

効率的な閾値署名の提案

猪本 卓也^{1,a)} 宮地 充子^{1,2,b)}

概要：やり取りする情報が送信者本人から送られてきたものか、正しい情報が送られてきたかを判断することは非常に重要である。デジタル署名はそのような判断に必要な技術であり、多重署名はその一種である。多重署名は、複数人でデータを回覧したりする際に用いられ、各々がデータに署名し、複数人で1つの署名を完成させる。閾値署名はこの多重署名の1つで、秘密鍵などの署名に関する情報を分割し、複数人に分散させる。署名復元に人数の下限を設けることで、それに満たない人数では署名が生成できない。秘密鍵を分散して保管できるため、仮想通貨などに利用されている。既存研究として Gennaro らによる閾値署名では、DSA 署名と同様の検証式で検証が可能である。一方で、計算過程に秘密鍵と乱数の積が必要となり、ラグランジュ補間が実行できないため、その積を和に変換する MtA と呼ばれるプロトコルが必要である。この MtA を用いることにより、各参加者の計算量が増えている。そこで本研究では、Gennaro らのものと同一の秘密分散法を用いつつ署名式を変更することで、MtA を必要としない効率的な閾値署名方式を提案する。

キーワード：デジタル署名, 閾値署名, 秘密分散

Efficient Threshold Signature

TAKUYA INOMOTO^{1,a)} ATSUKO MIYAJI^{1,2,b)}

Abstract: Threshold signature is one type of digital signature, which divides the information requires for signature generation and distributes them to multiple people. By setting the lower limit of the number of people to recover the signature, there is an advantage that fewer people than the lower limit cannot generate a signature. The threshold signature by Gennaro et al. can be verified by the same verification formula as the DSA. On the other hand, since multiplication of secret shares is required in the signature generation, a protocol called MtA is needed to convert the multiplication into addition. By using MtA, the amount of computation for each participant is increased. In this paper, we propose an efficient threshold signature scheme without MtA. Our scheme uses the same secret sharing for the key generation but modifies the signature formula from that of Gennaro et al.

Keywords: Digital Signature, Threshold Signature, Secret Sharing

1. 序章

現代社会ではスマートフォンやパソコンを用いて、離れた場所同士でもデータをやり取りできる。一方で離れた場所、つまり顔の見えない相手から情報が送られてくると

になる。送られてきた情報をそのまま鵜呑みにするのでは、ウィルス感染やなりすましの恐れがある。そのため、受信者はその情報が正しい送信者から送られてきたものか、送信者は自らが情報を送信したことを保証する必要がある。デジタル署名はこの確認に用いられている。送信者は自らの情報で署名をし、受信者はその情報を照会、検証することで確認する。離れた場所でのコミュニケーションを可能にする通信技術には不可欠な要素だと言える。

多重署名 [1, 2] はデジタル署名の一種で、複数人でデー

¹ 大阪大学大学院 工学研究科
Graduate School of Engineering, Osaka University

² JAIST
北陸先端科学技術大学院大学

^{a)} inomoto@cy2sec.comm.eng.osaka-u.ac.jp

^{b)} miyaji@comm.eng.osaka-u.ac.jp

タを回覧したりする際に用いられ、各々がデータに署名し、複数人で1つの署名を完成させる。閾値署名 [3–6] はこの多重署名の一種である。閾値署名では、閾値に満たない人数では署名を生成できず、秘密鍵を分散 [7, 8] させることで、一か所に鍵を保管している場合に比べ攻撃を受けた際に鍵が流出するリスクを減らしている。仮想通貨が注目を集めている近年、そのセキュリティ面で閾値署名が一役買っており [4, 9]、今後の社会でも活用が増えることが予想される。Gennaro, Goldfeder の閾値署名 [3] では DSA をベースにしている。一方で、DSA を原始的に閾値署名へ変換すると、各プレイヤーは秘密鍵の share にそれぞれが選択した乱数をかける必要がある。乱数をかけた share からでは、ラグランジュ補間を用いても上手く元の秘密鍵を回復することができない。そこで Gennaro ら [3] は MtA と呼ばれるプロトコルを用い、乱数と share の積を和へと変換することで、秘密鍵の回復を実現している。しかし MtA プロトコルを用いることで、各プレイヤーに対し準同型暗号の計算や、他プレイヤーとの share の変換のために、各プレイヤーは $t' - 1$ 人のプレイヤーへのブロードキャストと 1 回のマルチキャストが必要となる。つまり、 t'^2 回の $(t' - 1)$ 人へのブロードキャストを強いることになり、通信障害に弱い欠点を持つ。

