Copyright ©2024 The Institute of Electronics,
Information and Communication Engineers

SCIS 2024 2024 Symposium on
Cryptography and Information Security
Nagasaki, Japan, Jan. 23 – 26, 2024
The Institute of Electronics,
Information and Communication Engineers

連結不可能選択的開示のためのデジタル署名アルゴリズムの比較
Comparison of Digital Signature Algorithms for Unlinkable Selective Disclosure

堤 真聖 * 渡邉 健 * 佐古 和恵 *
Masato Tsutsumi Ken Watanabe Kazue Sako

あらまし 連結不可能な選択的開示は, ユーザのプライバシーを保護する概念として, Verifiable Credential や匿名認証 (Anonymous Authentication) などの技術において重要となる. そして, 連結不可能な選択的開示を実現する署名アルゴリズムの代表例として, CL 署名・BBS+署名・PS 署名の 3 つが挙げられる. これらの署名アルゴリズムは, メッセージの集合に対して署名者が 1 つの署名を施す. 署名されたメッセージの全集合を開示しなくても, ゼロ知識証明を活用して開示された部分集合に署名が付与されていることが検証できる. この研究ではそれらのアルゴリズムに対して, 鍵生成・署名・証明の計算コスト, 秘密鍵・公開鍵・署名・証明のデータサイズ, および実行時間を比較する. また, 署名するメッセージの要素数と, 開示するメッセージの要素数に応じて, ゼロ知識証明の生成や検証にかかる時間がどのように変化するかを測定し, 効率のよい運用について考察した.

キーワード 選択的開示, 匿名認証, Verifiable Credential, CL 署名, BBS+署名, PS 署名

1 はじめに

近年, マイナンバーカードやパスポートなどの電子証明書を用いた本人確認のデジタル化[1] がますます進んでいる. 同時に, セキュリティやプライバシーに対する不安の声も大きく[2] , それがデジタル化の妨げの原因の 1 つになっている.

連結不可能な選択的開示は, 個人情報のプライバシーを保護しながら必要な情報だけを開示する概念として, 上記の課題を解決するために重要となる. たとえば, 長崎で開催される学会に参加するために受付で氏名を証明する必要があるとき, 選択的開示の機能があれば, 参加者は, 政府から発行された身分証に記載された氏名, 住所, 生年月日の中から氏名だけを選んで開示することができる. さらに, 連結不可能の機能によって, お土産屋で地酒を購入するために同じ身分証から年齢のみを開示したときに, それが学会で開示した氏名と同じ身分証であることを紐付けることができない. これにより, どこで何をしたかという情報はバラバラに保たれる. 逆に, 連結不可能の機能がない場合, 様々な場所でこれまで開示した情報がパズルのように繋がってしまう恐れがある. これは, 上記の例では, 学会の受付で開示した「氏名」と, お土産屋で開示した「年齢」, あるいは他の場所で開示した「住所」や「生年月日」といった個人情報が全て繋がり, 開示の度に多くの個人情報を晒すことになる.

連結不可能な選択的開示が必要とされる技術として, Verifiable Credential と匿名認証 (Anonymous Authentication) などが挙げられる. Verifiable Credential は, 政府などの信頼機関が発行する身分証やワクチン接種証明書のような証明書のデータフォーマットであり, W3C などが主導で標準化[3] が進んでいる. これは, 物理的な身分証の代わりに, スマートフォンに表示されるデジタル身分証として使用することができる. 連結不可能な選択的開示によって, そのデジタル身分証に記載された個人情報の中から「氏名」だけを開示し, そのほかの個人情報を隠すことができる. 匿名認証 (Anonymous Authentication) は, 自身の氏名や住所などの本人を特定する情報を明かすことなく, 特定の資格や権利を保有していることを証明し, サービスやリソースにアクセスする手法である. 例えば, 成人用オンラインゲームをする際, 政府によって発行された成人者の資格を用いて, 氏名などの身元は開示せずに資格を保有していることを証明し, ゲームをプレイすることができる.

連結不可能な選択的開示を実現する方式として, 電子署

* 早稲田大学大学院基幹理工学研究科
[1] https://www.nichigosho.net/topics/images/20200227.pdf
[2] https://prtimes.jp/main/html/rd/p/000000028.000059855.html
[3] https://www.w3.org/TR/vc-data-model-2.0/

名とゼロ知識証明[4]を用いる方式が挙げられる．この方式ではユーザが保有している属性の集合 $(m_1, m_2, \ldots, m_l)$ に対して，信頼機関が一つのデジタル署名を施し，ユーザに送付する．ユーザは属性の集合のうち，(m_1, m_4) といった属性の部分集合が署名されたことを，それ以外のメッセージと署名を隠しながら証明するゼロ知識証明を提示する．受付スタッフなどの検証者は，開示された属性をもとにゼロ知識証明を検証する．

