SCIS 2018 2018 Symposium on
Cryptography and Information Security
Niigata, Japan, Jan. 23 - 26, 2018
The Institute of Electronics,
Information and Communication Engineers

ペアリングを用いた暗号方式の擬似コードレベルでのパフォーマンス評価
Pseudo-Code Performance Estimation for Pairing-Based Cryptographic Schemes

阿部 正幸 *† 星野 文学 * ‡ 大久保 美也子 §
Masayuki Abe Fumitaka Hoshino Miyako Ohkubo

あらまし ペアリングを利用する暗号方式の実装では主に BN 曲線が利用されてきたが、Kim らによる有限体上の離散対数問題に対する攻撃によって、ソース群のセキュリティパラメータを増大させるか、より埋め込み次数の大きな曲線を用いることでターゲット群のセキュリティパラメータを増大させる必要が生じた。いずれのアプローチが適切かは、対象となる暗号方式における演算の種類、回数、セキュリティレベルに依存する。したがって、暗号方式やセキュリティレベルに応じて、適切な曲線の種類やパラメータが異なることが考えられる。一方で、高速なペアリングライブラリの実装は特定の曲線やパラメータ限られることが多く、個別の暗号方式の効率を様々な曲線やパラメータに対して実装評価することは多大な労力・コストを要する。

本稿では，ペアリング群上の暗号方式の様々な曲線およびパラメータ設定におけるパフォーマンスを簡易的に評価するフレームワーク 'Opcount' を提案する。擬似コードによる暗号方式の記述と、評価対象の曲線における基本演算のパフォーマンスを記録した実装情報データベースから、その暗号方式のパフォーマンスを見積もることで、適切な曲線やパラメータの選択を可能とした。

キーワード ペアリング、パフォーマンス評価、ベンチマーク、タイトセキュリティ

1 はじめに

1.1 背景

128 ビットセキュリティレベルにおけるペアリングパラメータは BN 曲線を構成するパラメータが定着しつつあったが、2016 年 8 月以降 Kim 等による数体篩法の改良 (exTNFS) [8] によって様々な曲線の安全性が再検討される状況となった。また、セキュリティ帰着の厳密な解釈にもとづいて、タイト帰着を持たない暗号方式に対してより上位のセキュリティレベルでの実装が求められるようになった。ペアリング群に関する群の表現や演算速度は、具体的なパラメータやプラットフォーム、実装されている高速化技法によって大きく異なるため、ある暗号方式のパフォーマンスを最適にする、あるいは許容範囲に収めるためにセキュリティレベルやパラメータを選択するには、実際にその方式を実装して比較するしかないのが現状である。暗号理論における論文では、パフォーマンスの概算は支配的な演算の回数で評価するか、ひとつの具体的なパラメータやプラットフォームでの実装結果を示すに留まることが多い。

例えば、同じ困難性仮定に基づく暗号方式ＡとＢがあり、Ａはコンパクトで演算回数が少ないが 60bit の帰着ロスを持ち、Ｂは冗長でより多くの演算を必要とするがタイト帰着が示されているとする。同程度の安全性を確保するために、Ａを 192-bit 安全性のパラメータ、Ｂを 128-bit 安全性のパラメータで実装した場合、どちらがより効率的かを調べたい。異なるパラメータ、セキュリティレベル、プラットフォームに対して所望の暗号方式を逐次実装して最大限チューニングされたパフォーマンスを比較することは、実装及び方式に関する高度なスキルと労力を要する。それぞれの暗号方式における最も支配的な演算（多くの場合ペアリング）のコストのみを比較して議論することもあるが、各方式を実現するアルゴリズムは様々であり、それらの中で支配的な演算は異なる可能性があるため、結局のところ様々な演算のコストを考慮した上で、十分な精度でアルゴリズムの比較検討が必

