Integer Decomposition for Fast Scalar Multiplication on Elliptic Curves[⋆]

Dongryeol Kim and Seongan Lim

KISA (Korea Information Security Agency),
78, Garak-Dong, Songpa-Gu, Seoul 138-803, Korea
{drkim, seongan}@kisa.or.kr

Abstract. Since Miller and Koblitz applied elliptic curves to cryptographic system in 1985[3,6], a lot of researchers have been interested in this field and various speedup techniques for the scalar multiplication have been developed. Recently, Gallant *et al.* published a method that accelerates the scalar multiplication and is applicable to a larger class of curves[4]. In the process of their method, they assumed the existence of a special pair of two short linearly independent vectors. Once a pair of such vectors exists, their decomposition method improves the efficiency of the scalar multiplication roughly about 50%. In this paper, we state and prove a necessary condition for the existence of a pair of desired vectors and we also present an algorithm to find them.

Keywords. elliptic curve cryptosystem, scalar multiplication, integer decomposition, endomorphism

1 Introduction

Since the introduction of elliptic curve cryptosystem in 1985 by Miller and Koblitz, independently, cryptographic schemes using elliptic curves get lots of attention in the field of public key cryptography due to its low bandwidth and small space storage requirements. The scalar multiplication is the main operation in public key schemes using elliptic curves and it can be usually done by successive doubling and addition of points. The doubling and addition of points need a few inversions and multiplications over the underlying finite field.

Although the key size is small, the required complexity may still be relatively heavy and hence intensive researches have been done to improve its computational efficiency. Many different approaches to improve the computational efficiency for elliptic curve cryptography have been tried[1,2,3,5,7].

One of the approaches is to analyze the algebraic structure of elliptic curves and classify a class of special curves with better efficiency in the scalar multiplication. The most well-known and commonly used class is Koblitz curve. For

[⋆] This work was supported by R&D project 2002-s-073 of KISA

Koblitz curve, scalar multiplication does not need any point doubling by exploiting a feature of the Frobenious endomorphism[5,8,9,10]. When the underlying finite field is of characteristic 2, the Frobenious endomorphism can be efficiently computed since squaring is much faster than multiplication. The idea of using Frobenious endomorphism can be extended to elliptic curves with arbitrary characteristic, but the improvement of the efficiency can not be guaranteed.

Recently, a speedup idea of scalar multiplication kP using efficiently computable endomorphism of an elliptic curve E over a prime field F_q has been proposed by Gallant *et al.* for a point $P \in E$ of prime order n. In their paper, they introduced an idea using decomposition of $k = k_1 + \lambda k_2 \pmod{n}$ where λ is an integer that satisfies $\phi(P) = \lambda P$ and ϕ is an endomorphism on E. If the endomorphism is efficiently computable and one can guarantee that each component k_1, k_2 in the decomposition is short enough, say both of them are bounded by $\sqrt{n}$, then their method improves the computational efficiency up to 50 percent to the general method of computing scalar multiplication in elliptic curves over the prime field.

In order to find such a decomposition of $k = k_1 + \lambda k_2 \pmod{n}$ with $-\sqrt{n} < k_1, k_2 < \sqrt{n}$, Gallant *et al.* introduced a way to use two linearly independent short vectors v_1, v_2 in the kernel of the homomorphism $f : Z \times Z \to Z_n$ defined by $f(i, j) = i + j\lambda \pmod{n}$. For the simplicity of the notation, we call a set of such vectors v_1, v_2 by a GLV(R. Gallant, R. Lambert and S. Vanstone) generator that will be defined later. Gallant *et al.* also suggested a specific way to find a GLV generator, but the completeness of their method is claimed heuristically without any proof. In fact, the gaps unproven in their claim are

– One cannot guarantee the existence of a GLV generator.
– One cannot guarantee the success of finding a GLV generator even if there is a GLV generator.

In this paper, we propose

– A necessary condition for the existence of a GLV generator
– A method of finding a GLV generator when a GLV generator exists.

Our paper is organized as follows. In Section 2, we review briefly how to speedup the scalar multiplication using a GLV generator proposed by Gallant *et al.*'s in [4]. In Section 3, we present a necessary condition for the existence of a GLV generator and a new algorithm to find a GLV generator. Finally, we conclude and discuss some further works in Section 4.

2 Preliminary

Let F_q be a finite field of q elements and E be an elliptic curve defined over F_q with the point at infinity O. An endomorphism of E is a rational map $\phi : E \to E$ with $\phi(O) = O$.

Let $P \in E$ be a rational point of a large prime order n. Let ϕ be an efficiently computable endomorphism of E and ϕ acts on the subgroup $\langle P \rangle$ as a multiplication by λ which is an integer root of the characteristic polynomial of ϕ modulo n. Let k be an integer and be selected uniformly at random from the interval $[1, n-1]$. We consider a set of two linearly independent vectors in $Z \times Z$ that will be used to speed up the computation of kP and name it a GLV generator. For given λ, n, Gallant *et al.* introduced a homomorphism $f : Z \times Z \to Z_n$ defined by

$$f(i,j) = i + j\lambda \pmod{n}. \tag{1}$$

Here we define a GLV generator.