この方式のための署名アルゴリズムの代表例として，CL 署名・BBS+署名・PS 署名の 3 つの方式が挙げられる．まず，CL 署名 [1] は Verifiable Credential の形式の一つである Hyperledger AnonCreds v1 [5] の署名アルゴリズムに採用されている[6]．BBS+署名 [3] は MATTR，IETF[7]や Decentralized Identity Foundation[8]などの組織によって仕様策定や議論が進められている．また，実装例としては，docknetwork のライブラリ[9] などが挙げられる．PS 署名 [6] は，hyperledger AnonCreds v2 に[10] 採用されている．また，実装例としては，docknetwork のライブラリ[11] などが挙げられる．

現状，これら 3 つを比較した研究 [4, 6] では，署名生成・検証の計算コストと，署名・証明のデータサイズが比較されている．しかし，連結不可能な選択的開示にあたり，ユーザが開示の際にゼロ知識証明を生成するため，証明の生成・検証にかかる時間も重要となる．また，Verifiable Credential はハードウェアウォレットや物理カードなどに保存されることも考えられるため，鍵のサイズも重要となる．これらの項目の比較が抜けていることから，現状は，連結不可能な選択的開示にどの署名アルゴリズムを用いるかを判断することが難しい．

本研究では，CL 署名，BBS+署名，および PS 署名のアルゴリズムについて比較を行う．最初に，鍵生成，署名とその検証，証明生成とその検証にかかる計算コストについて比較する．次に，秘密鍵，公開鍵，署名，証明のデータサイズについて比較を行なう．最後に，OSS のベンチマークテストを用いた実行時間の比較を行なう．これらの比較より，それぞれの署名方式の特徴の整理，ストレージ制約下における運用といった考察を行なう．

本論文の構成は以下の通りである．第 2 章では，各署名方式のアルゴリズムを記す．第 3 章では，比較手法を記す．第 4 章では，比較結果を記す．第 5 章では，比較結果による考察を記す．第 6 章では，本論文のまとめを記す．

[4] ある事柄を知っているという事実を，その事実以外の知識を与えずに証明する技術
[5] https://www.hyperledger.org/projects/anoncreds
[6] https://hyperledger.github.io/anoncreds-spec/
[7] https://datatracker.ietf.org/doc/draft-irtf-cfrg-bbs-signatures/
[8] https://github.com/decentralized-identity/bbs-signature
[9] https://github.com/docknetwork/crypto/tree/main/bbs_plus
[10] https://github.com/hyperledger/anoncreds-v2-rs
[11] https://github.com/docknetwork/crypto/tree/main/coconut

2 準備

2.1 記法

- QR_n は n を法とする平方剰余の集合を指す．
- $a \xleftarrow{\$} \mathbb{Z}$ は群 Z の中からランダムな要素を選定して変数 a に代入することを指す．
- e は $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ の双線型写像 (ペアリング演算) を指す．
- $\mathbb{Z}_p$ は要素 $(0, 1, \ldots, p-1)$ からなる整数の集合を指す．
- $\mathbb{Z}_p^*$ は要素 $(1, 2, \ldots, p-1)$ のうち p と互いに素な整数の集合を指す．
- $(m_1, \ldots, m_l)$ は要素数 l 個のメッセージからなる集合を指す．
- $PK\{(w) : s\}$ は Proof of Knowledge を指し，w を秘密とする値，s を証明する命題とする．
- $SPK\{(w) : s\}(nonce)$ は Signature based on a proof of knowledge を指し，w を秘密とする値，s を証明する命題，$nonce$ は署名の対象とする．
- $\langle h \rangle$ は h から生成される群を指す．
- pk は署名者の公開鍵を指す．
- sk は署名者の秘密鍵を指す．
- k_c はセキュリティパラメータを指す．
- $|a|$ はある変数 a のビットサイズを指す．
- L はメッセージのインデックスの集合を指す．

2.2 CL 署名

CL 署名 [1] は，strong-RSA 仮定に基づくデジタル署名方式で，メッセージのコミットメントベクトルに署名を施すことで選択的開示を実現する．l 個のメッセージ $(m_1, \ldots, m_l)$ に対する CL 署名のアルゴリズムを以下に示す．k を署名のセキュリティパラメータ，k_c をコミットメントのセキュリティパラメータとする．$|m|$ をメッセージ空間とする．

Algorithm2.1. CLGenKey(l, k, k_c)

1. k ビットの素数 p, q と，k_c ビットの素数 p_c, q_c を選定．
2. $n := pq$, $n_c = p_c q_c$ を計算．
3. $(a_1, \ldots, a_l, b, c) \xleftarrow{\$} QR_n$ を選定．

4. $h \xleftarrow{\$} QR_{n_c}, g \xleftarrow{\$} \langle h \rangle$ を選定.
5. $sk := (p, q, p_c, q_c)$,
   $pk := (n, a_1, \ldots, a_l, b, c, n_c, g, h)$ とする.
6. $(sk\ ,\ pk)$ を出力.

Algorithm2.2. CLSign$(sk, (m_1, ..., m_l))$

1. 素数 $e(|e| = |m| + 2)$ を選定.
2. $s(|s| = 3k + |m|)$ を選定.
3. $v = (\Pi_{i \in l} a_i^{m_i} b^s c)^{\frac{1}{e}} \, mod\, n$ を計算.
4. $\sigma = (s, e, v)$ を出力.

