SCIS 2004 The 2004 Symposium on Cryptography and Information Security
Sendai, Japan, Jan.27-30, 2004
The Institute of Electronics, Information and Communication Engineers

Pairing 演算の高速化
Fast Computation of Pairing

小林 鉄太郎*† 斉藤 泰一 * 今井 秀樹†
Tetsutaro KOBAYASHI Taiichi SAITO Hideki IMAI

あらまし 概要: Weil pairing や Tate pairing は楕円曲線に関する写像であり、近年 ID ベース暗号や Short signature への応用が行われている。しかし、通常の楕円演算（スカラー倍）に比較して演算量が多いことがネックであった。本発表では、新しく高速に pairing を求めるアルゴリズムを提案し、従来法との効率の比較を行う。

Keywords: wail pairing, tate pairing, elliptic curve cryptosystem, fast computation

1 はじめに

Boneh らが 2000 年に提案した Weil pairing を用いた楕円曲線上の ID ベース暗号方式 [1] や Short signature 方式 [2] など、Weil pairing や Tate pairing を用いた方式が注目されている。

Weil pairing は楕円曲線の点のなす群の直積から有限体乗法群への双線形写像であり、多くの場合実装には Miller のアルゴリズム [3] を用いる。このアルゴリズムを用いて pairing 演算を実装すると、通常の楕円スカラー倍演算などに比較して演算量が多いため、速度が遅くなることが問題となる。このため、Miller のアルゴリズムの高速実装法の研究が盛んに行われている。[5, 4]

本発表では、Miller のアルゴリズムを改良し、より高速に pairing を求めるアルゴリズムを 2 つ提案する。

2 Weil pairing と Tate pairing

E を F_q 上定義される楕円曲線とする。$P \in E(F_q)$ であり、その位数 $m(= \#\{P\})$ は素数とする。$m \not| (q-1)$ かつ $\gcd(q,m) = 1$ とし、k を $m|(q^k-1)$ をみたす最小の整数とする。

Weil pairing e_m は、2 つの m-等分点 $E[m] \times E[m]$ から有限体乗法群 F_{q^k} への双線形写像であり、以下のように定義される [6], [7]。

まず $P, Q \in E[m]$ とする。D_P と D_Q を $D_P \sim (P) - (\mathcal{O})$, $D_Q \sim (Q) - (\mathcal{O})$ を満たし、disjoint な support を持つとする。

$f_P,\ f_Q \in \overline{\mathbf{F}}_q(E)$ を $\mathbf{div}(f_P) = mD_P$, $\mathbf{div}(f_Q) = mD_Q$ とすると、Weil pairing は e_m は、$S, S' \in E(F_{q^k})$ を用いて

$$\begin{aligned} e_m(P,Q) &:= f_P(D_Q)/f_Q(D_P) \\ &= \frac{f_P(Q+S)}{f_P(S)}\frac{f_Q(S')}{f_Q(P+S')} \end{aligned}$$

と定義できる。

Weil pairing は、双線形の性質を満たしているため、この性質を用いて ID ベース暗号 [1] や Short Signature [2] への応用を行うことが出来る。

双線形性:

$e_m(P_1+P_2, Q) = e_m(P_1,Q)e_m(P_2,Q)$

$e_m(P, Q_1+Q_2) = e_m(P,Q_1)e_m(P,Q_2)$;

(ただし $P, P_1, P_2, Q, Q_1, Q_2 \in E[m]$)

Tate pairing ϕ_m は $S \in E(F_{q^k})$ を用いて

$$\begin{aligned} \phi_m(P,Q) &:= (f_P(D_Q))^{(q^k-1)/m} \\ &= \left(\frac{f_P(Q+S)}{f_P(S)}\right)^{(q^k-1)/m} \end{aligned}$$

によって定義され、双線形な写像となるため Weil pairing と同様にアプリケーションに適用できる。Weil pairing より高速に演算できる利点がある。

* NTT 情報流通プラットフォーム研究所, 〒 239-0847 神奈川県横須賀市光の丘 1-1
NTT Laboratories, 1-1 Hikarinooka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan
† 東京大学生産技術研究所 大学院情報学環, 〒 153-8505 目黒区駒場 4-6-1
Institute of Industrial Science Interfaculty Initiative for Information Studies, University of Tokyo, 4-6-1 Komaba, Meguro-ku, Tokyo, 153-8505 Japan.