Definition A set $\{v_1, v_2\}$ of two linearly independent vectors v_1, v_2 in the kernel of the homomorphism f in (1) is called a GLV generator if each component of v_1, v_2 is bounded by $\sqrt{n}$.

2.1 How to Use a GLV Generator to Calculate *kP*

Now we briefly explain how Gallant *et al.* used a GLV generator to speed up the computation of kP. Suppose that $\{v_1, v_2\}$ is a GLV generator. Since v_1, v_2 are linearly independent over $Q \times Q$, they span $Q \times Q$. Therefore we have

$$(k, 0) = \beta_1 v_1 + \beta_2 v_2,$$

for some $\beta_1, \beta_2 \in Q$. Let b_1, b_2 be the nearest integers to β_1, β_2, respectively. Finally, set

$$\begin{aligned} x = (k_1, k_2) &= (k, 0) - (b_1 v_1 + b_2 v_2) \\ &= (k, 0) - (\beta_1 v_1 + \beta_2 v_2) + (\beta_1 v_1 + \beta_2 v_2) - (b_1 v_1 + b_2 v_2) \\ &= (\beta_1 - b_1) v_1 + (\beta_2 - b_2) v_2. \end{aligned}$$

Then we have $f(x) = k \pmod{n}$ and $\|x\| \le \frac{1}{2}(\|v_1\| + \|v_2\|)$. Since $\{v_1, v_2\}$ is a GLV generator, each component of v_1 and v_2 is bounded by $\sqrt{n}$ and we have $-\sqrt{n} < k_1, k_2 < \sqrt{n}$. Thus we see that one can always decompose $k = k_1 + k_2\lambda \pmod{n}$ with $-\sqrt{n} < k_1, k_2 < \sqrt{n}$ from any GLV generator $\{v_1, v_2\}$. Hence kP can be calculated by $k_1 P + k_2 \phi(P)$ using the windowed simultaneous multiple point multiplication method for P and $\phi(P)$ and the efficiency improvement is roughly 50% to the general scalar multiplication method for the currently recommended key sizes [4].

2.2 A Method to Find a GLV Generator by Gallant *et al.*

Gallant *et al.* suggested an algorithm of finding a GLV generator using extended Euclidean algorithm. In the procedure of extended Euclidean algorithm for n and λ, we have a sequence of equations,

$$s_i n + t_i \lambda = r_i, \quad i = 0, 1, 2, \cdots, \tag{2}$$

where $s_0 = 1$, $s_1 = 0$, $t_0 = 0$, $t_1 = 1$, $r_0 = n$, $r_1 = \lambda$ and s_i, t_i, r_i have the following properties,

$$\begin{aligned} r_i > r_{i+1} \geq 0, &\quad i \geq 0, \\ |s_i| < |s_{i+1}|, &\quad i \geq 1, \\ |t_i| < |t_{i+1}|, &\quad i \geq 0, \\ r_{i-1}|t_i| + r_i|t_{i-1}| = n, &\quad i \geq 1. \end{aligned}$$

Let m be the greatest index such that $r_m \geq \sqrt{n}$. Set $v_1 = (r_{m+1}, -t_{m+1})$ and take v_2 to be the shorter of $(r_m, -t_m)$ and $(r_{m+2}, -t_{m+2})$. From the above properties, it can be easily checked that each component of v_1 is less than $\sqrt{n}$. If each component of v_2 is also bounded by $\sqrt{n}$, then $\{v_1, v_2\}$ is a GLV generator. But Gallant *et al.* expected heuristically that v_2 would be also short without any proof. In addition, some cases could also occur so that there do not exist two linearly independent short vectors. Hence the method proposed by Gallant *et al.* only suggests a possible way of finding a GLV generator. For instance,

$$n = 85093, \quad \sqrt{n} \approx 291.707, \quad \lambda = 33206.$$

All solutions of the equation $f(i,j) = 0$, $|i| < \sqrt{n}$, $|j| < \sqrt{n}$ are

$$\pm(252, 246),\ \pm(210, 205),\ \pm(168, 164),\ \pm(126, 123),\ \pm(84, 82),\ \pm(42, 41),\ (0, 0).$$

Note that all of them are generated by one vector $(42, 41)$ over the field Q and v_2 does not exist under the assumption that each component of v_2 is bounded by $\sqrt{n}$.

In the next section, we shall present a necessary condition for the existence of a GLV generator and propose an algorithm to find the second vector v_2 whose components are bounded by $\sqrt{n}$ if it exists and compare with Gallant *et al.*'s algorithm.

3 Algorithm

3.1 A Necessary Condition for the Existence of a GLV Generator

As we've seen above, Gallant *et al.*'s method is not a complete solution of finding a GLV generator even though their method always gives the first small vector $v_1 = (r, t)$. In this subsection, we state and prove a necessary condition for the existence of a GLV generator.