Algorithm2.3. CLVerSig$(pk, (m_1, ..., m_l), \sigma)$

1. $v^e \stackrel{?}{\equiv} \Pi_{i \in l} a_i^{m_i} \cdot b^s \cdot c \, (mod\, n)$ をチェック.
2. $e > 2^{|e|-1}$ をチェック.

CL 署名におけるゼロ知識証明を用いた選択的開示のアルゴリズムを以下に示す. D は集合 L のうち開示メッセージのインデックスの集合を指す.

Algorithm2.4. CLGenProof$(pk, (m_1, ..., m_l), D, \sigma)$

1. $w, r_w \xleftarrow{\$} \mathbb{Z}_n$ を選定.
2. 以下を計算.
   (a) $C_x = g^{\sum m_i} h^{r_x} \, mod\, n_c$
   (b) $C_v = v g^w \, (mod\, n)$
   (c) $C_w = g^w h^{r_w} \, (mod\, n)$
3. 以下の証明を生成.
   $\pi \in PK\{(\alpha, \beta, \gamma, \epsilon, \xi, \sigma, \nu) :$
   $c \stackrel{?}{=} C_v^{\epsilon} (1/a)^{\epsilon} (1/b)^{\sigma} (1/g)^{\phi} \wedge$
   $C_w \stackrel{?}{=} g^{\nu} h^{\alpha} \wedge 1 \stackrel{?}{=} C_w^{\epsilon} (1/g)^{\phi} (1/h)^{\beta} \wedge$
   $C_x \stackrel{?}{=} g^{\xi} h^{\gamma} \wedge$
   $2^{e-1} < \epsilon < 2^e \wedge 2^{l_m - 1} < \xi < 2^{l_m}\}$
4. (π, C_x, C_v, C_w) を出力.

Algorithm2.5. CLVerProof$(PK, pk, \{m_i\}_{i \in D})$

1. π を検証.

2.3 BBS+署名

BBS+署名 [3] は, BBS グループ署名 (Group Signature)[5, 2] に由来するデジタル署名方式の一つで, メッセージの集合に一つの署名を施す. 署名サイズはメッセージの数ではなくセキュリティパラメータに依存する. この方式は署名の知識に関するゼロ知識証明 (zero knowledge proof of knowledge of a signature) をサポートし, メッセージを選択的に開示することができる. BBS+方式は双線形ペアリングを用いて署名の検証を行なう.

l 個のメッセージ $(m_1, \ldots, m_l)$ に対する BBS+署名のアルゴリズムを以下に示す. $\mathbb{G}_1$, $\mathbb{G}_2$ は位数 p をもつ楕円曲線の群, g_1, g_2 はそれぞれ $\mathbb{G}_1$, $\mathbb{G}_2$ の生成元を指す. BBS+署名は公開鍵を G_1 上の点とするか G_2 上の点とするか選ぶことができる. 以下のアルゴリズムでは公開鍵を G_2 上の点とする例を示す. BBS+署名における署名パラメータ $h_1, \ldots, h_l$ は, 公開情報として公開しなくても、シードだけ公開しておいて、必要なタイミングで再計算することができる[12] .

Algorithm2.6. BBS+GenKey$(\mathbb{G}_1, \mathbb{G}_2, g_1, g_2, p, l)$

1. $l + 1$ 個の群要素 $(h_0, \ldots, h_l) \xleftarrow{\$} \mathbb{G}_1^{l+1}$ を選定.
2. 秘密鍵 $x \xleftarrow{\$} \mathbb{Z}_{2p}^*$ を選定し $X := g_2^x$ を計算.
3. $sk := (x), pk := (X, h_0, \ldots, h_l)$ とする.
4. $(sk\ ,\ pk)$ を出力.

Algorithm2.7. BBS+Sign$(sk, (m_1, ..., m_l))$

1. $e, s \xleftarrow{\$} \mathbb{Z}_p$ を選定.
2. $A := (g_1 \cdot h_0^s \cdot \Pi_{i \in l} h_i^{m_i})^{\frac{1}{x+e}}$ を計算.
3. $\sigma = (A, e, s)$ を出力.

Algorithm2.8. BBS+VerSig$(pk, (m_1, ..., m_l), \sigma)$

1. 以下をチェック.

$$e(A, X \cdot g_2^e) \stackrel{?}{=} e(g_1 \cdot h_0^s \cdot \Pi_{i \in L} h_i^{m_i}, g_2) \quad (1)$$

BBS+署名におけるゼロ知識証明を用いた選択的開示のアルゴリズムを以下に示す. D は集合 L のうち開示メッセージのインデックスの集合を指す.