* NTT セキュアプラットフォーム研究所, 東京都武蔵野市緑町 3-9-11, NTT Secure Platform Laboratory, 3-9-11 Musashino, Tokyo
† 京都大学大学院情報学研究科 社会情報学専攻, 京都市左京区吉田本町, Graduate School of Informatics, Kyoto University, Depertment of Social Informatics
‡ 東京工業大学 情報理工学院 , 〒 152-8550 東京都目黒区大岡山 2-12-1, School of Computing, Tokyo Institute of Technology, 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550 Japan
§ 国立研究開発法人情報通信研究機構, 東京都小金井市貫井北町 4-2-1, National Institute of Communications and Information Technology, 4-2-1 Nukui-Kitamachi, Tokyo (JSPS 科研費 JP16K00027 の助成を受けたものです。)

要になる。例えば、Groth-Sahai 証明 [4] や Structure-Preserving 署名 [1] の検証ではペアリング積の等式を数個から数十個評価するが、この際に Batch Verification を導入して検証を高速化できるかどうかは、元の検証式に含まれるパラメータや変数、Batch 化のアルゴリズム、関連する全ての演算の回数と速度比に依存するため、容易ではない。暗号方式開発の過程で様々なアルゴリズムを比較する、あるいは新たなアルゴリズムの効果の有無を調べる場合には、暗号方式のパフォーマンスをそのアルゴリズムの記述に基づいて十分な精度で推定できることが望ましい。

1.2 本稿の内容

本稿では、ペアリングを用いた暗号方式のパフォーマンスを評価する容易なフレームワーク 'Opcount' を提案し、その有効性と有用性を検証する。この評価フレームワークは、ペアリング群上の暗号方式や応用プロトコルを実装するアルゴリズムの設計を支援すると期待できる。

本稿の評価フレームワーク 'Opcount' は、CSDL(C-like Scheme Description Language) と呼ぶ独自の擬似コードによる暗号方式のアルゴリズム記述、具体的なパラメータやプラットフォームにおけるペアリング群の基本的な演算に関するタイミングを記録したデータベース (Performance Database:PDB)、および出力形式を記述したスクリプト (CMD) を入力とし、指定されたパラメータの値や関数のタイミングを評価して出力するものである。Opcount は、CSDL 擬似コードに含まれる基本的な演算の個数をカウントしてパフォーマンスの評価式を作成する「演算カウントステップ」と、PDB から得られるタイミング値を評価式に代入して具体的な値を算出し、指定されたスクリプトに従って結果を出力する「評価ステップ」の 2 段階で構成されている。評価ステップには PARI/GP を用いた。図 1 に処理フローを示す。

方式の記述と基本的な演算のパフォーマンスを個別に扱っているため、これらの組み合わせによって様々な方式の様々なパラメータ設定におけるパフォーマンスを容易に比較検討することができる。また、CMD スクリプトによって CSDL に記述されている構造化された複数の関数や変数値から必要な評価対象を柔軟に選択できる。パフォーマンスデータベース PDB は実在するペアリングライブラリのベンチマークに基づくものはもちろん、新たな曲線や算法のタイミングを机上評価した理論値に基づくものでもあり得る。方式の記述は CSDL に、基本演算のアルゴリズムや曲線のパラメタに依存する記述は PDB にと明確に分離することで、パフォーマンスを必要なだけ詳細に推定できることが 'Opcount' フレームワークの利点の一つである。

提案する 'Opcount' フレームワークの有効性と有用性を検証するため、以下の評価実験を行う。

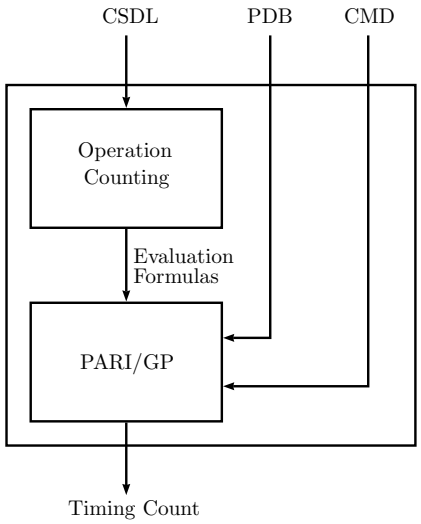

図 1: パフォーマンス評価手順の概念図

精度検証： 'Opcount' によるタイミングの評価値と実装によるタイミングの実測値を比較する。測定の対象としてペアリング演算を多数含む SP 署名方式 [2] を清村らのライブラリ [9] に基づいて 256-bit セキュリティのパラメータで実装し、鍵生成、署名生成、署名検証の各関数のタイミング実測値を本フレームワークによる評価値と比較した。