3 Pairing の計算法

Tate pairing のうち $f_P(Q+S)/f_P(S)$ の演算を行う代表的アルゴリズムが Miller のアルゴリズムである。Miller のアルゴリズムの出力値を F_{q^k} 上で $(q^k-1)/m$ 乗することで、容易に Tate pairing を求めることが出来る。したがって、Tate pairing の演算量はほぼ Miller のアルゴリズムの演算量と等しくなる。

また、これと同じアルゴリズムを 2 回用いれば $f_P(Q+S)/f_P(S)$ と $f_Q(P+S')/f_Q(S')$ をそれぞれ求めることができるので、Miller のアルゴリズム 2 回と F_{q^k} 上の除算 1 回を行うことで Weil pairing を求めることができる。この場合も、Miller のアルゴリズムが演算量のほとんどを占める。

したがって、以下では Miller のアルゴリズムの演算量に関する議論を行う。Miller のアルゴリズムは 3.1 章で示すとおり。入力値 m は P,Q の位数 (素数) であり、$m[i]$ は m の i 番目のビットをあらわす。

3.1 Miller のアルゴリズム

[Miller's Algorithm]

Input: $m, P \in E(F_q)[m], Q \in E(F_{q^k})[m], S \in E(F_{q^k})$
Output: $f_P(Q+S)/f_P(S) \in F_{q^k}$

Step 1: $T \leftarrow P, f \leftarrow 1$

Step 2: For $i = n-1$ downto 0

Step 3: $l_1 = T$ と $-2T$ を結ぶ直線、$l_2 = 2T$ と $\mathcal{O}$ を結ぶ直線を求める。

Step 4: $T \leftarrow 2T$

Step 5: $f \leftarrow f^2 l_1(Q+S)l_2(S)/l_1(S)/l_2(Q+S)$

Step 6: If $m[i] == 1$ then

Step 6-1: $l_1 = T$ と P を結ぶ直線、$l_2 = (T+P)$ と $\mathcal{O}$ を結ぶ直線を求める。

Step 6-2: $T \leftarrow T + P$

Step 6-3: $f \leftarrow f l_1(Q+S)l_2(S)/l_1(S)/l_2(Q+S)$

Step 7: f を出力

このアルゴリズムでは、$n = \log_2(m)$ 回の楕円 2 倍演算とそれに関係する直線 l_1, l_2 の計算、m のハミング重み $w_H(m)$ 回の楕円加算とそれに関係する直線 l_1, l_2 の計算をおこなう。

[5] では、直線に関する計算の効率化について研究している。本稿では、Miller のアルゴリズムを改良し、演算量を削減する方法について述べる。

4 改良 Miller アルゴリズム

オリジナルの Miller のアルゴリズムは、楕円 m 倍演算をバイナリ法を用いて行うアルゴリズムによく似ている。(m は点 P の位数)

本稿では、楕円スカラー倍演算を高速に行う方法としてよく用いられている、window 法や comb 法と似た考えかたを Miller のアルゴリズムに応用する方法について述べる。

4.1 提案法 1: window Miller アルゴリズム

[window Miller's Algorithm]

Input: $m, P \in E(F_q)[m], Q \in E(F_{q^k})[m], S \in E(F_{q^k})$
Output: $f_P(Q+S)/f_P(S) \in F_{q^k}$

(online-precomptation)

Step 1: $P_1 = P, f'_1 = 1$

Step 2: For $i = 2$ to $2^w - 1$

Step 3: $P_i = iP_i$

Step 4: $l_1 = -P_i$ と P を結ぶ直線、$l_2 = P_i$ と $\mathcal{O}$ を結ぶ直線を求める。

Step 5: $f'_i = f'_{i-1} l_1(Q+S)l_2(S)/l_1(\dot{S})/l_2(Q+S)$

(main computation)

Step 6: $T \leftarrow P, f \leftarrow 1$

Step 7: For $i = n-1$ downto 0 step w

Step 8: Step 8-1 から 8-3 を w 回繰り返す。

Step 8-1: $l_1 = T$ と $-2T$ を結ぶ直線、$l_2 = 2T$ と $\mathcal{O}$ を結ぶ直線を求める。

Step 8-2: $T \leftarrow 2T$

Step 8-3: $f \leftarrow f^2 l_1(Q+S)l_2(S)/l_1(S)/l_2(Q+S)$

Step 9: $m' \leftarrow \sum_{j=i-w+1}^{i} m[j]2^{j-i+w-1}$

Step 10: If $m' \neq 0$ then

Step 10-1: $l_1 = T$ と $P_{m'}$ を結ぶ直線、$l_2 = (T + P_{m'})$ と $\mathcal{O}$ を結ぶ直線を求める。

