} \in \mathcal{N}$ に対して $\mathsf{KeyGen}(\mathsf{id})$ から生成された秘密鍵・公開鍵ペア $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}})$ と，任意のメッセージ $\mathsf{m} \in \mathcal{M}$ とそのタグ $\tau \in \mathcal{T}$ に対して $\mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{m}, \ell = (\mathsf{id}, \tau))$ から生成された署名 σ は
  $$\mathsf{Verify}(\mathcal{I} = (I, \ell), \mathsf{m}, \sigma, \mathsf{pk}_{\mathsf{id}}) = 1$$
  を満足する．ここで，I は $\mathcal{M}$ 上の恒等関数である．
- Evaluation 正当性:
  $g : \mathcal{M}^k \to \mathcal{M}$ と $\mathsf{Verify}\left(\mathcal{P}_i, \mathsf{m}_i, \Sigma_i, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}_i}\right) = 1$ を満足する $((\mathcal{P}_i, \mathsf{m}_i, \Sigma_i))_{i\in[k]}$ に対して，$\mathcal{P} := g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$，$\mathsf{m} := g(\mathsf{m}_1, \ldots, \mathsf{m}_k)$ とするとき，$\mathsf{Eval}\left(g, \left\{\left(\Sigma_i, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\Sigma_i}\right)\right\}_{i\in[k]}\right)$ から生成された署名 Σ は
  $$\mathsf{Verify}\left(\mathcal{P}, \mathsf{m}, \Sigma, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}\right) = 1$$
  を満足する．

さらに MKHS スキームでは計算に対する入力長に対してその計算結果の署名長が十分に短いという次の条件を満たす必要がある．

- Succinctness:
  ラベル付きプログラム $\mathcal{P} = (f, \ell_1, \ldots, \ell_k)$ に対する計算結果の署名を Σ とするとき，
  $$|\Sigma| = O(t \log k)$$
  を満足する．ここで，t は $\mathcal{P}$ に入力データ与えているユーザー数である．

2.4 複数鍵準同型署名の安全性

[5, 3.1 節] では複数鍵準同型署名に対する攻撃者として Non-adaptive Corruption を仮定し，選択平文攻撃に対する MKHS スキームの安全性を定義している．本論文ではさらにランダムオラクルモデルを仮定してメッセージにつけるラベルを任意長とすることで，[5] で使われている dataset identifier を利用しない形で以下のように安全性ゲーム (MKHUF-CMA) を構成する[*1]．

(1) Setup: 挑戦者 $\mathcal{C}$ は $\mathsf{Setup}(1^\lambda)$ で生成した公開パラメータ pp を攻撃者 $\mathcal{A}$ に送る．また $\mathcal{C}$ は 3 つの集合 $L_{\mathrm{ID}} \leftarrow \emptyset, L_{\mathsf{sig}} \leftarrow \emptyset, L_{\mathsf{corr}} \leftarrow \emptyset$ を初期化する．

(2) Identity Queries: 攻撃者 $\mathcal{A}$ が id についての公開鍵を問い合わせてきたとき，この id に関して最初の問い合わせの場合すなわち $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathrm{ID}}$ ならば挑戦者 $\mathcal{C}$ は $\mathsf{KeyGen}(\mathsf{id})$ により秘密鍵・公開鍵ペア $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}})$ を生成し，$d := |L_{\mathrm{ID}}|+1$, $L_{\mathrm{ID}} \leftarrow L_{\mathrm{ID}} \cup \{(\mathsf{id}, d, \mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}})\}$ としてこの鍵ペアを記録し，公開鍵 $\mathsf{pk}_{\mathsf{id}}$ を攻撃者 $\mathcal{A}$ に送る．既にこの問い合わせの id について鍵ペアが生成済みすなわち $(\mathsf{id}, d, \mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) \in L_{\mathrm{ID}}$ ならばその公開鍵 $\mathsf{pk}_{\mathsf{id}}$ を攻撃者 $\mathcal{A}$ に送る．

(3) Corruption Query: 攻撃者 $\mathcal{A}$ が id についての秘密鍵を問い合わせてきたとき，$(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathrm{ID}}$ ならばこの要求を無視する．既に $(\mathsf{id}, d, \mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) \in L_{\mathrm{ID}}$ が存在するならばその鍵ペア $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}})$ を攻撃者 $\mathcal{A}$ に送る．また，$L_{\mathsf{corr}} \leftarrow L_{\mathsf{corr}} \cup \{\mathsf{id}\}$ として id を記録する．