アルゴリズム改善効果の測定： 署名検証にペアリング積を多用する幾つかの SP 署名方式について、ソース群のマルチベーススカラー倍演算を多用して複数のペアリング積式を 1 つの式にまとめるバッチ検証化の効果を本フレームワークによって測定する。

セキュリティパラメータの異なる方式の効率比較： セキュリティパラメータ λ と署名オラクルへのアクセス回数 Q に関する帰着コストがそれぞれ Q^2, $Q\log Q$, λ である 3 つの Structure-Preserving 署名方式を、異なるセキュリティパラメータのもとで比較する。まず、全ての署名方式の速度を同じ 128-bit セキュリティパラメータで評価し、次に、帰着コストの大きな方式をより大きな 192-bit セキュリティパラメータで評価して比較する。

精度検証によって本フレームワークの有効性を確認し、次に、2 つの効果測定･効率比較においてアルゴリズムの変更による効果を容易に評価できることによる本フレームワークの柔軟性、有用性を示す。

2 擬似コード

暗号方式をデータ構造や特定の API に依存しない擬似コード形式で記述するために、C 言語に類似した言語

を開発した。ペアリング群 $(\mathbb{G}_0, \mathbb{G}_1, \mathbb{G}_t)$ の要素を表すデータ型として'group0', 'group1', 'target' を備える。セキュリティパラメータおよび群の位数はグローバル値で表現され、評価ステップで具体的な値で評価される。（したがって、記述できる方式は一つのペアリング群を扱うものに限られる。）他に、整数を表す 'integer' 型、任意の型の変数の並びを指定する 'list' 型を利用できる。整数はリテラルも変数も $\mathbb{Z}_p$ の要素とみなされる。各群の群演算は乗法群の記法に則り、乗算 '$*$'、べき乗演算 '^'、ペアリング '$e(\cdot,\cdot)$' を記述できる。

制御構文は 'if' や 'for' が記述できるが、記述を簡単にするためのごく単純な分岐や繰り返しに限定され、変数の値に依存した制御は記述できない。

簡単な例として Linear Encryption の復号関数の CSDL による表現を図 2 に示す。

```
group0 Linear_Dec_0(list sk, list ct)
{
    integer r1, r2;
    integer x1, x2;
    group0  u1, u2, u3, M;

    (x1, x2) = sk;
    (u1, u2, u3) = ct;
    M = u3 * u1^(-x1) * u2^(-x2);

    return M;
}
```

図 2: CSDL 擬似コードの例

この擬似コードを専用のプログラムで処理し、図 3 に示す評価式を得る。

```
call_Linear_Dec_0 = (1)*GROUP0_assign+(1)*
  GROUP0_batch(2*1,1*1)+(2)*INTEGER_uminus+(2)*
  LIST_LITERAL_assign+(1)*LIST_VARIABLE_assign ;
```

図 3: 評価式の例

右辺の各項が基本演算（あるいは動作）にかかるコスト（タイミング）を表している。各項の変数に評価ステップで具体的な値が代入され、関数全体のコスト `call_Linear_Dec_0` が求められる。`GROUP0_batch(a,b)` は `a` 個の底からなる multi-base exponentiation と `b` 回の乗算を合わせたコストを表す変数であり、実際にそれがどのようなアルゴリズムで他の基本演算から計算されるか、あるいは、どのようなタイミング値であるかは次節に述べるパフォーマンス DB に記載する。