そこで本研究では、MtA プロトコルを利用しない閾値署名方式を提案する。MtA が必要となるのは秘密鍵と乱数の積が署名式に存在するためである。署名式から秘密鍵と乱数の積を削除するため、署名式を DSA ベースのものから、[1] へ変更した。MtA を利用しないことで、冪乗計算の回数を削減でき、通信回数については大きく削減している。

最後に本論文は次のように構成される。まず、2 章ではプロトコルを提案する上で必要となる諸定義や基礎的な知識について説明し、3 章で既存の閾値署名や提案方式でベースとした署名方式を紹介する。4 章で提案プロトコルを説明を行い、5 章で提案方式と既存方式との比較を行う。

2. 準備

ここでは、本研究に使用する数学的知識やプロトコルについて説明する。

2.1 DSA

DSA はデジタル署名の一種で、1994 年に NIST により標準採用された。

- PublicParameters 位数 q の巡回群 $\mathbb{G}$ 、その生成元 g 、ハッシュ関数 $H : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ 、別のハッシュ関数 $H' : \mathbb{G} \rightarrow \mathbb{Z}_q$
- KeyGen: 一様ランダムに選ばれた秘密鍵 $x \in \mathbb{Z}_q$ と、公開鍵 $y = g^x$ を出力する。
- Sign: メッセージ M を入力として、

- $m = H(M) \in \mathbb{Z}_q$ を計算
- $k \in_R \mathbb{Z}_q$ に対して、 $R = g^{k^{-1}}$, $r = H'(R)$ を計算
- $s = k(m + xr) \pmod{q}$ を計算
- $\sigma = (r, s) \in \mathbb{Z}_q^*$ を出力
- Verify: メッセージ M 、公開鍵 y 、署名 σ を入力として、
 - $r, s \in \mathbb{Z}_q$ か確認
 - $R' = g^{ms^{-1} \pmod{q}} y^{rs^{-1} \pmod{q}}$ を計算
 - $H'(R') = r$ か確認

2.2 閾値署名

閾値署名はデジタル署名の一種で、複数人で一つの署名を生成する。

(t, n) -閾値秘密分散を用いて、署名に必要な秘密鍵を n 分割する。分割した秘密鍵が $t + 1$ 個以上が集まれば、秘密鍵を回復することができ、署名を生成することができる。一方で、集まった数が t 以下であれば、秘密鍵を回復することはできない。

- Thresh – KeyGen 各プレイヤー P_i は分散された秘密鍵の share である sk_i と公開鍵 pk を得る。
- Thresh – Sign 各プレイヤーの sk_i により、メッセージ m に対する署名 σ を出力する。

2.3 コミットメント

コミットメント方式では、情報の送信者がプライバシーを確保しながら、メッセージを変更しないように相手に約束する。一方で、メッセージを公開する際には、トラップドアと呼ばれるものが必要である。コミットメントは以下の 4 つのアルゴリズムから構成される。

- KeyGen 公開鍵 pk と、トラップドア tk を出力する。
- Com 公開鍵 pk とメッセージ M を入力することで、出力 $[C(M), D(M)] = \text{Com}(pk, M, r)$ を得る (r は乱数)。 $C(M)$ はコミットメントとなったメッセージで、 $D(M)$ はコミットメントが開けられるまで秘密にされる元のメッセージである。
- Verify $C(M), D(M)$ と pk を入力することで、メッセージ M か $\perp$ を出力する。
- Equiv トラップドアを用いて、コミットメントを開けるアルゴリズム。 $pk, M, r, [C(M), D(M)], M', T$ を入力とする。もし、 $T = tk$ であれば、 $\text{Verify}(pk, C(M), D') = M'$ を満たす D' を出力する。

2.4 Feldman の VSS

Paul Feldman による VSS (Verifiable Secret Sharing) プロトコルを示す。このプロトコルは Shamir の方式 [7] を発展させたものである。分散したい秘密を $\sigma \in \mathbb{Z}_q$ とする。ディーラーが以下の形のランダムな t 次多項式を生成する。

$$p(x) = \sigma + a_1x + a_2x^2 + \dots + a_tx^t \pmod{q}$$

$p(x)$ を用いて、各プレイヤー P_i の σ の share σ_i で求める.

$$\sigma_i = p(i) \bmod q = \sigma + a_1 i + a_2 i^2 + \dots + a_t i^t \pmod{q}$$

また、追加の情報としてディールは以下の情報も発行する. g は巡回群 $\mathbb{G}$ の生成元である.

$$v_i = g^{a_i} \text{ in } \mathbb{G} \text{ for } \forall i \in [1, t], v_0 = g^\sigma \text{ in } \mathbb{G}$$

プレイヤー P_i はこの追加情報を用いて以下の式を計算することで、自身の share σ_i が正しいかを検証できる.

$$\begin{aligned} g^{\sigma_i} &= \prod_{j=0}^t v_j^{i^j} \text{ in } \mathbb{G} \\ &= \prod_{j=0}^t (g^{a_j})^{i^j} \text{ in } \mathbb{G} \\ &= g^{a_0 + a_1 i + \dots + a_t i^t} \text{ in } \mathbb{G} \end{aligned}$$

全てのプレイヤーがこの検証に成功しなければ、share か v_i が間違っていることになるので、この VSS プロトコルを終了する.