(4) Sign Queries: 攻撃者 $\mathcal{A}$ がラベル $\ell = (\mathsf{id}, \tau)$ についてのメッセージ m への署名を要求してきたとき，id の鍵ペアが存在しないか，またはラベル ℓ が異なるメッセージ m' への署名に既に使われていた場合，すなわち $(\mathsf{id}, \cdot, \cdot, \cdot) \notin L_{\mathrm{ID}}$ または $(\ell, \mathsf{m}') \in L_{\mathsf{sig}}, \mathsf{m}' \neq \mathsf{m}$ ならばこの要求を無視する．上記以外の場合挑戦者 $\mathcal{C}$ は $\mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}, \mathsf{m}, \ell)$ として署名 σ を生成し，σ を攻撃者 $\mathcal{A}$ に送る．また，$L_{\mathsf{sig}} \leftarrow L_{\mathsf{sig}} \cup \{(\ell, \mathsf{m})\}$ と記録する．

(5) Forgery: 攻撃者 $\mathcal{A}$ は上記問い合わせを繰り返した後，偽造結果として $(\mathcal{P}^*, \mathsf{m}^*, \Sigma^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*})$ を出力する．

この攻撃者 $\mathcal{A}$ の偽造結果が次の偽造成功条件を満足するとき $\mathcal{A}$ はこのゲームに勝ったと定義する．

MKHUF-CMA における攻撃者 $\mathcal{A}$ の出力 $(\mathcal{P}^*, \mathsf{m}^*, \Sigma^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*})$ が偽造成功とは，$\mathcal{P}^*$ に含まれるすべてのユーザ識別子の秘密鍵が一切漏洩していない，すなわち $\forall \mathsf{id} \in \mathcal{P}^* : \mathsf{id} \notin L_{\mathsf{corr}}$ であり，かつ
$$\mathsf{Verify}(\mathcal{P}^*, \mathsf{m}^*, \Sigma^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}) = 1$$
を満足し，$\mathcal{P}^* = (f^*, \ell_1^*, \cdots, \ell_k^*)$ としたとき次のどちらかの条件が成り立つ場合:
Type1: $\exists i \in [k]$ s.t. $(\ell_i^*, \cdot) \notin L_{\mathsf{sig}}$,
Type2: $\forall i \in [k], (\ell_i^*, \mathsf{m}_i) \in L_{\mathsf{sig}}, \mathsf{m}^* \neq f^*(\mathsf{m}_1, \ldots, \mathsf{m}_k)$
である．

定義 4 (MKHS の安全性)．複数鍵準同型署名 (MKHS) が選択平文攻撃に対して偽造不可であるとは，セキュリティパラメータを λ として上記 MKHUF-CMA ゲームで任意の確率的多項式時間攻撃者 $\mathcal{A}$ が勝つ確率が
$$\mathrm{Adv}_{\mathcal{A}}^{\text{MKHUF-CMA}}(\lambda) \leq \mathrm{negl}(\lambda)$$
となるときである．

3. 提案スキーム

本節では [3] の中で提案された格子を用いた単一鍵線形準同型署名をもとにして [2] の手法にしたがって複数鍵線形準同型署名 (MKLHS) を構成する．

[*1] [5] の Type1 Forgery がなくなり Type3 Forgery が本論文の Type1 に相当する．

3.1 複数鍵線形準同型署名スキームの構成

2.3 節で述べた複数鍵準同型署名の Evaluation 正当性は初めにメッセージにつけた署名 (一次署名) に対する計算結果の署名 (合成署名) に対してだけではなく, それらの合成署名を持つ計算結果を入力とした計算結果の署名, すなわちラベル付きプログラムを合成した計算結果に対しても署名が検証可能 (マルチホップ) であることを求める．本節で構成する複数鍵線形準同型署名スキーム MKLHS では, このラベル付きプログラムの合成回数の上限 (最大ホップ数) を $C \in \mathbb{N}$ として, $\mathrm{MKLHS} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Eval}, \mathsf{Verify})$ を以下のように構成する．

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp} = (m, n, p, q, s, \delta, H, \mathcal{M}, \mathcal{N}, \mathcal{T}, \mathcal{F})$
  $n(\lambda)$ を自然数, $q(\lambda)$ を素数, $m \in O(n \log q)$ を自然数, p を q と異なる素数，実数 s, δ をそれぞれ $s > \sqrt{m}$, $\delta > Kp^{C+1}s\sqrt{m}, K = O(n)$ とし，$H : \{0,1\}^* \to \mathbb{Z}_q^n$ をハッシュ関数とする．メッセージ空間を $\mathcal{M} := \mathbb{F}_p^m$, ユーザ識別子空間を $\mathcal{N} := \{0,1\}^*$，タグ空間を $\mathcal{T} := \{0,1\}^*$，準同型署名計算可能な関数の集合を $\mathcal{F} := \left\{ f = \sum_{i\in[k]} a_i X_i, k \in \mathbb{N}, a_i \in \mathbb{F}_p \right\}$ とする．
- $\mathsf{KeyGen}(\mathsf{id}) \to (\mathsf{sk}_\mathsf{id}, \mathsf{pk}_\mathsf{id})$

(1) $(A_\mathsf{id} \in \mathbb{Z}_q^{n\times m}, R_\mathsf{id}) \leftarrow \mathsf{TrapGen}(n, q, m)$

(2) $\mathsf{sk}_\mathsf{id} := R_\mathsf{id}$, $\mathsf{pk}_\mathsf{id} := A_\mathsf{id}$.