Algorithm2.9. BBS+GenProof$(pk, (m_1, ..., m_l), D, \sigma)$

1. $b = g_1 \cdot h_0^s \cdot \Pi_{i \in l} h_i^{m_i}$ とおく.
2. $r_1 \xleftarrow{\$} \mathbb{Z}_p^*, r_2 \xleftarrow{\$} \mathbb{Z}_p$ を選定し $r_3 = \frac{1}{r_1}$ とおく.
3. 以下を計算.
   (a) $A' := A^{r_1}$
   (b) $\bar{A} := A'^{-e} \cdot b^{r_1} (= A'^x)$
   (c) $d := b^{r_1} \cdot h_0^{-r_2}$
   (d) $s' := s - r_2 \cdot r_3$
   (e) $\pi \in SPK\{(\{m_i\}_{i \notin D}, e, r_2, r_3, s') :$
   $\frac{\bar{A}}{d} = A'^{-e} \cdot h_0^{r_2} \wedge$
   $g_1 \cdot \Pi_{i \in D} h_i^{m_i} = d^{r_3} \cdot h_0^{-s'} \Pi_{i \notin D} h_i^{-m_i}$

[12] https://github.com/mattrglobal/bbs-signatures/blob/master/docs/ALGORITHM.md

}(*nonce*)

4. $(A', \bar{A}, d, \pi)$ を出力.

Algorithm2.10. BBS+VerProof($SPK, pk, \{m_i\}_{i\in D}, s'$)

1. $A' \stackrel{?}{\neq} 1_{\mathbb{G}_1}$ をチェック.
2. $e(A', X) \stackrel{?}{=} e(\bar{A}, g_2)$ をチェック.
3. π を検証.

2.4 PS 署名

PS 署名 [6] は, PS 仮定 [6] に基づく署名方式であり, BBS+署名と同様に双線形ペアリングを用いて署名の検証を行なう. l 個のメッセージ $(m_1, \ldots, m_l)$ に対する PS 署名のアルゴリズムを以下に示す. $\mathbb{G}_1$, $\mathbb{G}_2$ は位数 p をもつ楕円曲線の群, g_1, $\tilde{g}$ はそれぞれ $\mathbb{G}_1$, $\mathbb{G}_2$ の生成元を指す.

Algorithm2.11. PSGenKey($\mathbb{G}_1, \mathbb{G}_2, g_1, \tilde{g}, p, l$)

1. $l+1$ 個の秘密鍵 $(x, y_1, \ldots, y_l) \stackrel{\$}{\leftarrow} \mathbb{Z}_{2p}^*$ を選定.
2. $(\tilde{g}, \tilde{X}, \tilde{Y_1}, \ldots, \tilde{Y_l}) := (\tilde{g}, \tilde{g}^x, \tilde{g}^{y_1}, \ldots, \tilde{g}^{y_l})$ を計算.
3. $sk := (x, y_1, \ldots, y_l)$, $pk := (\tilde{g}, \tilde{X}, \tilde{Y_1}, \ldots, \tilde{Y_l})$ とする.
4. (sk, pk) を出力.

Algorithm2.12. PSSign($sk, (m_1, ..., m_l)$)

1. $w \stackrel{\$}{\leftarrow} \mathbb{Z}_p^*$ を選定し, $h := g_1^w$ を計算.
2. $(\sigma_1, \sigma_2) := (h, h^{x+\sum_{i\in L} y_i \cdot m_i})$ を計算.
3. $\sigma = (\sigma_1, \sigma_2)$ を出力.

Algorithm2.13. PSVerSig($pk, (m_1, ..., m_l), \sigma$)

1. $\sigma_1 \stackrel{?}{\neq} 1_{\mathbb{G}_1}$ をチェック.
2. $e(\sigma_1, \tilde{X} \cdot \Pi_{i\in L} \tilde{Y_i}^{m_i}) \stackrel{?}{=} e(\sigma_2, \tilde{g})$ をチェック.

PS 署名におけるゼロ知識証明を用いた選択的開示のアルゴリズムを以下に示す. D は集合 L のうち開示メッセージのインデックスの集合を指す.

Algorithm2.14. PSGenProof($pk, (m_1, ..., m_l), D, \sigma$)

1. $r, t \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ を選定.
2. 以下を計算.
   (a) $\sigma_1' := \sigma_1^r$
   (b) $\sigma_2' := (\sigma_2 \cdot \sigma_1^t)^r$
   (c) $\pi \in SPK\{(\{m_i\}_{i\notin D}, t) : e(\sigma_1', \tilde{X}) \cdot \Pi_{i\in D} e(\sigma_1', \tilde{Y_i})^{m_i} \cdot e(\sigma_1', \tilde{g})^t \stackrel{?}{=} e(\sigma_2', \tilde{g})\}(nonce)$
3. $(\sigma_1', \sigma_2', \pi)$ を出力.

Algorithm2.15. PSVerProof($SPK, pk, \{m_i\}_{i\in D}, \sigma'$)

1. π を検証.

3 比較手法

3.1 比較項目

本研究で比較する項目を表 1 に示す. 計算コスト, データサイズ, ベンチマークの 3 種類に大別する. ベンチマークの項目では, 後述するソフトウェアを実行し, 計算時間の計測を行う.

