SCIS 2002 The 2002 Symposium on Cryptography and Information Security
Shirahama,Japan, Jan.29-Feb.1,2002
The Institute of Electronics, Information and Communication Engineers

離散対数検証法

Checking $Q = kP$

小林 鉄太郎* 星野 文学 *
Tetsutaro Kobayashi Fumitaka Hoshino
kotetsu@isl.ntt.co.jp *fhoshino@isl.ntt.co.jp*

Abstract— We present a new method to check $Q = kP$ without computing kP. We also present two methods to check $Q = k_1P_1 + k_2P_2$ or to check $Q = \sum_i k_iP_i$. Since these methods don't depend on a specific group, we can use them for almost all discrete logarithm based system that uses checking function. These methods accelerate a verification about twice faster than traditional system, at most.

Keywords: discrete logarithm problem, fast verification, scalar multiplication, LLL

1 はじめに

DSA 署名や ElGamal 暗号などの離散対数問題に基づく方式を実装する場合、群の上でスカラー倍演算 (乗法群の場合はスカラー乗演算) を行なうことが主要な処理となる。この演算の高速化手法について様々な研究が行なわれてきた。たとえば、 Window 法や、事前演算による方法、楕円曲線暗号の場合はフロベニウス展開による方法などである。

ところで、署名検証の場合は、 $Q = kP$ であるかどうかを検証するために必ずしも P を k 倍する必要はない。 $Q = kP$ がなりたっていることだけを検証できれば十分である。この観点で検証の効率化を行なった方法にバッチ検証法 [1] などがある、これは大量の署名を一度に検証する場合などに有効な方法である。しかし、単一の署名検証には適用できないため、利用できる場合が限られていた。

本稿は、 $Q = kP$ であるかどうかを kP を求めずに検証する新しい方法を提案する。また、 $Q = k_1P_1 + k_2P_2$ のような複数基底の場合の検証に適用可能な方法も示す。この方法は、楕円曲線暗号や乗法群上の方式など、離散対数問題に基づく方式ならば一般に適用可能である。

この後の構成は次の通りである。第 2 章で提案法の基本的なアイデアを述べる。第 3 章で複数基底の場合一般に適用可能な方式を示す。第 4 章で考察を行ない、第 5 章でまとめを行なう。

* 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 Information Sharing Platform Laboratories, 1-1 Hikarinooka Yokosuka-shi Kanagawa 239-0847 Japan

2 アプローチ

2.1 高速楕円スカラー倍演算法

本稿の方法は、楕円スカラー倍演算を高速化する方法 [2] にヒントを得ている。 [2] の方法は、 $\alpha : P \to \alpha P$ なる非自明な写像が存在する特殊な楕円曲線 $E(\mathrm{GF}(p))$ において、楕円スカラー倍演算を高速化する方法である。

準同型写像法

Input: k, P, α $(P \in E(\mathrm{GF}(p)))$
Output: $R = kP$

Step 1: $k = k_1 + k_2\alpha$ を満たす整数 k_1, k_2 を $\mathrm{XGCD}(\frac{1}{2}, k, q, \alpha)$ により求める。ただし $|k_1|, |k_2| < \sqrt{q}$ であり、 q は P の位数とする。
Step 2: $Q = \alpha P$ を求める。
Step 3: $R = k_1P + k_2Q$ を求めて出力。

k_1, k_2 が q の半分のビット長であるため、 Step 3 は 直接 kP を求める場合の約半分の計算量ですむ。

Step 1 で用いている XGCD を以下に示す。これは拡張ユークリッドの互除法に基づく方法である。

XGCD 法

Input: $x, k, q, \alpha \qquad (0 < x < 1)$
Output: $l, m \quad (lk = m \bmod q)$

Step 1: 入力値 q, α を拡張ユークリッドの互除法に入力し、

$$\begin{aligned} s_i q + t_i \alpha &= r_i && \text{for } i \geq 0 \\ r_i &> r_{i+1} \geq 0 && \text{for } i \geq 0 \\ |s_i| &< |s_{i+1}| && \text{for } i \geq 1 \\ |t_i| &< |t_{i+1}| && \text{for } i \geq 0 \\ r_{i-1}|t_i| + r_i|t_{i-1}| &= q && \text{for } i \geq 1 \end{aligned}$$