- $\mathsf{Sign}\left(\mathsf{sk}_\mathsf{id}, \mathsf{pk}_\mathsf{id}, \mathsf{m}, \ell = (\mathsf{id}, \tau)\right) \to \sigma$

(1) $\mathbf{s} \leftarrow \mathsf{SamplePre}\left(\mathsf{pk}_\mathsf{id}, \mathsf{sk}_\mathsf{id}, H(\ell), s\right)$,

(2) $\mathbf{z} \equiv \mathsf{m} \pmod p$ かつ $\mathbf{z} \equiv \mathbf{s} \pmod q$ を満足する $\mathbf{z} \in \mathbb{Z}^m$ を求め $\sigma := \mathbf{z}$ とする．
/* $\|\sigma\| < ps\sqrt{m}$, $A_\mathsf{id}\sigma \equiv H(\ell) \pmod q$ */

- $\mathsf{Eval}\left(g, \left\{\left(\Sigma_i, \{\mathsf{pk}_\mathsf{id}\}_{\mathsf{id}\in\Sigma_i}\right)\right\}_{i\in[k]}\right) \to \Sigma$
  $g = \sum_{i\in[k]} g_i X_i \in \mathbb{F}_p[X_1, \ldots, X_k]$, $\Sigma_i = \left(\sigma_\mathsf{id}^{(i)}\right)_{\mathsf{id}\in\Sigma_i}$ とする．$\{\Sigma_i\}_{i\in[k]}$ に含まれるすべてのユーザ識別子の集合を $\{\mathsf{id}_j\}_{j\in[t]} := \bigcup_{i\in[k]} \{\mathsf{id}\}_{\mathsf{id}\in\Sigma_i}$ とかく．

(1) for $j \in [t]$:
  $\sigma_{\mathsf{id}_j} \leftarrow 0$
  for $i \in [k]$:
    $\sigma_{\mathsf{id}_j} \leftarrow \sigma_{\mathsf{id}_j} + g_i \sigma_{\mathsf{id}_j}^{(i)}$,

(2) $\Sigma := (\sigma_{\mathsf{id}_1}, \ldots, \sigma_{\mathsf{id}_t})$.

- $\mathsf{Verify}\left(\mathcal{P} = (f, \ell_1, \ldots, \ell_k), \mathsf{m}, \Sigma, \{\mathsf{pk}_\mathsf{id}\}_{\mathsf{id}\in\mathcal{P}}\right) \to 1/0$
  $f = \sum_{i\in[k]} f_i X_i \in \mathbb{F}_p[X_1, \ldots, X_k]$ とし，$(\ell_i)_{i\in[k]}$ に含まれるすべてのユーザ識別子の集合を $\{\mathsf{id}_j\}_{j\in[t]} := \{\mathsf{id} : \ell_i = (\mathsf{id}, \tau)\}_{i\in[k]}$ とかき，$\Sigma = (\sigma_{\mathsf{id}_j})_{j\in[t]}$ とする．さらに，各 $\mathsf{id}_j, j \in [t]$ を $(\ell_i)_{i\in[k]}$ の添え字の部分集合とみて，$\ell_i = (\mathsf{id}, \tau)$ に対して $\mathsf{id}_j = \mathsf{id}$ ならば "$i \in \mathsf{id}_j$" とかくことにする．このとき，次の 3 条件:

a. $\left\|\sigma_{\mathsf{id}_j}\right\| \overset{?}{<} \delta$ for $j \in [t]$,

b. $\sum_{j\in[t]} \sigma_{\mathsf{id}_j} \overset{?}{\equiv} \mathsf{m} \pmod p$,

c. for $j \in [t]$:
  $A_{\mathsf{id}_j}\sigma_{\mathsf{id}_j} \overset{?}{\equiv} \sum_{i\in\mathsf{id}_j} f_i H(\ell_i) \pmod q$,

の a. と b. と c. すべてが成り立てば 1 を出力，そうでなければ 0 を出力する．

最初にこのスキームの Authentication 正当性を確認する．$\mathsf{KeyGen}(\mathsf{id}) \to (\mathsf{sk}_\mathsf{id}, \mathsf{pk}_\mathsf{id})$ に対して，$\mathsf{Sign}(\mathsf{sk}_\mathsf{id}, \mathsf{pk}_\mathsf{id}, \mathsf{m}, \ell = (\mathsf{id}, \tau))$ で生成された署名 σ が Verify の a.，b.，c. の 3 条件を満たすことを確かめればよい． 2.2 節で述べた SamplePre の性質より圧倒的確率で $\|\sigma\| < ps\sqrt{m}$ となり，$\delta > Kp^{C+1}s\sqrt{m}$ より $\|\sigma\| < \delta$ であるので条件 a. を満足する．また Sign アルゴリズムのステップ (2) より，$\sigma \equiv \mathsf{m} \pmod p$ かつ $A_\mathsf{id}\sigma \equiv H(\ell) \pmod q$ であるので, 条件 b. c. も満足する．