Step 10-2: $T \leftarrow T + P_{m'}$

Step 10-3: $f \leftarrow f f'_{m'} l_1(Q+S)l_2(S)/l_1(S)/l_2(Q+S)$

Step 11: f を出力

window Miller アルゴリズムは、オンライン事前演算を用いる方法である。

まず、$i = 1, \ldots, 2^w - 1$ に対して 2^w 個の楕円曲線上の点 $P_i = iP$ を求め、各 P_i において

$$i(P) - i(\mathcal{O}) = (P_i) - (\mathcal{O}) + div(f_i)$$

が成り立つような有理式 f_i に対し、

$$f_i' = f_i(Q+S)/f_i(S)$$

なる $f_i' \in F_{q^k}$ を求める。

この P_i, f_i' を用いることで、楕円加算および直線 l_1, l_2 に関係する演算を削減する。上記アルゴリズムの説明では、簡単にするために $w|n$ を仮定している。

Miller のアルゴリズムでは m のハミング重み $w_H(m)$ 回の楕円加算を行っていたが、window Miller アルゴリズムでは、n/w 回行うことになるため、適切な w を用いれば、楕円加算および直線 l_1, l_2 の計算を削減することができる。また、楕円 2 倍演算を必ず w 回ずつ連続して行うことになるが、[5] で連続する楕円 2 倍演算の高速化も提案されているため、この点でも効率をあげることができる。

4.2 提案法 2: comb Miller アルゴリズム

comb Miller アルゴリズムは、オフライン事前演算を用いる方法である。

あらかじめ、オフライン事前演算として、ある $O(\sqrt{m})$ 以上の整数を j として、楕円曲線上の点 $P' = jP$ および

$$j(P) - j(\mathcal{O}) = (jP) - (\mathcal{O}) + div(f_j)$$

が成り立つような有理式 f_j に対して、

$$f' = f_j(Q+S)/f_j(S)$$

なる $f' \in F_{q^k}$ を用意しておく。

その後、$m = m_0 + m_1 j$ なる m_0, m_1 を求め、Miller のアルゴリズムを m_0, m_1 に対して並列的に実行する。$m_0[i], m_1[i]$ はそれぞれ m_0, m_1 の i 番目のビットをあらわす。また、 m_0, m_1 の大きいほうのビット長を n とする。(ここでは簡単のために仮に $m_0 > 2m_1$ の場合を示す。)

楕円加算、楕円 2 倍演算および l_1, l_2 の直線に関係するする演算を約半分に削減することができる。

[comb Miller's Algorithm]

Input: $m, P \in E(F_q)[m], Q \in E(F_{q^k})[m], S \in E(F_{q^k}), P' = jP \in E(F_q), f' \in F_{q^k}$
Output: $f_P(Q+S)/f_P(S) \in F_{q^k}$
Step 1: $P'' = P + P'$
Step 2: $l_1 = P$ と P' を結ぶ直線、$l_2 = P''$ と $\mathcal{O}$ を結ぶ直線を求める。
Step 3: $f'' \leftarrow f' l_1(Q+S) l_2(S)/l_1(S)/l_2(Q+S)$
Step 4: $T \leftarrow P, f \leftarrow 1$
Step 5: For $i = n-1$ downto 0
Step 6: $l_1 = T$ と $-2T$ を結ぶ直線、$l_2 = 2T$ と $\mathcal{O}$ を結ぶ直線を求める。
Step 7: $T \leftarrow 2T$
Step 8: $f \leftarrow f^2 l_1(Q+S) l_2(S)/l_1(S)/l_2(Q+S)$
Step 9: If $m_0[i] == 1$ & $m_1[i] == 0$ then
Step 9-1: $l_1 = T$ と P を結ぶ直線、$l_2 = (T+P)$ と $\mathcal{O}$ を結ぶ直線を求める。
Step 9-2: $T \leftarrow T + P$
Step 9-3: $f \leftarrow f l_1(Q+S) l_2(S)/l_1(S)/l_2(Q+S)$
Step 10: If $m_0[i] == 0$ & $m_1[i] == 1$ then
Step 10-1: $l_1 = T$ と P' を結ぶ直線、$l_2 = (T+P')$ と $\mathcal{O}$ を結ぶ直線を求める。
Step 10-2: $T \leftarrow T + P'$
Step 10-3: $f \leftarrow f f' l_1(Q+S) l_2(S)/l_1(S)/l_2(Q+S)$
Step 11: If $m_0[i] == 1$ & $m_1[i] == 1$ then
Step 11-1: $l_1 = T$ と P'' を結ぶ直線、$l_2 = (T+P'')$ と $\mathcal{O}$ を結ぶ直線を求める。
Step 11-2: $T \leftarrow T + P''$
Step 11-3: $f \leftarrow f f'' l_1(Q+S) l_2(S)/l_1(S)/l_2(Q+S)$
Step 12: f を出力