表 1: 比較項目一覧

大項目	小項目	説明
計算コスト	GenKey Cost	鍵の生成
	GenSig Cost	署名の生成
	VerSig Cost	署名の検証
	GenProof Cost	証明の生成
	VerProof Cost	証明の検証
データサイズ	SecKey Size	秘密鍵
	PubKey Size	公開鍵
	Sig Size	署名データ
	Proof Size	証明データ
ベンチマーク	GenSig Time	署名生成時間
	VerSig Time	署名検証時間
	GenProof Time	証明作成時間
	VerProof Time	証明検証時間

3.2 マシンスペック

ベンチマークで用いたマシンのスペックを以下に示す. Macbook Pro 13-inch, M1 2020, 16GB Memory, macOS Ventura 13.5.2, 512GB Storage.

3.3 ベンチマークで用いる OSS

CL 署名のプログラムは hyperledger/anoncreds-clsignatures-rs [13] を使用する. 言語は Rust で実装されており, 暗号プリミティブは amcl0.2 [14] を使用している.

BBS+署名と PS 署名のプログラムは docknetwork/crypto [15] を使用する. 言語は Rust で実装されており, 暗号プ

[13] https://github.com/hyperledger/anoncreds-clsignatures-rs
[14] https://docs.rs/amcl/0.2.0/amcl/
[15] https://github.com/docknetwork/crypto

リミティブは arkworks-rs/algebra [16] の ark_ff, ark_ec, ark_std, ark_serialize を用いている.

3.4 ベンチマーク実行手順

ベンチマーク実行手順を以下に示す.
hyperledger/anoncreds-clsignatures-rs, docknetwork/crypto のどちらの OSS も, 'cargo bench'のコマンドを実行することでベンチマークを行なうことができる. Cargo [17] は Rust パッケージマネージャであり, 'cargo bench'コマンド[18] は, Rust ソフトウェアのコンパイルとベンチマークの実行を行なうコマンドである.

hyperledger/anoncreds-clsignatures-rs に関しては, 固定なメッセージ要素数から 1 つの署名のみを実行する実装から, 複数の異なるメッセージ要素数パターンにおけるいくつかの署名を同時に実行するよう修正したブランチ[19] を実行する.

CL 署名のベンチマーク実行すると, 各プロセスにおいて, 100 回繰り返した実行時間の合計が以下のように出力される.

Log1. CL 署名のベンチマーク実行ログ

```
    Starting for attr size 2...
Generate credential definition is 1.85s
Generate registry for 10000 is 33.56ms
Generate tails for 10000 is 5.65s
Issuance time for 100 is 6.89s
Verify Signature time for 100 is 9.70s
Total witness generation time for 100 is 2.04s
Total witness update time for 100 is 67.58µs
Total proving time for 100 is 4.00s
Proof gen time for 100 is 6.04s
Verif time for 100 is 3.41s
...
```

BBS+/PS 署名のベンチマーク実行すると, 各プロセスにおいて, 100 回繰り返した実行時間の平均が以下のように出力される.

Log2. BBS+/PS 署名のベンチマーク実行ログ

```
    BBS+ signing/2 time: [623.70 µs 624.79 µs
626.05 µs]
change: [+0.5889% +0.9651% +1.3956%] (p =
0.00 ¡ 0.05)
Change within noise threshold.
Found 12 outliers among 100 measurements
(12.00%)
1 (1.00%) low mild
4 (4.00%) high mild
7 (7.00%) high severe
BBS+ signing/4 time: [653.45 µs 655.79 µs 659.06
µs]
change: [+1.7464% +2.5686% +3.5560%] (p =
0.00 ¡ 0.05)
Performance has regressed.
Found 12 outliers among 100 measurements
(12.00%)
...
```

4 比較結果

4.1 計算コスト

計算コストに対する理論値の比較を表 2, 3 に示す. ここでは, 乱数生成や乗算を行なう回数を比較する. ここで挙げる数値の計算は, アルゴリズムと実装をもとに計算を行なったものである.

表記について, l は全体のメッセージの要素数, d は開示するメッセージの要素数, d' は開示しないメッセージの要素数を表す.
R_{Z_p} は群 Z_p の中から一つ抽出するランダム生成の単位, R_{E_G} は楕円曲線の群 G 上の点の中から一つ抽出するランダム生成の単位, E_{Z_p} はべき乗計算, E_G は楕円曲線の群 G 同士のマルチスカラ倍算, P は楕円曲線の群 G_1 と G_2 のペアリング演算と仮定する.
例えば, $5E_{G_1}$ は楕円曲線の群 G_1 におけるマルチスカラ倍算を 5 度行うことを意味する.

表 2: 各署名アルゴリズムの計算コストの比較 1

	GenKey	GenSig	VerSig
CL 署名	$(l+8)R_{Z_p}$	$2R_{Z_p}+2E_{Z_p}$	$2E_{Z_p}$
BBS+署名	$1R_{G_1}+1E_{G_1}$	$2R_{Z_p}+(l)R_{G_1}+(l+2)E_{G_1}$	$(l+1)E_{G_1}+1E_{G_2}+2P$
PS 署名	$(l+1)R_{G_1}+(l+1)E_{G_2}$	$1R_{G_1}+1E_{G_1}$	$(l)E_{G_2}+2P$

表 3: 各署名アルゴリズムの計算コストの比較 2

	GenProof	VerProof
CL 署名	$2R_{Z_p}+(d'+16)E_{Z_p}$	$(d+16)E_{Z_p}$
BBS+署名	$2R_{Z_p}+(d'+8)E_{G_1}+1E_{G_2}$	$(l+3)E_{G_1}+2P$
PS 署名	$(d'+3)R_{Z_p}+3E_{G_1}+(2d'+2)E_{G_2}$	$(l+1)E_{G_2}+2P$

表 2 と表 3 の結果を以下にまとめる.