次に Evaluation 正当性を確認する．$\mathcal{P} = (f, \ell_1, \ldots, \ell_k)$, $f = \sum_{i\in[k]} f_i X_i \in \mathbb{F}_p[X_1, \ldots, X_k]$ とし，ラベル $(\ell_i)_{i\in[k]}$ がつけられたメッセージを $(\mathsf{m}_i)_{i\in[k]}$ とし, それぞれの一次署名を $(\sigma_i)_{i\in[k]}$ とする．このとき，$\mathsf{Eval}\left(f, \{(\sigma_i, \{\mathsf{pk}_\mathsf{id}\})\}_{i\in[k]}\right)$ で合成された署名 $\Sigma = (\sigma_{\mathsf{id}_1}, \ldots, \sigma_{\mathsf{id}_t})$ が Verify の 3 条件を満足するか確認する．Eval のステップ (1) では関数 f の入力の署名についてユーザーごとの部分和を計算している，すなわち $\sigma_{\mathsf{id}_j} = \sum_{i\in\mathsf{id}_j} f_i\sigma_i$ である．一方 SamplePre で生成した一次署名は $\|\sigma_i\| < ps\sqrt{m}$ であるので，$\|\sigma_{\mathsf{id}_j}\| < kp^2 s\sqrt{m} < \delta$ となり，条件 a. を満足する．また，$\sum_{j\in[t]} \sigma_{\mathsf{id}_j} = \sum_{j\in[t]} \sum_{i\in\mathsf{id}_j} f_i\sigma_i = \sum_{i\in[k]} f_i\sigma_i \equiv \sum_{i\in[k]} f_i\mathsf{m}_i \equiv \mathsf{m} \pmod p$ より，条件 b. も満足する．さらに，各 $j \in [t]$ に対して一次署名 $\{\sigma_i\}_{i\in[k]}$ は $A_{\mathsf{id}_j}\sigma_i \equiv H(\ell_i) \pmod q, i \in \mathsf{id}_j$ を満足するので, $A_{\mathsf{id}_j}\sigma_{\mathsf{id}_j} = A_{\mathsf{id}_j}\sum_{i\in\mathsf{id}_j} f_i\sigma_i = \sum_{i\in\mathsf{id}_j} f_i A_{\mathsf{id}_j}\sigma_i \equiv \sum_{i\in\mathsf{id}_j} f_i H(\ell_i)$ より，条件 c. も満足する．

さらに合成署名に対する Evaluation 正当性を確認する．$\mathcal{P} = g(\mathcal{P}_1, \ldots, \mathcal{P}_k)$, $g = \sum_{i\in[k]} g_i X_i$ とし，$((\mathcal{P}_i, \mathsf{m}_i, \Sigma_i))_{i\in[k]}$ が $\mathsf{Verify}\left(\mathcal{P}_i, \mathsf{m}_i, \Sigma_i, \{\mathsf{pk}_\mathsf{id}\}_{\mathsf{id}\in\mathcal{P}_i}\right) = 1$ を満足するとき，$i \in [k]$ に対して $\Sigma_i = \left(\Sigma_\mathsf{id}^{(i)}\right)_{\mathsf{id}\in\Sigma_i}$ とかけば Verify の条件 b. より，$\sum_{\mathsf{id}\in\Sigma_i} \Sigma_\mathsf{id}^{(i)} \equiv \mathsf{m}_i \pmod p$ である．したがって，$\mathsf{Eval}\left(g, \left\{\left(\Sigma_i, \{\mathsf{pk}_\mathsf{id}\}_{\mathsf{id}\in\Sigma_i}\right)\right\}_{i\in[k]}\right)$ から生成された署名を $\Sigma = (\Sigma_\mathsf{id})_{\mathsf{id}\in\Sigma}$ とかけば，$\Sigma_\mathsf{id} = \sum_{i\in\mathsf{id}} g_i\Sigma_\mathsf{id}^{(i)}$ であるので,

$$\begin{aligned}\sum_{\mathsf{id}\in\Sigma} \Sigma_\mathsf{id} &= \sum_{\mathsf{id}\in\Sigma} \sum_{i\in\mathsf{id}} g_i\Sigma_\mathsf{id}^{(i)} = \sum_{i\in[k]} \sum_{\mathsf{id}\in\Sigma_i} g_i\Sigma_\mathsf{id}^{(i)} \\ &\equiv \sum_{i\in[k]} g_i\mathsf{m}_i \equiv g(\mathsf{m}_1, \ldots, \mathsf{m}_k) \equiv \mathsf{m} \pmod p\end{aligned}$$