3 パフォーマンス DB

PDB はペアリング群に関する基本的な演算のタイミングを記載したデータベースである。タイミングは実在するペアリングライブラリのベンチマークであることが想定されているが、理論的に求めた評価値とすることも可能である。ペアリング群の位数や埋め込み次数の具体的な値や、各群（たとえば $\mathbb{G}_0$） における乗算コスト 'GROUP0_mul'、べき乗演算コスト 'GROUP0_pow'、あるいは更に低位の演算のコストが具体的な数値あるいは評価式で与えられる。上位の関数では Miller-Loop のコスト 'Miller-Loop'、最終ベキのコスト 'Final_Exponentiation' も記載される。前述のペアリング積のコスト 'pairing_batch' は、提供される API や計算方法を反映した評価式で記述される。図 4 に PDB 記述の例を示す。

```
\\ Common Parameters
CURVE  = "KSS-32";
Sep    = 256;  \\ security parameter
Order  = 738;  \\ Group order in bits
EmbDeg = 32;   \\ Embedding degree
ExtDeg = 8;    \\ Extension Degree

\\ Measured functions
GROUP0_pow            = 7.399618 * Mclk;
GROUP1_pow            = 41.328962 * Mclk;
TARGET_pow            = 49.085291 * Mclk;
Miller_Loop           = 26.057327 * Mclk;
Final_Exponentiation = 156.147854 * Mclk;
GROUP0_mul            = 0.017542 * Mclk;
GROUP0_square         = 0.009950 * Mclk;
GROUP1_mul            = 0.129000 * Mclk;
GROUP1_square         = 0.147782 * Mclk;
TARGET_mul            = 0.158917 * Mclk;
TARGET_square         = 0.116364 * Mclk;
TARGET_inv            = 0.314343 * Mclk;
GROUP0_batch(n,m) = GROUP0_pow * n + GROUP0_mul
  * (max(0,n-1) + m);
GROUP1_batch(n,m) = GROUP1_pow * n + GROUP1_mul
  * (max(0,n-1) + m);
pairing_batch(n)  = n * Miller_Loop +
  Final_Exponentiation + (n-1) * TARGET_mul;
```

図 4: PDB エントリーの例

今回はそれぞれ性質の異なる 3 種類の PDB を作成した。

Kiyo17 PDB: ACNS2017 で発表された清村らによるタイミングデータ [9] に基づく PDB。BLS-24, KSS-32, KSS-36, BLS-42, BLS-48 曲線について、それぞれ exTNFS を考慮した 256-bit セキュリティレベルのパラメータを選定して実装している。公開されていない低位の演算コストを実測して 'Opcount' の精度向上を図るため、[9] と同一のライブラリを別のプラットフォーム (Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz stepping 01, Linux 2.6 x86_64 (CentOS 6.8), gcc version 6.3.0 (GCC)) 上で新たに測定して得られたタイミングデータを PDB 化した。測定データを 表 1 に示す。曲線毎、演算毎の相対的なタイミングの傾向は [9] のベンチマークと同様だが、2 割程度速度が向上している。

BD17 PDB Barbulescu ら [3] による、exTNFS を考慮した 128, 192-bit セキュリティレベルにおけるペアリング関連演算のタイミングを 16 ビット × 16 ビット の乗算回数で見積もった PDB。 128-bit セキュリティレベルでは BN, BLS-12, KSS-16, KSS-18 の 4 つの曲線パ

表 1: Kiyo17 のタイミングデータ (Xeon)

		BLS-24	KSS-32	KSS-36	BLS-42	BLS-48
Pairing	ML	42.00	26.05	29.97	30.52	16.78
	FE	68.01	156.14	114.79	82.15	77.59
	Total	110.01	182.19	144.76	112.67	94.37
スカラー倍算 in $\mathbb{G}_0$		10.43	7.39	5.78	3.70	3.34
スカラー倍算 in $\mathbb{G}_1$		30.92	41.32	28.73	39.34	21.18
べき乗算 in $\mathbb{G}_t$		52.58	49.08	58.28	43.94	49.56

(in Mclock)

ラメータを選定し、192-bit レベルでは KSS-18, BLS-24 のパラメータを選定してタイミングを見積もっている。

Khandaker らは論文 [6] で上記の BN, BLS-12, KSS-16 について実装し、[3] とほぼ同様の比較結果となることを検証している。特に KSS-16 については新たな手法を導入して Miller-Loop を高速化しており、[3] と同様に、128-bit セキュリティレベルでは KSS-16 が優位であると結論している。

RV15 PDB: Rajeev Verma による BN 曲線の ARM v7 上でのベンチマーク [10] に基づく PDB。exTNFS を考慮していない旧来のパラメータ設定で 128, 192, 256-bit セキュリティレベルでそれぞれタイミングを計測している。今回の実験には使用していないが、現状実装されている BN 曲線上のアプリケーションについて、単純にセキュリティパラメータを増加してセキュリティレベル向上を図った場合のパフォーマンスへの影響を測ることに利用できる。