なお、秘密を復元するには、次のラグランジュ補間を計算する. x 座標がすべて異なる $t+1$ 個の点 (x_i, y_i) があるとき、これらの点を通る t 次以下の多項式が 1 つ定まり、以下の式で表される.

$$P(x) = \sum_{i=1}^{t+1} y_i \frac{f_i(x)}{f_i(x_i)}, \quad f_i(x) = \prod_{j \neq i} (x - x_j)$$

3. 既存研究

既存研究として、Gennaro らによる閾値署名方式 [3] を示す.

3.1 Share 変換プロトコル (MtA)

署名方式を示す前に、その方式内で用いられている Share 変換プロトコルを示す. これは、ある秘密 $x = ab \pmod{q}$ を満たす 2 つの乗法的秘密 $a, b \in \mathbb{Z}_q$ が存在する. このとき、以下の式を満たすような加法的秘密 α, β を手に入れるプロトコル MtA (Multiplicative to Additive) である.

$$\alpha + \beta = x = ab \pmod{q}$$

まず、Alice は a , Bob は b をそれぞれ持っている想定する. また、加法準同型暗号 E (Gennaro らの研究では Pailler 暗号) を用いる. Alice は他に E の公開鍵 E_A と秘密鍵を持っているとする. MtA では、Bob の持つ情報 $B = g^b$ を公開する場合があります、この場合はその情報が正しいものかを Bob が追加で証明する必要がある. このため、この場合の MtA を MtAwc (with check) と呼ぶ.

(1) Alice は

- $c_A = E_A(a)$ を Bob に送る.

- $a < K$ を Bob に対してゼロ知識証明 ($\text{ZKP}\{a \mid a < K\}$) する.

(2) Bob は $\beta' \in \mathbb{Z}_N$ を一様ランダムに選択し以下の計算を行う.

$$c_B = b \times_E c_A +_E E_A(\beta') = E_A(ab + \beta')$$

また、 $\beta = -\beta' \pmod{q}$ として、 β を設定し、以下の操作を行う.

- c_B を Alice に送る
- $b < K$ を Alice に対してゼロ知識証明 ($\text{ZKP}\{b \mid b < K\}$) する.
- $B = g^b$ を公開する場合は、Bob が b, β' を知っていることを Alice に対してゼロ知識証明 ($\text{ZKP}\{b \mid g^b, \text{ZKP}\{\beta' \mid c_B = b \times_E c_A +_E E_A(\beta')\}\}$) する.

(3) Alice は持っている秘密鍵を用いて c_B を復号し α' を得ることで、 $\alpha = \alpha' \pmod{q}$ を得る.

3.2 閾値署名プロトコルの説明

この方式は、鍵生成、署名生成、署名検証の 3 つの要素から構成されており、各プレイヤーは位数 q の巡回群 $\mathbb{G}$ とその生成元 g の上でプロトコルを回す. 各プレイヤー P_i は Paillier の加法準同型暗号方式 E の公開鍵 E_i を持っており、後に share 変換プロトコルを利用するのに使われる.

• 鍵生成プロトコル

署名生成に必要な秘密鍵を秘密分散し、その share を n 人に分配する. この share を持つ n 人の中から、閾値となる t 人よりも多いプレイヤーが集まれば、ラグランジュの補完公式を用いて share から秘密鍵を復元することができ、署名生成が可能となる.

(1) 各プレイヤー P_i はランダムに $u_i \in \mathbb{Z}_q$ をとり、 $[KGC_i, KGD_i] = \text{Com}(g^{u_i})$ を計算、 KGC_i, E_i を公開する. ここで、 KGC_i はコミットメントされた値、 KGD_i はデコミットメントされる値であり、以下も同様の記述があるが、これに準拠する. E_i は Paillier の暗号方式で利用する公開鍵である.

(2) 各プレイヤー P_i は KGD_i を公開し、 $y_i = g^{u_i}$ とする. 署名の公開鍵を $y = \prod_i y_i$ とすると、それに対応する署名の秘密鍵は $x = \sum_i u_i$ である.

P_i は Feldman の (t, n) -VSS プロトコルで u_i を n 分割し、各プレイヤーそれぞれに配布. 各プレイヤーは受け取った n 個の share を合計することで、秘密鍵 x の秘密分散 x_i を得る.

(3) 各プレイヤーは、自身が x_i を知っていることをゼロ知識証明 ($\text{ZKP}\{x_i \mid g^{x_i}\}$) する.