より，Σ は Verify の条件 b. を満足する．また，各 $\Sigma_i = \left(\Sigma_\mathsf{id}^{(i)}\right)_{\mathsf{id}\in\Sigma_i}$ は Verify の条件 c. も満足することから，$\forall i \in [k]$, $\forall \mathsf{id} \in \Sigma_i$, に対して $A_\mathsf{id}\Sigma_\mathsf{id}^{(i)} \equiv \sum_{j\in\Sigma^{(i)}} f_j^{(i)} H(\ell_j^{(i)}) \pmod q$ を満足する．ここで，$\mathcal{P}_i = \left(f^{(i)} = \sum_{j\in[k_i]} f_j^{(i)} X_j, \ell_1^{(i)}, \ldots, \ell_{k_i}^{(i)}\right), i \in [k]$ とした．さらに $\forall \mathsf{id} \in \Sigma$ に対して，$\Sigma_\mathsf{id} = \sum_{i\in\mathsf{id}} g_i\Sigma_\mathsf{id}^{(i)}$ であるので,

$$A_{\mathsf{id}}\Sigma_{\mathsf{id}} = A_{\mathsf{id}}\sum_{i\in\mathsf{id}} g_i\Sigma_{\mathsf{id}}^{(i)} = \sum_{i\in\mathsf{id}} g_i A_{\mathsf{id}}\Sigma_{\mathsf{id}}^{(i)}$$
$$\equiv \sum_{i\in\mathsf{id}}\sum_{j\in\Sigma^{(i)}} g_i f_j^{(i)} H(\ell_j^{(i)}) \pmod q$$

となる．このとき，ラベル付きプログラムの合成は $\mathcal{P} = g(\mathcal{P}_1,\ldots,\mathcal{P}_k) = \left(\sum_{i\in[k]} g_i \sum_{j\in\Sigma_i} f_j^{(i)} X_i, \ell_1,\ldots,\ell_k\right)$ となるので，Σ は Verify の条件 c. を満足する．一次署名のサイズは $\|\sigma_i\| < ps\sqrt{m}$ であり，ラベル付きプログラムの合成の最大回数が C であるので，各 Σ_{id} のサイズは $\|\Sigma_{\mathsf{id}}\| < kp^{C+}s\sqrt{m} < \delta$ であり．Verify の条件 a. を満足する．したがって，

$$\mathsf{Verify}\left(\mathcal{P}, \mathsf{m}, \Sigma, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}}\right) = 1$$

を満足する．

最後に $\mathsf{Eval}\left(f, \{(\sigma_i, \{\mathsf{pk}_{\mathsf{id}}\})\}_{i\in[k]}\right)$ で合成された $\Sigma = (\sigma_{\mathsf{id}_1},\ldots,\sigma_{\mathsf{id}_t})$ は Verify の条件 a. を満たすことから，署名長は $\|\Sigma\| < t\delta$ である．$\delta \in O(\log k)$ ととれば Succinct を満足する．

3.2 提案スキームの安全性

前節で構成した MKLHS の安全性を示すために，MKHUF-CMA ゲームで偽造をおこなう攻撃者 $\mathcal{A}$ を用いて SIS 問題を解くシミュレータ $\mathcal{S}$ を次のように構成する．

SIS 問題の入力を $A \in \mathbb{Z}_q^{n\times m}$ として，

(1) Setup: $\mathcal{S}$ は n, m, q 以外の公開パラメータを MKLHS の Setup を利用して生成し，それらを $\mathcal{A}$ に送り起動する．MKHUF-CMA ゲームと同様に集合 $L_{\mathrm{ID}} \leftarrow \emptyset, L_{\mathsf{sig}} \leftarrow \emptyset, L_{\mathsf{corr}} \leftarrow \emptyset$ を初期化する．またハッシュ値の記録のため集合 $L_H \leftarrow \emptyset$ を初期化する．

(2) Keygen: $\mathcal{A}$ がゲーム中に問い合わせるユーザ識別子の総数を $D \in \mathbb{N}$ として，$\mathcal{S}$ は $d \in [D]$ をあらかじめ選択する．$\mathcal{A}$ から d 番目に問い合わせられたユーザ識別子 $\tilde{\mathsf{id}}$ に対しては $\mathcal{S}$ は公開鍵として $\mathsf{pk}_{\tilde{\mathsf{id}}} := A$ を $\mathcal{A}$ に送り，$L_{\mathrm{ID}} \leftarrow L_{\mathrm{ID}} \cup \left\{(\tilde{\mathsf{id}}, d, *, \mathsf{pk}_{\tilde{\mathsf{id}}})\right\}$ を記録する．d 番目以外の j 番目のユーザ識別子 id に対しては 通常通り $(\mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) \leftarrow \mathsf{KeyGen}(\mathsf{pp}, \mathsf{id})$ で生成した公開鍵 $\mathsf{pk}_{\mathsf{id}}$ を $\mathcal{A}$ に送り，$L_{\mathrm{ID}} \leftarrow L_{\mathrm{ID}} \cup \{(\mathsf{id}, j, \mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}})\}, j \in [D]\backslash\{d\}$ を記録する．