1. 表 2 の GenKey Cost の項目より, BBS+署名の鍵の計算コストのオーダーはメッセージ要素数に関わらず一定となり, CL 署名と PS 署名の鍵の計算コストのオーダーはメッセージ要素数に比例する.
2. 表 2 の GenSig Cost の項目より, PS 署名と CL 署名の証明生成の計算コストのオーダーはメッセー

[16] https://github.com/arkworks-rs/algebra
[17] https://doc.rust-lang.org/cargo/index.html
[18] https://doc.rust-lang.org/cargo/commands/cargo-bench.html
[19] https://github.com/0xvon/anoncreds-clsignatures-rs/tree/0xvon/benchmark-for-multiple-message-size

ジ要素数に関わらず一定となり, BBS+署名の証明生成の計算コストのオーダーはメッセージ要素数に比例する.

3. 表 2 の VerSig Cost の項目より, CL 署名の署名検証の計算コストのオーダーはメッセージ要素数に関わらず一定となり, BBS+署名と PS 署名の署名検証の計算コストのオーダーはメッセージ要素数に比例する.

4. 表 3 の GenProof Cost の項目より, どの署名方式も証明生成の計算コストのオーダーは非開示メッセージ要素数 d' に比例する.

5. 表 3 の VerProof Cost の項目より, CL 署名の証明検証の計算コストのオーダーは開示するメッセージ要素数 d に比例し, BBS+署名と PS 署名の証明検証の計算コストのオーダーは全体のメッセージ要素数 l に比例する.

4.2 データサイズ

データサイズの理論値の比較を以下の表 4 に示す.

Z_n は $n = pq$ のビットサイズ, G_1 は楕円曲線 G_1 上の点, G_2 は楕円曲線 G_2 の点, F は体要素と仮定する.

表 4: 各署名アルゴリズムのデータサイズの比較

	SecKey Size	PubKey Size	Sig Size	Proof Size
CL 署名	$2Z_n$	$(l+6)Z_n$	$3Z_n$	$(d'+6)Z_n$
BBS+署名	$1F$	$1F+1G_2$	$1G_1+2F$	$5G_1+(d'+4)F$
PS 署名	$(l+1)F$	$(l+1)G_2$	$2G_1$	$2G_1+2G_2+(d'+1)F$

上記の理論値を実際の数値に置き換えたものを表 5 に示す. CL 署名はセキュリティパラメータを 1024bit と仮定 [1] し, Z_n は 256Bytes とする. BBS+署名と PS 署名で用いる楕円曲線は BLS123-81 であることから, セキュリティパラメータを 384bit と仮定[20]し, G_1 は 48Bytes, G_2 は 96Bytes, F は 32Bytes とする.

表 5: 各署名アルゴリズムのデータサイズの理論値の比較

	SecKey Size	PubKey Size	Sig Size	Proof Size
CL 署名	$512B$	$(256l+1536)B$	$768B$	$(256d')B$
BBS+署名	$32B$	$128B$	$112B$	$(32d'+368)B$
PS 署名	$(32l+32)B$	$(96l+96)G_2$	$96B$	$(32d'+310)B$

表 4 と表 5 の結果を以下にまとめる.

1. 表 4 の SecKey Size, PubKey Size の項目より, BBS+署名と CL 署名の鍵サイズはメッセージ要素数 l に関わらず一定となり, PS 署名の鍵サイズはメッセージ要素数 l に比例する.

2. 表 4 の Sig Size の項目より, CL 署名, BBS+署名, PS 署名のいずれも署名サイズはメッセージ要素数 l に関わらず一定となる.

3. 表 5 の Sig Size の項目より, PS 署名の署名サイズは他の書名方式に比べて一番小さく, BBS+署名の署名サイズは PS 署名に比べて 16B ほど大きく, CL 署名の署名サイズは BBS+署名, PS 署名に比べて 5 倍以上になる.

4. 表 4 の Proof Size の項目より, CL 署名, BBS+署名, PS 署名のいずれも証明サイズは非開示メッセージ要素数 d' に比例する.

4.3 ベンチマーク

ベンチマークの実行ログより, 実行時間をプロットしたグラフを以下の 4 つの図 1, 2, 3, 4 に示す. GenSig Time, VerProof Time は BBS+署名の G_1 署名と G_2 署名, PS 署名, CL 署名の 4 種類を比較している. GenProof Time, VerProof Time は BBS+署名と PS 署名はそれぞれ $d = 1$(開示するメッセージが 1 つのみ) と $d' = 1$(開示しないメッセージが 1 つのみ) のときの 2 パターン, CL 署名は $d' = 1$ の 1 パターンを比較している. それぞれ, 全体のメッセージの要素数 l を変化させながら実行にかかる時間 (ms) の推移を計測した. 実行時間には, 100 回の試行の平均実行時間を記録した.