• 署名生成プロトコル

実際に署名 (r, s) を作成するプロトコル. S を署名プロトコルに参加するプレイヤーの集合とする. ここで S の要素数 t' について、 $t < t' \leq n$ と想定する.

また、この方式の (t, n) -secret sharing では、 t 次多項式を用いているため、ラグランジュの補完公式から、分散させた秘密 x を求めるには (t, n) である必要はなく (t', t') で十分である。従って、そのラグランジュの補完公式を満たすように適切なラグランジュ係数 $\lambda_{i,S}$ を用いて各プレイヤーは自身の (t, n) -share である x_i を (t', t') -share となる $w_i = (\lambda_{i,S})(x_i)$ に変換する。 $x = \sum_t w_i$ である。

- (1) P_i はランダムに $k_i, \gamma_i \in \mathbb{Z}_q$ を選択し、 $[C_i, D_i] = \text{Com}(g^{\gamma_i})$ を計算、 C_i を公開する。 $k = \sum_{i \in S} k_i, \gamma = \sum_{i \in S} \gamma_i$ と定義すると以下が成り立つ。

$$k\gamma = \sum_{i,j \in S} k_i \gamma_j \pmod{q}$$

$$kx = \sum_{i,j \in S} k_i w_j \pmod{q}$$

- (2) P_i, P_j のすべてのペアで加法から乗法へ変換する 2 つの share 変換プロトコル MtA, MtAwc を動かし、各プレイヤーは $k\gamma, kx$ の share を手に入れる。

– MtA

P_i は k_i を、 P_j は γ_j を、それぞれ α_{ij}, β_{ij} に変換し、 P_i は α_{ij} 、 P_j は β_{ij} を得る。

$$k_i \gamma_j = \alpha_{ij} + \beta_{ij}$$

これを各 P_i, P_j のペア全てで行うと、 P_i は $\alpha_{i1}, \dots, \alpha_{it'}, \beta_{1i}, \dots, \beta_{t'i}$ を得ることになる。 P_i は δ_i を次のように設定する。この δ_i が $k\gamma = \sum_{i \in S} \delta_i$ の (t', t') -share である。

$$\delta_i = k_i \gamma_i + \sum_{i \neq j} \alpha_{ij} + \sum_{j \neq i} \beta_{ji}$$

– MtAwc

P_i は k_i を、 P_j は w_j を、それぞれ μ_{ij}, ν_{ij} に変換し、 P_i は μ_{ij} 、 P_j は ν_{ij} を得る。

$$k_i w_j = \mu_{ij} + \nu_{ij}$$

これを各 P_i, P_j のペア全てで行うと、 P_i は $\mu_{i1}, \dots, \mu_{it'}, \nu_{1i}, \dots, \nu_{t'i}$ を得ることになる。 P_i は σ_i を次のように設定する。この σ_i が $kx = \sum_{i \in S} \sigma_i$ の (t', t') -share である。

$$\sigma_i = k_i w_i + \sum_{i \neq j} \mu_{ij} + \sum_{j \neq i} \nu_{ji}$$

- (3) 各プレイヤーは δ_i を公開することで、 $\delta = \sum_{i \in S} \delta_i = k\gamma$ を構築し、 $\delta^{-1} \pmod{q}$ を計算する。

- (4) 署名 (r, s) の一部 r を求める。プレイヤー P_i は

D_i を公開し、 $\Gamma_i = g^{\gamma_i}$ とする。この際、 P_i は γ_i を知っていることをゼロ知識証明 ($\text{ZKP}\{\gamma_i \mid g^{\gamma_i}\}$) する。その後、全てのプレイヤーで以下の計算をする。

$$\begin{aligned} R &= \left[\prod_{i \in S} \Gamma_i \right]^{\delta^{-1}} = g^{(\sum_{i \in S} \gamma_i) k^{-1} \gamma^{-1}} \\ &= g^{\gamma k^{-1} \gamma^{-1}} = g^{k^{-1}} \end{aligned}$$

ハッシュ関数 H を用いて、 $r = H(R)$ を計算し、閾値署名 (r, s) の一部である r を得る。

- (5) 署名 (r, s) の残り部分 s を求める。各プレイヤー P_i は s の (t', t') -share となる s_i を次の様に設定する。

$$\begin{aligned} s_i &= m k_i + r \sigma_i \\ \sum_{i \in S} s_i &= m \sum_{i \in S} k_i + r \sum_{i \in S} \sigma_i \\ &= m k + r k x \\ &= k(m + r x) = s \end{aligned}$$