(3) Sign: $\mathcal{A}$ からの $(\ell = (\mathsf{id}, \tau), \mathsf{m})$ に対する署名要求にはその id が d 番目のユーザ識別子についての署名要求すなわち $(\mathsf{id} = \tilde{\mathsf{id}}, d, \mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) \in L_{\mathrm{ID}}$ ならば $\mathcal{S}$ は離散ガウス分布より $\sigma \leftarrow D_{\Lambda_q^{\perp}(A),s}$ をサンプリングして $H(\ell) := A\sigma \pmod q$ をハッシュ値として $L_H \leftarrow L_H \cup \{(\ell, H(\ell))\}$ に記録し，$\mathcal{A}$ に署名として σ を応答する．また，d 番目以外のユーザ識別子についての署名要求すなわち $(\mathsf{id}, j \neq d, \mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) \in L_{\mathrm{ID}}$ ならば $\mathcal{S}$ は通常の署名生成 $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{id}}, \mathsf{m}, \ell, \mathsf{pk}_{\mathsf{id}})$ を行い σ を $\mathcal{A}$ に応答する．

(4) Hash: $\mathcal{A}$ からのラベル $\ell = (\mathsf{id}, \tau)$ についてのハッシュ値の問い合わせに対して，その id が d 番目のユーザ識別子についてのラベルのハッシュ値要求すなわち $(\mathsf{id} = \tilde{\mathsf{id}}, d, \mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) \in L_{\mathrm{ID}}$ ならば $\mathcal{S}$ は離散ガウス分布より $\sigma \leftarrow D_{\Lambda_q^{\perp}(A),s}$ をサンプリングして $H(\ell) := A\sigma \pmod q$ を L_H に記録し，$\mathcal{A}$ にハッシュ値 $H(\ell)$ を応答する．また，d 番目以外のユーザ識別子のラベルについてのハッシュ値要求すなわち $(\mathsf{id}, j \neq d, \mathsf{sk}_{\mathsf{id}}, \mathsf{pk}_{\mathsf{id}}) \in L_{\mathrm{ID}}$ ならば $\mathcal{S}$ は 通常のハッシュ値 $H(\ell)$ を計算し $\mathcal{A}$ に応答する．

(5) Solution: $\mathcal{A}$ が出力した偽造を

$$\left(\mathcal{P}^*, \mathsf{m}^*, \Sigma^*, \{\mathsf{pk}_{\mathsf{id}}\}_{\mathsf{id}\in\mathcal{P}^*}\right) \tag{1}$$

とし，$\Sigma^* = \left(\sigma^*_{\mathsf{id}_1},\ldots,\sigma^*_{\mathsf{id}_t}\right), \mathcal{P}^* = (f^*, \ell_1^*,\ldots,\ell_k^*)$ とする．

(1) が Type1 の偽造，すなわち $(\ell_i^*, \cdot) \notin L_{\mathsf{sig}}$ となる $\ell_i^* = (\mathsf{id}_s, \tau)$ が存在した場合，$\mathsf{id}_s \neq \tilde{\mathsf{id}}$ ならば abort する．$\mathsf{id}_s = \tilde{\mathsf{id}}$ ならば $\mathbf{x} := \sigma^*_{\mathsf{id}_s} - \sum_{i\in\mathsf{id}_s} f_i^*\sigma_i$ を計算し，$\mathbf{x} \neq \mathbf{0}$ ならば，$\mathbf{x}$ を出力する．ここで，σ_i はハッシュ値 $H(\ell_i^*) = A\sigma_i \pmod q$ を計算する際にサンプリングしたものである．

また (1) が Type2 の偽造, すなわち $1 \le i \le k$ に対して $(\ell_i^*, \mathsf{m}_i) \in L_{\mathsf{sig}}$ の場合，$\mathsf{id}_s = \tilde{\mathsf{id}}$ となる $1 \le s \le t$ が存在しないならば abort する．$\mathsf{id}_s = \tilde{\mathsf{id}}$ となる $1 \le s \le t$ が存在するならば，$\mathcal{S}$ は $\mathbf{x} := \sigma^*_{\mathsf{id}_s} - \sum_{i\in\mathsf{id}_s} f_i^*\sigma_i$ を計算し，$\mathbf{x} \neq \mathbf{0}$ ならば，$\mathbf{x}$ を出力する．ここで，σ_i は $(\ell_i^*, \mathsf{m}_i) \in L_{\mathsf{sig}}$ に対する署名である．

上記 step.5 における $\mathcal{A}$ の出力 (1) が偽造成功ならば, Verify の中の 3 条件:

$$\|\sigma^*_{\mathsf{id}_j}\| < \delta \text{ for } j = [t], \tag{2}$$

$$\sum_{j=1}^{t} \sigma^*_{\mathsf{id}_j} \equiv \mathsf{m}^* \pmod p, \tag{3}$$

$$A_{\mathsf{id}_j}\sigma^*_{\mathsf{id}_j} \equiv \sum_{i\in\mathsf{id}_j} f_i^* H(\ell_i) \pmod q \text{ for each } j = [t], \tag{4}$$