文献では様々な環境でのベンチマークが公開されているが、exTNFS を考慮したパラメータに基づくものは少ない。特に、192-bit セキュリティにおける実装は報告がなく、今後の進展が期待される。

4 精度検証

本フレームワークによるタイミングの評価値と実際に動作する実装のタイミングの実測値を比較する。測定の対象としてペアリング演算を多数含む SP 署名方式 [2] を清村らのライブラリ Kiyo17 に基づいて 256-bit セキュリティの曲線 5 種類に対して実装し、鍵生成、署名生成、署名検証の各関数のタイミング実測値を 'Opcount' による評価値と比較した結果を表 2 に示す。

実測値は 10 回のランダム試行の平均値であり、標準偏差は、鍵生成で 1.93 ∼ 2.96 Mclock、署名生成で 2.24 ∼ 3.81 Mclock、検証で 0.35 ∼ 5.28 Mclock であった。

一部の署名生成で誤差がマイナスとなっている（すなわち、実測値のほうが速い）が、これは、評価値が参照している PDB のベンチマーク測定時と、署名方式として実装測定した時のキャッシュの効果の違いによるものと考えられる。

5 アルゴリズム改善効果の測定

ペアリング積式の Batch Verification とは、

$$1 = \prod_i e(X_{1i}, Y_{1i}), \cdots, 1 = \prod_i e(X_{ni}, Y_{ni})$$

のような n 個のペアリング積の等式を検証する際に、1 個別に各式を検証する代わりに乱数 r_j を使って

$$1 = \prod_j \prod_i e(X_{ji}, Y_{ji})^{r_j}$$

のようにまとめた式を検証する高速化の方法である。まとめた式の計算において乱数 r_j を更にペアリング内部に取り込むことおよび、同じ変数を持つ複数のペアリング積 $e(X, Y_i)^{r_i}\, e(X, Y_j)^{r_j}$ を $e(X, Y_i^{r_i} Y_j^{r_j})$ のようにマージすることによって、さらに効率的に計算することができるが、$e(X_{ji}^{r_j}, Y_{ji})$ とするか $e(X_{ji}, Y_{ji}^{r_j})$ とするかの選択、あるいはどの共通変数に基づくかによって、マージの結果は 1 通りになるとは限らない。とにかくペアリング演算回数を最小にするようマージする (本稿では Full-batch 法と呼ぶ)、乱数ベキを演算コストが比較的小さい $\mathbb{G}_0$ 側に移して $\mathbb{G}_1$ 側で可能な限り多くの共通用を持つ形にマージする ($\mathbb{G}_1$-batch 法とよぶ) などが考えられる。いずれが有効かは、元の検証式に含まれる変数やパラメータ、計算アルゴリズム、さらに関連する全ての演算の速度比によって異なると考えられる。

そこで、多数のペアリング積式を署名検証に用いる Structure-Preserving 署名方式（AHNOP 方式 [2]）に対して、Batch 検証なし、$\mathbb{G}_1$-batch 法、Full-batch 法の 3 つの署名検証アルゴリズムの様々な曲線パラメータにおけるパフォーマンスを 'Opcount' で評価した。$\mathbb{G}_0$ の群要素 10 個からなるメッセージに対して、それぞれのアルゴリズムにおける主要な演算とその回数は、Batch 検証なしでは 69 回のペアリング演算、$\mathbb{G}_1$-batch 法では 56 回の $\mathbb{G}_0$ 群スカラー倍演算と 34 回のペアリング演算、

表 2: AHNOP 署名方式の実測値と評価値

ペアリング曲線	鍵生成	署名生成	検証
BLS-24	944 / 941 (0.2%)	903 / 901 (0.1%)	3934 / 3927 (0.1%)
KSS-32	1098 / 1094 (0.3%)	1005 / 1008 (-0.3%)	4152 / 4149 (0.07 %)
KSS-36	782 / 777 (0.7%)	723 / 720 (0.4%)	3827 / 3802 (0.6%)
BLS-42	1005 / 958 (4.6%)	890 / 860 (3.3%)	3356 / 3349 (0.1 %)
BLS-48	553 / 550 (0.6%)	503 / 503 (-0.05%)	2337 / 2334 (0.1%)