また、単に s_i を各プレイヤーたちが公開し s を再構築するだけでは、参加者の中に攻撃者がいた場合、わざと誤った値で計算させることにより誤った s が出力される。当然この s は無効な署名であるが、その際に計算結果から攻撃者が他の s_i についての情報をわずかでも入手する可能性がある。これを防ぐために、これ以降のステップではランダムな値を用いて一度 s_i を隠す操作を行っている。

- (5A) 各プレイヤー P_i はランダムに $l_i, \rho_i \in \mathbb{Z}_q$ ($l = \sum_i l_i, \rho = \sum_i \rho_i$) を選択し、以下を計算する。

$$V_i = R^{s_i} g^{l_i}, A_i = g^{\rho_i}, [\hat{C}_i, \hat{D}_i] = \text{Com}(V_i, A_i)$$

ここでは、 V_i, A_i 両方に不正がないことを示すために Com に 2 つをまとめてインプットしている。その後 $\hat{C}_i$ を公開する。

- (5B) P_i は $\hat{D}_i$ を公開することで、 V_i, A_i に不正がないことを示し。また自身が s_i, l_i, ρ_i を知っていることをゼロ知識証明する。このゼロ知識証明ができなければこのプレイヤー P_i を信用できないことになるので、プロトコルを中止する。また、以下のように定義する。

$$\begin{aligned}
V &= g^{-m} y^{-r} \prod_{i \in S} V_i \\
&= g^{-m} y^{-r} R^s g^l = g^l y^{-r} g^{-m} (g^{k^{-1}})^s \\
&= g^l y^{-r} g^{-m} (g^{k^{-1}})^{k(m+xr)} \\
&= g^l g^{xr} y^{-r} = g^l y^r y^{-r} \\
&= g^l \\
A &= \prod_{i \in S} A_i = g^p
\end{aligned}$$

(5C) 各プレイヤー P_i は以下を計算する.

$$U_i = V^{\rho_i}, T_i = A^{l_i}, [\tilde{C}_i, \tilde{D}_i] = \text{Com}(U_i, T_i)$$

ここでは, U_i, T_i 両方に不正がないことを示すために Com に 2 つをまとめてインプットしている. その後 $\tilde{C}_i$ を公開する.

(5D) P_i は $\tilde{D}_i$ を公開することで, U_i, T_i に不正がないことを示す. ここで以下の計算が合わなければプロトコルを中止する.

$$\prod_{i \in S} U_i = \prod_{i \in S} T_i$$

(5E) ここまで行い, 問題がなければ各プレイヤー P_i は s_i を公開し, $s = \sum_{i \in S} s_i$ を計算し. 署名 (r, s) を得る.

署名検証

署名に参加していない検証者が, 生成された閾値署名 (r, s) が有効な署名かどうかを以下の式を用いて検証する. この際, 検証者は署名に参加していないので, 各プレイヤーの乱数などの秘密にされている値については当然知らない. 従って, 検証式は公開鍵などの公開されている情報のみで構成される必要があることに注意する.

$$R' = g^{m \cdot s^{-1}} \pmod{q} y^{r \cdot s^{-1}} \pmod{q}$$

$H(R') = r$ が成立すれば検証成功, この (r, s) は有効な署名であることがわかる.

実際, 以下の計算から, 正しい署名であれば検証を通ることが確認できる.

$$\begin{aligned}
s &= k(m+xr) \\
s^{-1} &= k(m+xr)^{-1} \\
r &= g^{k^{-1}} \\
g^{m \cdot s^{-1}} y^{r \cdot s^{-1}} &= g^{m \cdot s^{-1}} (g^x)^{r \cdot s^{-1}} \\
&= g^{(m+xr) \cdot s^{-1}} \\
&= g^{k^{-1}} \\
&= R
\end{aligned}$$

3.3 Gennaro らの方式の考察

Gennaro らの方式は DSA をベースにしている. 署名プロトコル中では, それぞれの秘密鍵 w_i にそれぞれ値の異なる乱数 k_i をかける必要がある. つまり, MtA を使わずに DSA の署名式 s_i に単純に w_i を適用すると, $s_i = k_i(m + rw_i)$ となる. $s = \sum_{i \in S} s_i$ とする. w_i 自体はあらかじめラグランジュ係数を作作用させているため, w_i 全てを加算すると秘密鍵 x が回復できる. しかし今 s_i 内の w_i には k_i がかけられている. このため, $s = \sum_{i \in S} s_i$ としても, k_i によりラグランジュ補間の計算がずれ, うまく秘密鍵を回復することができず, 署名を生成することができない.

そこで Gennaro らは MtA プロトコルを用いて, share w_i と乱数 k_i の積を和に変換することで, ラグランジュ補間を容易に実行できるようにしていると考えられる. しかし MtA プロトコル内で, 各プレイヤーに対し準同型暗号の計算や, 他プレイヤーとの share の計算を行うために他プレイヤー人数にあたる $t-1$ 回の通信やゼロ知識証明を強いることになり, 計算量や通信回数が多いという課題がある.