を満足する．さらに $A_{\mathsf{id}_s} = A_{\tilde{\mathsf{id}}} = A$ であるので 条件 (4) より

$$A\mathbf{x} = A_{\mathsf{id}_s}\left(\sigma^*_{\mathsf{id}_s} - \sum_{i\in\mathsf{id}_s} f_i^*\sigma_i\right) = A_{\mathsf{id}_s}\sigma^*_{\mathsf{id}_s} - A_{\mathsf{id}_s}\sum_{i\in\mathsf{id}_s} f_i^*\sigma_i$$
$$\equiv \sum_{i\in\mathsf{id}_s} f_i^* H(\ell_i) - \sum_{i\in\mathsf{id}_s} f_i^* A_{\mathsf{id}_s}\sigma_i$$
$$\equiv \sum_{i\in\mathsf{id}_s} f_i^* \left(H(\ell_i) - A_{\mathsf{id}_s}\sigma_i\right) \pmod q$$

である．このとき，各 $i \in \mathsf{id}_s$ について $A_{\mathsf{id}_s}\sigma_i \equiv H(\ell_i) \pmod q$ であることから

$$A\mathbf{x} \equiv \sum_{i\in\mathsf{id}_s} f_i^* \left(H\left(\ell_i\right) - A_{\mathsf{id}_s}\sigma_i\right) \equiv \mathbf{0} \pmod q$$

となるので，$\mathbf{x} \neq \mathbf{0}$ ならば $\mathbf{x}$ は A に対する $\mathrm{SIS}_{n,q,\beta}$ 問題の解である．ここで，(3)Sign および (4)Hash オラクルにおけるハッシュ値 $H(\ell_i) := A\sigma_i \pmod q$, $i \in \mathsf{id}_s$ の生成では離散ガウス分布 $\sigma_i \leftarrow D_{\Lambda_q^\perp(A),s}$ を利用しているので，2.2 節で述べたように同一の $H(\ell_i)$ に対して σ_i が衝突する確率は $\mathrm{negl}(n)$ である [12, Lemma 2.7]．したがって攻撃者 $\mathcal{A}$ が $\mathbf{x} = \sigma^*_{\mathsf{id}_s} - \sum_{i\in\mathsf{id}_s} f_i^*\sigma_i = \mathbf{0}$ を満足する $\sigma^*_{\mathsf{id}_s}$ を出力する確率は無視可能である．

(5)Solution step における $\mathcal{A}$ の出力 (1) が Type1 の偽造のとき $\mathcal{S}$ が abort する確率を評価する．$\mathcal{S}$ は $\mathcal{A}$ が問い合わせた総数 D のユーザ識別子の中から $\tilde{\mathsf{id}}$ を選んでいるので，$\mathsf{id}_s = \tilde{\mathsf{id}}$ となる確率は $1/D$ である．したがって，$\mathcal{S}$ が $\mathrm{SIS}_{n,q,\beta}$ 問題を解く確率を $\mathrm{Adv}^{\mathrm{SIS}}_{\mathcal{S}}(n)$，$\mathcal{A}$ が MKLHS の Type 1 偽造を出力する確率を $\mathrm{Adv}^{\mathrm{MKLHS}}_{\mathcal{A}_1}(n)$ とすると

$$\frac{1}{D}\mathrm{Adv}^{\mathrm{MKLHS}}_{\mathcal{A}_1}(n) \leq \mathrm{Adv}^{\mathrm{SIS}}_{\mathcal{S}}(n)$$

である．

次に，Solution step 5. における $\mathcal{A}$ の出力 (1) が Type2 の偽造のとき $\mathcal{S}$ が abort する確率を評価する．Type2 の偽造であることから式 (3) より条件:

$$\sum_{j=1}^{t} \sigma^*_{\mathsf{id}_j} \equiv \mathsf{m}^* \not\equiv f^*(m_1, \cdots, m_k) \pmod p$$

が成り立つ．一方すべての $j \in [t]$ に対して $\sigma^*_{\mathsf{id}_j} \equiv \sum_{i\in\mathsf{id}_j} f_i^* m_i \pmod p$ ならば，上記条件が成り立たないので

$$\sigma^*_{\mathsf{id}_j} \not\equiv \sum_{i\in\mathsf{id}_j} f_i^* m_i \pmod p$$