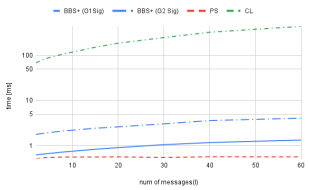

図 1: GenSig Time

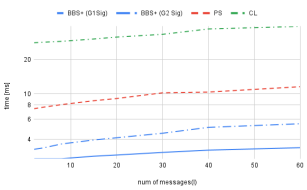

図 2: VerSig Time

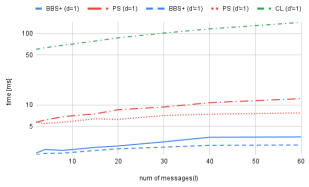

図 3: GenProof Time

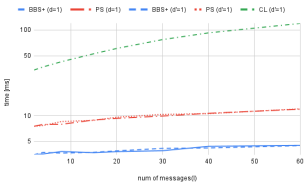

図 4: VerProof Time

図 1, 2, 3, 4 の結果を以下にまとめる.

1. 図 1, 2 より, BBS+署名の G_1 署名と G_2 署名について比較すると, BBS+署名の G_1 署名は G_2 署名より署名の生成と検証のどちらにおいても高速となる. メッセージ要素数 $l = 30$ のとき G_1 署名の方が署名生成が 2ms 程度高速となり, 署名検証が 1.5ms 高速となる.

[20] https://github.com/mattrglobal/bbs-signatures

2. 図 1 より, PS 署名は署名の生成速度が他の書名方式と比べて最も高速で, メッセージ要素数 $l = 2$ のときに 0.51ms, $l = 60$ のときに 0.56ms の実行時間とななる. 対して BBS+署名は $l = 2$ のときに 0.64ms, $l = 60$ のときに 1.33ms となる. CL 署名は最も低速となり, $l = 2$ のときに 68ms, $l = 60$ のときに 446ms となる.

3. 図 2, 3, 4 より, BBS+署名は署名の検証, 証明の生成, 証明の検証の速度が他の書名方式と比べて最も高速となる. $l = 30$ において PS 署名と比較すると, 署名の検証は 7.1ms だけ高速となり, 証明の生成は 6.3ms だけ高速となり, 証明の検証は 6.1ms だけ高速となる.

4. 図 2 より, BBS+署名 (G_2 署名) は PS 署名より署名検証が高速となる. メッセージ要素数 $l = 30$ において 5.7ms だけ高速となる.

5. 図 3 より, BBS+署名と PS 署名のどちらも開示メッセージ要素数 d が増えるほど証明の生成は高速となる.

6. 図 4 より,BBS+署名と PS 署名のどちらも開示メッセージ要素数 d が増えても証明の検証速度は変わらない.

7. 図 3, 4 より, BBS+署名と PS 署名のどちらも全体メッセージ要素数 l が大きくなるにつれて証明の生成と検証の速度は低速となる.

8. 図 1, 2, 3, 4 より, CL 署名は他の書名方式に比べて, 署名の生成, 署名の検証, 証明の生成, 証明の検証のすべての速度において低速となる.

5 考察

比較結果をもとに考察を行なう.

5.1 BBS+署名と PS 署名の特徴

結果より, BBS+署名と PS 署名を比較したときの, それぞれの特徴をまとめる. BBS+署名の特徴は以下の通りである.

1. 鍵のサイズが固定で小さい.
2. 署名の検証が高速.
3. 証明の生成が高速.
4. 証明の検証が高速.

一方で, PS 署名の特徴は以下の通りである.

1. 署名の生成が高速.
2. 署名のサイズが若干小さい.
3. 証明のサイズが若干小さい.

安全性については, BBS+署名はランダムオラクルモデルの下で安全であることが証明 [3] されており, PS 署名はジェネリックグループモデルの下で安全であることが証明 [6] されている.

その他にも, PS 署名ベースの閾値署名方式 [8] は BBS+署名ベースの閾値署名方式 [7] に比べ, 署名中に署名者同士の数回のやりとりを必要とせず, 効率的になると言われている [9].

5.2 ストレージ制約下の運用

日本のマイナンバーカード[21] のように, ストレージ制約のある物理カードやハードウェアデバイスに鍵や署名を保存する場合, BBS+署名の方が有利となる. これは, 結果の表 4, 5 より, BBS+署名は鍵サイズがメッセージ要素数に依存せず一定となり, 秘密鍵サイズが 32B, 公開鍵サイズが 96B, 署名サイズが 112B と小さいためである. 例えば, 日本のマイナンバーカードでは, 電子証明書や秘密鍵のデータが物理カードの IC チップに格納[22] されているが, このような IC チップのストレージ容量は非常に小さいため, できるだけ小さなサイズのデータを格納できる方が好ましい. 一方で, このような場合では, PS 署名や CL 署名は, ストレージ容量を超えないように, 署名者がメッセージ要素数 l の上限を定める必要がある.