そこで本論文では, MtA プロトコルを用いない閾値署名方式を実現するために, 以下の改良を行った.

署名式の変更: MtA が必要になるのは, 秘密鍵と乱数の積が署名式にあることが原因である. 実際, DLP に基づく署名式は 6 パターンに変形できることから, 秘密鍵と乱数の積がない署名式 [1] へと変更する.

3.4 ElGamal 署名の変形

[1] に記載されている ElGamal 署名を変形した署名式について説明する. 本論文ではこの署名式を参考になっている.

- (1) 署名者 P は秘密鍵 $x \in \mathbb{Z}_q$ を取得し, 公開鍵 $y = g^x \pmod{p}$ を計算する.
- (2) P は $k \in_R \mathbb{Z}_q^*$ に対して, $r = g^k \pmod{p}$ を計算する. このとき, $\gcd(r, q) = 1$ でなければ, k を取得しなおす.
- (3) P はメッセージ M について, $s = x + kr \cdot \text{hash}(r, M) \pmod{q}$ を計算し, M に対する署名 (r, s) を得る.
- (4) $g^s = yr^{r \cdot \text{hash}(r, M)} \pmod{p}$ を計算して, 署名を検証する.

4. 提案方式

本章では提案プロトコルの詳細について説明する.

4.1 方針

3 章から, Gennaro らの方法において MtA の計算は, 秘密鍵となる w_i と乱数 k_i 積を扱いやすくし, ラグランジュ補間による秘密鍵の回復を容易にするために重要である.

そこで, MtA を削除するために, 本提案方式では以下の変更を行う.

署名式の変更：DSA の署名式に従い、MtA を利用せずに原始的に秘密鍵の share に置き換えるだけでは、うまく署名を生成できないため、元とする署名式そのものを DSA のものから以下の形に変更する。

$$s_i = x_i + k_i Rr, r = H(R, m)$$

鍵生成は Gennaro らの方式と同様に行う。

4.2 提案プロトコル

使用する記号は以下の通りである。

- $\mathbb{G}$: 巡回群.
- g : 巡回群 $\mathbb{G}$ の生成元.
- q : 巡回群 $\mathbb{G}$ の位数.
- 鍵生成プロトコル

署名生成に必要な秘密鍵を秘密分散し、その share を n 人に分配する。この share を持つ n 人の中から、閾値となる t 人よりも多いプレイヤーが集まれば、ラグランジュの補完公式を用いて share から秘密鍵を復元することができ、署名生成が可能となる。

(1) 各プレイヤー P_i はランダムに $u_i \in \mathbb{Z}_q$ をとり、 $[KGC_i, KGD_i] = \text{Com}(g^{u_i})$ を計算、 KGC_i を公開する。

(2) 各プレイヤー P_i は KGD_i を公開し、 $y_i = g^{u_i}$ とする。署名の公開鍵を $y = \prod_i y_i$ とすると、それに対応する署名の秘密鍵は $x = \sum_i u_i$ である。
 P_i は Feldman の (t, n) -VSS プロトコルで u_i を n 分割し、各プレイヤーそれぞれに配布。各プレイヤーは受け取った n 個の share を合計することで、秘密鍵 x の秘密分散 x_i を得る。

(3) 各プレイヤーは、自身が x_i を知っていることをゼロ知識証明 ($\text{ZKP}\{x_i \mid g^{x_i}\}$) する。

- 署名生成プロトコル

(1) S を署名プロトコルに参加するプレイヤーの集合とする。ここで S の要素数 t' について、 $t < t' \leq n$ と想定する。また、この方式の (t, n) -secret sharing では、 t 次多項式を用いているため、ラグランジュの補完公式から、分散させた秘密 x 及び k を求めるには (t, n) である必要はなく (t', t') で十分である。従って、そのラグランジュの補完公式を満たすように適切なラグランジュ係数 $\lambda_{i,S}$ を用いて各プレイヤーは自身の (t, n) -share である x_i を (t', t') -share となる $w_i = (\lambda_{i,S})(x_i)$ に変換する。
 $x = \sum_{i'} w_i$ である。

(2) S 内の各プレイヤー P_i は、ランダムに $k_i \in \mathbb{Z}_q$ をとる。ここで、 $k = \sum_{i'} k_i$ とする。その後、 g^{k_i} を公開し、 k_i を知っていることをゼロ知識証明 ($\text{ZKP}\{k_i \mid g^{k_i}\}$) する。その後、全てのプレイヤーが以下の計算をする。

$$R = \prod_{i \in S} g^{k_i} = g^{\sum k_i} = g^k$$

(3) ハッシュ関数 H を用いて、 $r = H(m, R)$ を計算する。

(4) 各プレイヤー P_i は s の (t', t') -share となる s_i を次の様に設定する。

$$s_i = w_i + k_i r R$$

(5) 最後に、以下の計算を行う。

$$\begin{aligned} s &= \sum_{i \in S} s_i = \sum_{i \in S} w_i + r R \cdot \sum_{i \in S} k_i \\ &= x + krR \end{aligned}$$