4.3 効率

[5] によると、Miller のアルゴリズムにおける楕円加算部分楕円二倍部分に相当する演算量をそれぞれ TADD, TDBL とすると、以下のようになる。

$$\begin{aligned} \texttt{TADD} &= 4M_k + (6k+10)M + 4S. \\ \texttt{TDBL} &= 4M_k + 2S_k + (6k+7)M + 7S. \end{aligned}$$

ただし、F_q 上の乗算、自乗の演算量を M, S, F_{q^k} 上の乗算、自乗の演算量を M_k, S_k であらわしている。

k が比較的小さい値 (supersingular な楕円曲線の場合、$k = 2, 3, 6$) であることを考えると、ほぼ $M_k = k^2 M$ と考えられる。ここで、 $S = M$ とすると

$$\begin{aligned} \texttt{TADD} &= (4k^2 + 6k + 14)M \\ \texttt{TDBL} &= (6k^2 + 6k + 14)M \end{aligned}$$

また、m は n ビットのランダムな数とすると、Miller のアルゴリズムの演算量は

$$\texttt{MILLER} = n\texttt{TDBL} + (n/2)\texttt{TADD}$$

提案法 1 は、TADD を $n/2$ 回から n/w 回に削減する。ただし、オンライン事前計算部分で $2^w - 2$ 回追加で TADD を行う必要がある。演算量は以下のようになる。

$$\texttt{wMILLER} = n\texttt{TDBL} + (n/w + 2^w - 2)\texttt{TADD}$$

提案法 2 は、TDBL を n 回から $n/2$ 回に削減し、tt TADD を $n/2$ 回から $n/4$ 回に削減する。演算量は以下のようになる。

$$\texttt{cMILLER} = (n/2)\texttt{TDBL} + (n/4)\texttt{TADD}$$

例として、 $k = 6, n = 170, w = 4$ の場合を考えると、

$$\begin{aligned} \texttt{MILLER} &= 55590M, \\ \texttt{wMILLER} &= 48009M, \\ \texttt{cMILLER} &= 27795M \end{aligned}$$

となり、それぞれ 13.6 %, 50 % の演算量の削減効果がある。

提案法の中では comb Miller 法の効率が良いが、あらかじめ P' および f' を求めておく必要がある。P が固定の場合などは P' や f' をあらかじめ求めておくことは容易であるが、P 固定の場合の高速な演算法が [4] で提案されているため、comb Miller 法が有効なのは P が固定ではないが、P' や f' を高速に計算することができる場合に限定される。window Miller 法は、そのような制限条件は存在しない。

したがって、window Miller 法と、comb Miller 法は、場合によって使い分ける必要がある。

5 むすび

Tate Pairing や Weil Pairing の実装を行う際の一般的アルゴリズムである Miller のアルゴリズムの改良版として、window Miller 法と comb Miller 法の 2 つを提案した。位数がランダムな素数であると仮定すると、それぞれ、13.6 %, 50 % の高速化効果を見込める。

comb Miller 法は、高速化効果が高いが、事前に演算を行う必要があるため、適用範囲が限られる。

従来 提案された Miller のアルゴリズムの高速実装法 [5, 4] などと組み合わせることもできると考えられるが、具体的にどの程度の演算量になるかは今後の課題である。

参考文献

[1] D. Boneh and M. Franklin, “Identity based encryption from the Weil pairing,” *Proc. of Crypto 2001*, LNCS 2139, Springer-Verlag, pp.213–229, 2001.

[2] D. Boneh, B. Lynn and H. Shacham, “Short signatures from the Weil pairing,” *Proc. of Asiacrypt 2001*, LNCS 2248 of LNCS, Springer-Verlag, pp.514–532, 2001.

[3] S. D. Galbraith, K. Harrison, and D. Soldera, “Implementing the Tate pairing”, ANTS V, LNCS 2369, pp.324-337, Springer-Verlag, 2002.

[4] P.S.L.M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, “Efficient Algorithms for Pairing-Based Cryptosystems”, CRYPTO 2002, LNCS2442, pp354-368, Springer-Verlag, 2002.

[5] T. Izu and T. Takagi, “Efficient Computations of the Tate Pairing for the Large MOV Degrees” ICISC 2002, LNCS 2587, pp283-297, Springer-Verlag, 2003.

[6] A. Menezes, *Elliptic Curve Public Key Cryptosystem*, Kluwer Academic Publishers, 1993.

[7] J. H. Silverman, *The Arithmetic of Elliptic Curves*, GTM 106, Springer-Verlag, 1986.