誤差率 (%) = (実測値 - 評価値)/ 実測値. (in Mclock)

Full-batch 法では 38 回の $\mathbb{G}_0$ 群スカラー倍演算、18 回の $\mathbb{G}_1$ 群スカラー倍演算、および 26 回のペアリング演算である。[1]Batch 化に用いる乱数 r_j のサイズは 128 ビットで統一した。(曲線によっては、Group Order が 2^{128} を大きく超えるため、r_j を $\mathbb{Z}_p$ から一様ランダムに選ぶと効率が著しく悪くなる。乱数 r_j のサイズは安全性の帰着では攻撃者の能力に依存しない statistical なパラメータなので、128 ビットの一様ランダムな変数としても安全性仮定への影響はない。)

BV17 PDB に登録されている 128-bit セキュリティの曲線 KSS-18, KSS-16, BLS-12, BN によるパフォーマンスの評価結果を 表 3 に示す。いずれの曲線の場合でも、Full-batch 法すなわち可能な限りペアリングの数を減らすように検証式をマージすることが最も効率が良い検証となった。

表 3: AHNOP 署名における Batch 検証アルゴリズムの効果比較

	曲線			
Batch 方法	KSS-18	KSS-16	BLS-12	BN
なし	111.9	97.1	141.8	207.3
$\mathbb{G}_1$-batch	46.6	38.7	70.5	103.7
Full-batch	40.3	35.6	57.0	82.4

(in M clocks)

6 セキュリティパラメータの異なる方式の効率比較

SXDH 仮定に基づく Structure-Preserving 署名方式として Kiltz-Pang-Wee(KPW) 方式 [7]、Jutla-Roy(JR) 方式 [5]、Abe-Hofheinz-Nishimaki-Ohkubo-Pan(AHNOP) 方式 [2] が知られている。これらはいずれも異なる署名サイズと検証式を持ち、同一のセキュリティパラメータで比較した場合の効率は、直感的には、最小の署名サイズを持つ JR が最も効率が良く、最も大きな署名サイズの AHNOP の効率が最も悪いと予想できる。まず、その直感を確認するため、全ての方式の鍵生成、署名生成、署名検証のタイミングを BD17 PDB を用いてセキュリティパラメータ 128-bit での KSS-16 曲線上で評価した。$\mathbb{G}_0$ の要素 10 個に対するパフォーマンスを評価した結果を表 4 の上段に示す。

一方、安全性の帰着コストは KPW が $\mathcal{O}(Q^2)$ (Q:署名生成クエリ回数)、JR が $\mathcal{O}(Q \log Q)$ であるのに対し、AHNOP は "almost tight" 帰着であり $\mathcal{O}(\lambda)$ (λ:セキュリティパラメータ) の帰着コストである。$Q = 2^{30}$ のとき、AHNOP をセキュリティパラメータ $\lambda = 128$ で実装したとすると、帰着ロスが 30 ビット以上ある KPW や JR が AHNOP と同等以上のセキュリティマージンを確保するためには これらの方式を 1 つ上の安全性レベルである 192-bit セキュリティのパラメータで実装することになる。(192-bit セキュリティは帰着ロスを補償するには過大な設定であるが、128-bit と 192-bit の間に入る任意のセキュリティパラメータにおける適切な曲線の選定や実装は、現状見当たらない。) KPW および JR の 192-bit セキュリティパラメータにおけるパフォーマンスを 'Opcount' で評価した結果を表 4 の下段に示す。署名検証は Fullj-batch 法の検証式による。PDB は BD17 で、曲線は BLS-24 を用いた。メッセージは $\mathbb{G}_0$ の群要素 10 個とした。

鍵生成、署名生成、署名検証の全てで 128-bit レベルの AHNOP が 192-bit レベルの JR、KPW より 2~3 倍高速になっていることがわかる。 帰着がタイトであることの現実的な有効性を示す結果となった。