メッセージ m に対する署名として、 (R, s) が得られる。

- 署名検証

署名に参加していない検証者が、生成された閾値署名 (R, s) が有効な署名かどうかを以下の式を用いて検証する。

$$g^s = y \cdot R^{R \cdot H(R, m)}$$

実際、以下の計算から、正しい署名であれば検証を通ることが確認できる。

$$\begin{aligned} g^s &= g^{x+krR} \\ y \cdot R^{R \cdot H(R, m)} &= g^x \cdot (g^k)^{R \cdot H(R, m)} \\ &= g^x \cdot g^{krR} \\ &= g^{x+krR} \end{aligned}$$

4.3 安全性

この提案方式の安全性について、以下の定理を示すことでその安全性を検証する。

3.4 節で示した ElGamal 署名を変形した署名式を MEIG と表す。

定理 1 MEIG が偽造不可能なとき、本論文の閾値署名方式も偽造不可能である。

証明 1 この定理 1 について、対偶を証明することで定理 1 を証明する。つまり、本論文の閾値署名の偽造が可能なとき、MEIG も偽造が可能であることを証明する。

– $A : P_2, \dots, P_{t+1}$ の秘密情報 (w_i など) を把握している。

– $F :$

* P_1 の役を行うが、 P_1 の秘密鍵の share は有していない。

* A の有する情報を有している。

この F が MEIG の偽造署名を生成できることを証明する。

– 鍵生成

- (1) F は $u_1 \in \mathbb{Z}_q$ をとり、鍵生成プロトコルに従い、 $P_2, \dots, P_n$ とともに公開鍵 y を生成する。
 (2) 次に F は以下の計算を行う。

$$\hat{y}_1 = y \cdot \prod_{i=2}^n y_i^{-1} = g^{x_1}$$

得られた $\hat{y}_1$ に対応する秘密鍵 x_1 の値を F は把握していない。この公開鍵 $\hat{y}_1$ について、MEIG の署名偽造を試みる。

偽造署名生成

公開鍵 $\hat{y}_1$ を用いて、MEIG の署名偽造を試みる。仮定より、公開鍵 y における本署名方式の偽造署名を (R, s) とする。

$$R = g^k, r = H(R), s = \sum_{i=1}^{t+1} s_i = \sum_{i=1}^{t+1} w_i + kRr$$

また、ラグランジュ係数 $\lambda_{1,S}$ を用いることで、 $\hat{y}_1$ について、以下のように計算できる。

$$\hat{y}_1^{\lambda_{1,S}} = (g^{x_1})^{\lambda_{1,S}} = g^{\hat{w}_1}$$

F は A の有する情報も知っており、 $P_2, \dots, P_{t+1}$ の秘密鍵の share である $w_2, \dots, w_{t+1}$ をすべて把握しているため、以下の計算を行うことができる。

$$\begin{aligned} \hat{s}_1 &= s - \sum_{i=2}^{t+1} w_i = \sum_{i=1}^{t+1} w_i + kRr - \sum_{i=2}^{t+1} w_i \\ &= \hat{w}_1 + kRr \end{aligned}$$

得られた $\hat{s}_1$ について、これは MEIG において公開鍵 $g^{\hat{w}_1}$ 、乱数 k としてメッセージ m に施す署名に等しくなる。 F は公開鍵 $g^{\hat{w}_1}$ に対応する秘密鍵 $\hat{w}_1$ の値を知らないため、 $(R, \hat{s}_1)$ は m に対する偽造署名である。

以上のことから、メッセージ m に対する本閾値署名が偽造可能であれば、MEIG も偽装可能である。対偶が示せたことから、定理 1 についての証明を終える。

5. 性能評価

本章では提案プロトコルの実行結果について記載する。

各プレイヤー 1 人あたりの冪演算回数および通信回数を、Gennaro らの研究と比較する。本方式と Gennaro らの方式は同一の手順で鍵生成を行うため、鍵生成プロトコルでの計算量の比較は省略する。

計算量

Gennaro らの方式

- * $(g^{x_i})^{\lambda_{i,S}}$ で冪演算 1 回
- * g^{γ_i} で冪演算 1 回
- * MtA 内で Pailler 暗号化 1 回、 S のすべての参加