を満足する $j \in [t]$ が存在する．したがって，少なくとも 1 つの $j \in [t]$ について $\sigma^*_{\mathsf{id}_j} - \sum_{i\in\mathsf{id}_j} f_i^*\sigma_i \neq \mathbf{0}$ である．一方 s について $\mathbf{x} = \sigma^*_{\mathsf{id}_s} - \sum_{i\in\mathsf{id}_s} f_i^*\sigma_i \neq \mathbf{0}$ ならば $\mathcal{S}$ は abort しない．$\mathcal{S}$ は $\mathcal{A}$ が問い合わせた総数 D のユーザ識別子の中から id_s を選んでいるので，$\mathcal{S}$ が Solution step 5. で abort しない確率は $1/D$ である．すなわち $\mathcal{A}$ が MKLHS の Type 2 偽造を出力する確率を $\mathrm{Adv}^{\mathrm{MKLHS}}_{\mathcal{A}_2}(n)$ とすると

$$\frac{1}{D}\mathrm{Adv}^{\mathrm{MKLHS}}_{\mathcal{A}_2}(n) \leq \mathrm{Adv}^{\mathrm{SIS}}_{\mathcal{S}}(n)$$

である．したがって $\mathcal{A}$ が MKLHS において偽造成功する確率は

$$\mathrm{Adv}^{\mathrm{MKLHS}}_{\mathcal{A}}(n) \leq D \cdot \mathrm{Adv}^{\mathrm{SIS}}_{\mathcal{S}}(n)$$

である．以上から，このシミュレータ $\mathcal{S}$ の出力は

$$\|\mathbf{x}\| \leq \left\|\sigma^*_{\mathsf{id}_s} - \sum_{i\in\mathsf{id}_s} f_i^*\sigma_i\right\| \leq \|\sigma^*_{\mathsf{id}_s}\| + \left\|\sum_{i\in\mathsf{id}_s} f_i^*\sigma_i\right\| < 2\delta$$

であるので，$\delta = O(p^{C+1}n^2\log q)$ より大きな β を選ぶことが可能で，次の定理が得られる．

定理 1 (MKLHS の安全性)**.** 上記の MKLHS スキームは $\mathrm{SIS}_{n,q,\beta}$ 問題が困難ならば、選択平文攻撃に対して偽造不可である．

4. まとめ

本論文では [3] の格子を用いた単一鍵線形準同型署名をベースに，[2] の構成方法を参考に，複数鍵線形準同型署名を構成した．また，その安全性を SIS 問題に帰着することで，量子計算機耐性があることを示した．

謝辞 本研究は，JST，CREST，JPMJCR1503 の支援を受けたものである．

参考文献

[1] Ajtai, M.: Generating hard instances of lattice problems, 28th ACM STOC, ACM, pp. 99–108 (1996).
[2] Aranha, D. F. and Pagnin, E.: The simplest multi-key linearly homomorphic signature scheme, LATINCRYPT 2019, Springer, pp. 280–300 (2019).
[3] Boneh, D. and Freeman, D. M.: Homomorphic signatures for polynomial functions, EUROCRYPT 2011, Springer, pp. 149–168 (2011).
[4] Boneh, D. and Freeman, D. M.: Linearly homomorphic signatures over binary fields and new tools for lattice-based signatures, International Workshop on Public Key Cryptography, Springer, pp. 1–16 (2011).
[5] Fiore, D., Mitrokotsa, A., Nizzardo, L. and Pagnin, E.: Multi-key Homomorphic Authenticators, ASIACRYPT 2016, pp. 499–530 (2016).
[6] Fiore, D. and Pagnin, E.: Matrioska: a compiler for multi-key homomorphic signatures, SCN 2018, Springer, pp. 43–62 (2018).
[7] Gennaro, R. and Wichs, D.: Fully homomorphic message authenticators, ASIACRYPT 2013, Springer, pp. 301–320 (2013).
[8] Gentry, C., Peikert, C. and Vaikuntanathan, V.: Trapdoors for Hard Lattices and New Cryptographic Constructions, 40th ACM STOC, STOC '08, New York, NY, USA, ACM, pp. 197–206 (2008).
[9] Gorbunov, S., Vaikuntanathan, V. and Wichs, D.: Leveled Fully Homomorphic Signatures from Standard Lattices, 47th ACM STOC, STOC '15, New York, NY, USA, ACM, pp. 469–477 (2015).
[10] Johnson, R., Molnar, D., Song, D. and Wagner, D.: Homomorphic signature schemes, CT-RSA 2002, Springer, pp. 244–262 (2002).
[11] Lai, R. W., Tai, R. K., Wong, H. W. and Chow, S. S.: Multi-key homomorphic signatures unforgeable under insider corruption, ASIACRYPT 2018, Springer, pp. 465–492 (2018).
[12] Micciancio, D. and Peikert, C.: Trapdoors for lattices: Simpler, tighter, faster, smaller, EUROCRYPT 2012, Springer, pp. 700–718 (2012).
[13] Schabhüser, L., Butin, D. and Buchmann, J.: Context hiding multi-key linearly homomorphic authenticators, CT-RSA 2019, Springer, pp. 493–513 (2019).

ⓒ 2021 Information Processing Society of Japan