を満たす s_i, t_i, r_i を得る。
Step 2: $r_n \geq q^x$ を満たす最大の整数 n を求める。
Step 3: $\boldsymbol{v}_1 = (r_{n+1}, -t_{n+1}), \boldsymbol{v}_2 = (r_n, -t_n)$ とする。
Step 4: $(k, 0) = \beta_1 \boldsymbol{v}_1 + \beta_2 \boldsymbol{v}_2$ となる有理数 β_1, β_2 を求める。
Step 5: 整数 b_1, b_2 を、それぞれ β_1, β_2 に最も近い整数とする。
Step 6: $(l, m) = (k, 0) - b_1 \boldsymbol{v}_1 - b_2 \boldsymbol{v}_2$ を出力する。

$$r_{i-1}|t_i| + r_i|t_{i-1}| = q \quad (\text{for } i = 0, 1, 2, \ldots)$$

が成り立つため、$r_n \geq q^x$, $r_n|t_{n+1}| + r_{n+1}|t_n| = q$ より $|t_{n+1}| < q^{1-x}$ であり、$r_{n+1} < q^x$ である。$\boldsymbol{v}_2$ もほぼ $(q^x, \pm q^{1-x})$ に近いサイズのベクトルとなる。これにより、XGCD は $l < q^x, |m| < q^{1-x}$ を満たす (l, m) を出力することができる。

2.2 高速 $Q = kP$ 検証法

本稿で提案する方法の基本的なアイデアは、2.1 章の準同型写像法を逆に用いて検証に使うことである。$Q = kP$ であるかどうかを調べたいときに上記アルゴリズムと同様の演算を行なうことにより、間接的に検証を行なう。

提案法 1 [単基底検証]

Input: $P, Q, k \qquad (P, Q \in G)$
Output: $Q = kP$ が成り立っている / いない

Step 1: $q = k_1 + k_2 k$ を満たす整数 k_1, k_2 を $\mathrm{XGCD}(\frac{1}{2}, q, q, k)$ により求める。ただし $|k_1|, |k_2| < \sqrt{q}$ であり、q は G の位数とする。
Step 2: $R = k_1 P + k_2 Q$ を求める。
Step 3: $R = O$ ならば「成り立っている」を、そうでなければ「成り立っていない」を出力。

提案法 1 による検証の原理は $\mathrm{GCD}(q, k_1) = 1$ かつ $0 < k_2 < q$ ならば、

$$Q = kP \quad \Leftrightarrow \quad O = k_1 P + k_2 Q$$

が成り立つことによる。

準同型写像法は特殊な楕円曲線でのみ有効な方法だったが、本稿の方法は特殊な楕円曲線である必要はなく、楕円曲線以外の一般の群 (乗法群や超楕円曲線など) でも用いることが出来る。

3 $Q = \sum_i k_i P_i$ 検証法

2 章で説明した基本的なアイデアは、$Q = kP$ を検証する場合 (これを単基底検証と呼ぶ) の方法であるが、$Q = k_1 P_1 + k_2 P_2$ のような検証を行なう場合 (これを複数基底検証と呼ぶ) にも拡張を行なうことができる。

提案法 2 [複数基底検証 (LLL 版)]

Input: $Q, P_1, P_2, \ldots, P_i, k_1, k_2, \ldots, k_i$
Output: $Q = k_1 P_1 + k_2 P_2 + \cdots + k_i P_i$ が成り立っている / いない

Step 1: $\boldsymbol{u}[0], \ldots, \boldsymbol{u}[i]$ を

$$\begin{aligned} \boldsymbol{u}[0] &= (1, k_1, k_2, \ldots, k_i) \\ \boldsymbol{u}[1] &= (0, q, 0, \ldots, 0) \\ &\vdots \qquad\qquad \vdots \\ \boldsymbol{u}[i] &= (0, 0, 0, \ldots, q) \end{aligned}$$

とする。i 個のベクトル $\boldsymbol{u}[0], \ldots, \boldsymbol{u}[i]$ の線形結合で表せるベクトルのうち、ノルムの小さいベクトルをひとつ求め、$\boldsymbol{v} = (v_0, v_1, \ldots, v_i)$ とする。[a]
Step 2: $R = -v_0 Q + v_1 P_1 + v_2 P_2 + \cdots + v_i P_i$ を求める。
Step 3: $R = O$ ならば「成り立っている」を、そうでなければ「成り立っていない」を出力。

[a] この方法には LLL 法、gauss 法 ($i = 2$ の場合) などがある [3]。

提案法 2 は、従来法の検証方法と比較して、$\frac{i+1}{i}$ 倍高速である。ただし、LLL アルゴリズムなどの最短ベクトル問題の近似解アルゴリズムの使用を前提としている。LLL アルゴリズムは、i に関して指数時間の演算時間がかかるため、i が大きくなると不利になる場合がありうる。