者の暗号を復号するため、復号が $t' - 1$ 回

- * MtAwc 内で同様に Pailler 暗号化 1 回、復号 $t' - 1$ 回
- * δ^{-1} で逆元計算 1 回
- * $R = [\prod_{i \in S} \Gamma_i]^{\delta^{-1}}$ の計算で冪演算 1 回

本閾値署名方式

- * $(g^{x_i})^{\lambda_{i,S}}$ で冪演算 1 回
- * g^{k_i} で冪演算 1 回
- * $R = \prod_{i \in S} g^{k_i} = g^{\sum k_i}$ の計算で冪演算 1 回

通信回数

Gennaro らの方式

- * MtA 内で $1 + (t' - 1) = t'$ 回
- * MtAwc 内で $1 + 2(t' - 1) = 2t' - 1$ 回
- * g^{γ_i} で 1 回

本閾値署名方式

- * g^{k_i} で 1 回

通信回数

Gennaro らの方式

- * g^{γ_i} の公開で 1 回
- * MtA で互いに share を交換するために 1 回 (Alice 側). S のすべての参加者と交換するため、 $t' - 1$ 回 (Bob 側)
- * MtAwc についても同様に 1 回および $t' - 1$ 回
- * δ_i の公開で 1 回
- * s_i の公開で 1 回

本閾値署名方式

- * g^{k_i} の公開で 1 回
- * s_i の公開で 1 回

通信量

$\mathbb{Z}_q$ 上のデータ量を高々 a 、 $\mathbb{G}$ 上のデータ量を高々 b として、通信回数より通信量を計算する。

Gennaro らの方式

- * g^{γ_i} の公開で b
- * MtA 内で a (Alice 側). S のすべての参加者と交換するため、 $a \cdot (t' - 1) = a(t' - 1)$ (Bob 側)
- * MtAwc についても同様に a および $a(t' - 1)$
- * δ_i の公開で a
- * s_i の公開で a

1 人あたり計 $a(2t' + 2) + b$ の通信量が必要となる。

本閾値署名方式

- * g^{k_i} の公開で b
- * s_i の公開で a

1 人あたり計 $a + b$ の通信量が必要となる。

以上より、冪演算回数を削減できており、また通信回数においては、MtA プロトコルを削除した分、大幅に削減することができている。

6. まとめ

本論文では、DSA をベースとしない閾値署名方式を提案した。鍵生成プロトコルは既存の閾値署名と同様に行う一方で、署名式を DSA ベースのものから、ElGamal 署名を変形した署名式へと変更した。これにより DSA ベースの閾値署名方式に必要であった、秘密鍵と乱数の積を和に変換する MtA プロトコルを削除することが可能となった。それに伴い、MtA プロトコルを用いるにあたって強いられていた準同型暗号の計算や通信を削除できるため、大幅に通信回数や計算量を削減することに成功している。

6.1 参考文献・謝辞

謝辞 本研究の一部は文部科学省「Society5.0 に対応した高度技術人材育成事業成長分野を支える情報技術人材の育成拠点の形成 (enPiT)」さらに文部科学省の平成 30 年度「Society 5.0 実現化研究拠点支援事業」の助成を受けています。

参考文献

- [1] Mike Burmester, Yvo Desmedt, Hiroshi Doi, Masahiro Mambo, Eiji Okamoto, Mitsuru Tada, and Yuko Yoshi-fuji. A structured elgamal-type multisignature scheme. In *International Workshop on Public Key Cryptography*, pages 466–483. Springer, 2000.
- [2] Shirow Mitomi and Atsuko Miyaji. A general model of multisignature schemes with message flexibility, order flexibility, and order verifiability. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 84(10):2488–2499, 2001.
- [3] Rosario Gennaro and Steven Goldfeder. Fast multiparty threshold ecDSA with fast trustless setup. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1179–1194, 2018.
- [4] Yehuda Lindell and Ariel Nof. Fast secure multiparty ecDSA with practical distributed key generation and applications to cryptocurrency custody. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1837–1854, 2018.
- [5] Guilhem Castagnos, Dario Catalano, Fabien Laguillaumie, Federico Savasta, and Ida Tucker. Bandwidth-efficient threshold ec-dsa. In *PKC 2020-23rd IACR International Conference on Practice and Theory of Public-Key Cryptography*, pages 266–296, 2020.
- [6] Ran Canetti, Rosario Gennaro, Steven Goldfeder, Nikolaos Makriyannis, and Udi Peled. Uc non-interactive, proactive, threshold ecDSA with identifiable aborts. In *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, pages 1769–1787, 2020.
- [7] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
- [8] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pages 427–438. IEEE, 1987.
- [9] Rosario Gennaro, Steven Goldfeder, and Arvind

Narayanan. Threshold-optimal dsa/ecdsa signatures and an application to bitcoin wallet security. In *International Conference on Applied Cryptography and Network Security*, pages 156–174. Springer, 2016.