5.3 実用的なメッセージ要素数 l の上限の検討

クレデンシャルの証明と検証にかかる合計の時間を 1s 以内に抑えたい場合, BBS+署名はメッセージ要素数を 50000 個以下, PS 署名はメッセージ要素数を 10000 個以下, 合計の時間を 100ms 以内に抑えたい場合, BBS+署名はメッセージ要素数を 5000 個以下, PS 署名はメッセージ要素数を 500 個以下にすることが望ましい. BBS+署名における証明の生成と検証にかかる合計実行時間は, $l = 50000$ のときに 696ms, $l = 5000$ のときに 92ms となる. PS 署名における証明の生成と検証にかかる合計実行時間は, $l = 10000$ のときに 804ms, $l = 500$ のときに 75ms となる. インターネット通信や, デバイス環境の変化による遅延時間を考慮すると, この程度のメッセージ要素数が現実的な水準となると考えられる.

ただし, 1000 個のメッセージ要素に対してクレデンシャルを発行する場合, 500 個 2 つのクレデンシャルに分けてクレデンシャルを 2 つ発行するよりは, 1 つのクレデンシャルに 1000 個のメッセージを含める方が効率

[21] https://www.kojinbango-card.go.jp/mynumber/
[22] https://www.soumu.go.jp/kojinbango_card/03.html

がよい. これは, CL, BBS+, PS のいずれの署名方式においても, メッセージ要素数 $l \leq 10000$ のとき, 証明の生成と検証にかかる時間はいずれもメッセージ要素数 $l/2$ のときにかかる時間より 2 倍未満となるためである.

5.4 本人確認用クレデンシャルの別途発行

政府が発行した身分証明書を用いて金融サービスで本人確認を行なう場合などでは, 本人確認のために必要な情報があらかじめ分かっていることが多い. そういった場合, 発行者は本人確認用のクレデンシャルを別途発行しておくことが望ましい. これは, 結果の図 3 より, BBS+ 署名, PS 署名ともに, 開示するメッセージの要素数 d が少ないほど証明の生成が低速になるためである. これは, 例えば, 本人確認に必要な属性が (年齢) の 1 つのみであるのに対し, クレデンシャルに (氏名, 年齢, 住所, 配偶者, 性別, ...) と数十個も属性が含まれていた場合, 属性が 2〜3 個のクレデンシャルに比べ, 証明の生成は非効率になる.

6 まとめ

この研究では, CL 署名, BBS+署名, PS 署名の 3 つのアルゴリズムの計算コスト・データサイズ・ベンチマークによる比較を行った. 比較より, 証明の生成・検証, 鍵サイズ, ベンチマークにおいて BBS+署名が最も有利となることが分かった. 更に, 署名に含めるメッセージ要素数 l の実用的な上限は 500〜10000 個であることや, 本人確認用クレデンシャルを別途発行することで効率を高めることを考察した.

参考文献

[1] Jan Camenisch, and Anna Lysyanskaya, ”A signature scheme with efficient protocols ”in SCN 02, set. LNCS, Stelvio Cimato, Clemente Galdi, and Giuseppe Persiano, Eds., vol. 2576, Springer, Heidelberg, Sep. 2003, pp. 268–289.

[2] Isamu Teranishi, and Kazue Sako, Teranishi, I., Sako, K., ”k-Times Anonymous Authentication with a Constant Proving Cost”in: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (eds) Public Key Cryptography - PKC 2006. PKC 2006. Lecture Notes in Computer Science, vol 3958. Springer, Berlin, Heidelberg., 2006, pp 525–542.

[3] Man Ho Au, Willy Susilo, and Yi Mu, “Constant-Size Dynamic k-TAA ”in SCN 06, ser. LNCS, R. D. Prisco, and M. Yung, Eds., vol. 4116. Springer, Heidelberg, Sep. 2006, pp. 111-125.

[4] Camenisch, J., Drijvers, M., Lehmann, A., ”Anonymous Attestation Using the Strong Diffie Hellman Assumption Revisited ”in: Franz, M., Papadimitratos, P. (eds) Trust and Trustworthy Computing. Trust 2016. Lecture Notes in Computer Science(), vol 9824. Springer, Cham, 2016, pp. 1–20.

[5] Dan Boneh, Xavier Boyen, and Hovav Shacham, “Short group signatures”in Advances in Cryptology - CRYPTO, 2004, pp. 44-55.

[6] David Pointcheval and Olivier Sanders, ”Short Randomizable Signatures ”in Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016, vol. 9610. Springer, Heidelberg, 2016, pp. 111–126.

[7] Jack Doerner, Yashvanth Kondi, Eysa Lee, abhi shelat, and LaKyah Tyner, ”Threshold BBS+ Signatures for Distributed Anonymous Credential Issuance ”in Cryptology ePrint Archive, Paper 2023/602, 2023.

[8] Sonnino, A., Al-Bassam, M., Bano, S., Meiklejohn, S., Danezis, G., ”Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers ”in 26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019, The Internet Society, 2019.

[9] Alfredo Rial and Ania M. Piotrowska, ”Security Analysis of Coconut, an Attribute-Based Credential Scheme with Threshold Issuance”in Cryptology ePrint Archive, Paper 2022/011, 2022.