最短ベクトル問題の近似解アルゴリズムを前提としない方式として提案法 3 のような構成も可能である。

提案法 3 [複数基底検証 (ユークリッドの互除法版)]

Input: $Q, P_1, P_2, \ldots, P_i, k_1, k_2, \ldots, k_i$
Output: $Q = k_1P_1 + k_2P_2 + \cdots + k_iP_i$ が成り立っている / いない

Step 1:

$$\begin{aligned}(v_0, v_1, \ldots, v_i) &\leftarrow (1, k_1, \ldots, k_i)\\ j &\leftarrow 1\end{aligned}$$

Step 2: $(l, m) \leftarrow \mathrm{XGCD}(\frac{1}{2^j}, q, q, k)$
Step 3: すべての $z \in \{0, \ldots, i\}$ に対して

$$\left\{\begin{array}{rcl} v_z &\leftarrow& lv_z \\ v_z &\leftarrow& m \\ v_z &\leftarrow& lv_z \bmod q \end{array}\right.$$

とする。
Step 4: もし $j \leq n$ なら $j \leftarrow j+1$ として Step 2 へ
Step 5: $R = -v_0Q + v_1P_1 + v_2P_2 + \cdots + v_iP_i$ を求める。
Step 6: $R = O$ ならば「成り立っている」を、そうでなければ「成り立っていない」を出力。

提案法 3 は、従来法の検証方法と比較して、 $\frac{2^i}{2^i-1}$ 倍高速である。i が大きくなると高速化の効果が少なくなるが、Step 2 ～ Step 4 は全体として拡張ユークリッドの互除法 1 回と同等の演算量で行なうことができるため、オーバーヘッドは少ない。

4 考察

4.1 演算量

本稿で提案した検証法は、$Q = kP$ の検証 (単基底) の場合、従来法に比べて約 2 倍の効率で検証することができる。$Q = k_1P_1 + \cdots + k_iP_i$ の検証 (複数基底) の場合は、提案法 2 は $\frac{i+1}{i}$ 倍、提案法 3 は $\frac{2^i}{2^i-1}$ 倍高速である。

i が大きくなるほど高速化の効果が小さくなるため、$i \geq 3$ の場合には有効ではないが、通常の署名検証や暗号復号で必要となる検証処理は $i = 1, 2$ のものが多いため、実用上問題ない。

例：

- PSEC 暗号の復号時の検査: $i = 1$
- Shnorr 認証: $i = 2$
- ElGamal 署名の署名検証: $i = 2$

4.2 安全性

提案法は、$P_i, Q \in G$ であることを前提としている。P_i, Q が 群 G に入っているかどうかが不明な場合、バッチ検証と同様にセキュリティ上の考慮をする必要があり、$P_i, Q \in G$ であることも検証する必要がある [4]。

実用上用いられている多くの群の場合、この検査は十分短い時間で行うことができることが知られている [4]。

5 まとめ

離散対数に基づく方式において、暗号の復号や署名の検証などに用いられる離散対数検証を効率化する方法を提案した。

k 倍演算を行なわずに $Q = kP$ であることを検証することを可能とし、楕円曲線暗号、超楕円曲線暗号を含む任意の群に適用することができる。$Q = kP$ であることを検証する場合、従来の方法にくらべて約 2 倍の速度で検証を行なうことができる。

また、 $Q = k_1P_1 + k_2P_2$ のような複数基底の検証の場合に拡張を行なった。これにより、離散対数に基づくほとんど方式に提案法による検証を適用可能とした。

参考文献

[1] Bellare, M., Garay, J.A., Rabin, T.: Fast batch verification for modular exponentiation and digital signatures. In Nyberg, K., ed.: Advances in Cryptology — EUROCRYPT'98. Volume 1403 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1998) 236–250

[2] Gallant, R.P., Lambert, R.J., Vanstone, S.A.: Faster point multiplication on elliptic curves with efficient endomorphisms. In Kilian, J., ed.: Advances in Cryptology — CRYPTO2001. Volume 2139 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (2001) 190–200

[3] Cohen, H.: A Course in Computational Algebraic Number Theory. Springer (1996)

[4] Hoshino, F., Abe, M., Kobayashi, T.: Lenient/strict batch verification in several groups. In Davida, G.I., Frankel, Y., eds.: ISC2001. Volume 2200 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg (2001) 81–94