7 まとめ・今後の課題

ペアリング群上の暗号方式の擬似コードに基づいて容易にパフォーマンスを評価できるフレームワーク 'Opcount' を提唱し、その有効性、有用性を確認した。CSDL の記述力の向上と、より多様なベンチマーク結果による PDB の拡充が今後の課題である。

[1] 但し、n 個のベースに対する multi-scalar 倍演算を n 個の scalar 倍算と同じコストとした場合。BV17 PDB では、multi-scalar 倍演算において squaring の回数を低減するアルゴリズムを仮定している。

表 4: 方式間のパフォーマンス比較

Curve/Security	方式	帰着コスト	署名サイズ	鍵生成	署名生成	署名検証
KSS-16/128	AHNOP	$\mathcal{O}(\lambda)$	(13,12)	18.1	14.8	35.6
	JR	$\mathcal{O}(Q \log Q)$	(5,1)	14.1	6.3	19.1
	KPW	$\mathcal{O}(Q^2)$	(6,1)	13.1	7.0	21.0
BLS-24/192	JR	$\mathcal{O}(Q \log Q)$	(5,1)	59.6	27.9	109.9
	KPW	$\mathcal{O}(Q^2)$	(6,1)	55.0	30.6	121.1

$(n, m) : n * |\mathbb{G}_0| + m * |\mathbb{G}_1|$
(in Mclock)

謝辞

PDB 作成および比較のための実装にあたってご協力頂きました NTT 西日本の清村氏、NTT セキュアプラットフォーム研究所の川原氏に感謝します。

参考文献

[1] M. Abe, G. Fuchsbauer, J. Groth, K. Haralambiev, and M. Ohkubo. Structure-preserving signatures and commitments to group elements. *J. Cryptology*, 29(2):363–421, 2016.

[2] M. Abe, D. Hofheinz, R. Nishimaki, M. Ohkubo, and J. Pan. Compact structure-preserving signatures with almost tight security. In J. Katz and H. Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 548–580. Springer, 2017.

[3] R. Barbulescu and S. Duquesne. Updating key size estimations for pairings. *IACR Cryptology ePrint Archive*, 2017:334, 2017.

[4] J. Groth and A. Sahai. Efficient noninteractive proof systems for bilinear groups. *SIAM J. Comput.*, 41(5):1193–1232, 2012.

[5] C. S. Jutla and A. Roy. Improved structure preserving signatures under standard bilinear assumptions. In S. Fehr, editor, *Public-Key Cryptography - PKC 2017 - 20th IACR International Conference on Practice and Theory in Public-Key Cryptography, Amsterdam, The Netherlands, March 28-31, 2017, Proceedings, Part II*, volume 10175 of *Lecture Notes in Computer Science*, pages 183–209. Springer, 2017.

[6] M. A. Khandaker, Y. Nanjo, L. Ghammam, S. Duquesne, Y. Nogami, and Y. Kodera. Efficient optimal ate pairing at 128-bit security level. In A. Patra and N. P. Smart, editors, *Progress in Cryptology - INDOCRYPT 2017 - 18th International Conference on Cryptology in India, Chennai, India, December 10-13, 2017, Proceedings*, volume 10698 of *Lecture Notes in Computer Science*, pages 186–205. Springer, 2017.

[7] E. Kiltz, J. Pan, and H. Wee. Structure-preserving signatures from standard assumptions, revisited. In R. Gennaro and M. Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 275–295. Springer, 2015.

[8] T. Kim and R. Barbulescu. Extended tower number field sieve: A new complexity for the medium prime case. In M. Robshaw and J. Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*, pages 543–571. Springer, 2016.

[9] Y. Kiyomura, A. Inoue, Y. Kawahara, M. Yasuda, T. Takagi, and T. Kobayashi. Secure and efficient pairing at 256-bit security level. In D. Gollmann, A. Miyaji, and H. Kikuchi, editors, *Applied Cryptography and Network Security - 15th International Conference, ACNS 2017, Kanazawa, Japan, July 10-12, 2017, Proceedings*, volume 10355 of *Lecture Notes in Computer Science*, pages 59–79. Springer, 2017.

[10] R. Verma. Efficient implementations of pairing-based cryptography on embedded systems. Master's thesis, Rochester Institute of Tehcnology, Rochester Institute of Technology, Rochester New York, Dec. 2015.