Assume that $v_1 = (r, t)$ and $v_2 = (u, v)$ are two linearly independent integer vectors in the kernel of f such that $-\sqrt{n} < r, t, u, v < \sqrt{n}$. Then we have

$$r + t\lambda = sn, \quad u + v\lambda = wn, \tag{3}$$

for some $s, w \in Z$. By multiplying the first and the second equations in (3) by u and r, respectively, we get

$$(tu - rv)\lambda = (su - rw)n. \tag{4}$$

Similarly, we have

$$rv - tu = (sv - tw)n. \tag{5}$$

Note that $|tu - rv| < 2n$. If $tu - rv = 0$, (r, t) and (u, v) are linearly dependent. Thus $tu - rv = -n, n$ since n divides $tu - rv$. From (5) we have

$$sv - tw = -1, 1. \tag{6}$$

Therefore we conclude that t is relatively prime to s and v is also relatively prime to w. We shall state and prove the following Lemmas.

Lemma 1 *Let n be prime and $\lambda \in [1, n-1]$. Assume that $v_1 = (r, t)$, $v_2 = (u, v) \in kerf$ and $-\sqrt{n} < r, t, u, v < \sqrt{n}$. If v_1, v_2 are linearly independent, then r is relatively prime to t and u is relatively prime to v.*

Proof : Since $v_1, v_2 \in kerf$, we have $s, w \in Z$ which satisfy (3). Assume that the greatest common divisor of r and t is $\alpha > 1$. Then α becomes a common divisor of s and t from (3) since n is prime and this contradicts to (6). This completes the proof. □

Lemma 1 shows a necessary condition for the existence of a GLV generator. If $(r, t) \in kerf$, $\gcd(r, t) \neq 1$ and $|r| < \sqrt{n}, |t| < \sqrt{n}$, then the second vector $v_2 = (u, v)$, $|u| < \sqrt{n}, |v| < \sqrt{n}$ never exists. In fact, Lemma 1 itself shows that if $\gcd(r, t) \neq 1$, there is no GLV generator that contains the vector (r, t).

Lemma 2 *Let n be prime and $\lambda \in [1, n-1]$. If there is a vector $v = (r, t)$ in the kernel of f such that $\gcd(r, t) \neq 1$ and $-\sqrt{n} < r, t < \sqrt{n}$, then there exists no GLV generator.*

Proof : Suppose that $\{v_1, v_2\}$ is a GLV generator. It is easy to see that either $\{v, v_1\}$ or $\{v, v_2\}$ is a GLV generator containing v. This contradicts to Lemma 1. Therefore, there exists no GLV generator from Lemma 1. □

Suppose that $v = (r, t)$ is in the kernel of f and $-\frac{1}{2}\sqrt{n} < r, t < \frac{1}{2}\sqrt{n}$. Then $2v = (2r, 2t)$ is also contained in the kernel of f. Therefore, from Lemma 2, we know that at least one component of each vector in a GLV generator, say a, should satisfy either $\frac{1}{2}\sqrt{n} < a < \sqrt{n}$ or $-\sqrt{n} < a < -\frac{1}{2}\sqrt{n}$.

3.2 A Proposed Algorithm to Find a GLV Generator

Using the method proposed by Gallant *et al.* described in subsection 2.2, one can always get the first vector v_1 where each component of v_1 is bounded by $\sqrt{n}$. Now we present an algorithm of finding the second short vector v_2 after one gets the first vector v_1 if there is any such v_2.

Suppose we have the first vector $v_1 = (r_{m+1}, -t_{m+1})$ in the kernel of f as in Gallant *et al.*'s algorithm. We know that $|r_{m+1}|$, $|t_{m+1}|$ are already less than $\sqrt{n}$.

Finding v_2. Let $v_2 = (u, v)$ be the second vector so that $\{v_1, v_2\}$ is a GLV generator. Suppose $v_1 = (r, t), v_2 = (u, v)$ satisfy the equation (3) for some $s, w \in Z$. From the equation (6), we know that s is relatively prime to $-t$. We apply the extended Euclidean algorithm to find the greatest common divisor of s and $-t$. Then the algorithm returns v' and w' which satisfy

$$sv' - tw' = 1. \tag{7}$$

In general, every integer vector (v, w) which satisfies $sv - tw = 1$ can be represented by $(v' + \alpha t, w' + \alpha s)$, $\alpha \in Z$. Our purpose is to find a suitable α. Set

$$v = v' + \alpha t, \qquad w = w' + \alpha s.$$

Since $|v| < \sqrt{n}$ and $t = -t_{m+1} \neq 0$, we have

$$\begin{aligned} -\frac{v'}{t} - \frac{\sqrt{n}}{t} < \alpha < -\frac{v'}{t} + \frac{\sqrt{n}}{t}, \quad t > 0, \\ -\frac{v'}{t} + \frac{\sqrt{n}}{t} < \alpha < -\frac{v'}{t} - \frac{\sqrt{n}}{t}, \quad t < 0. \end{aligned} \tag{8}$$

Note that $u = wn - v\lambda$ and $r = r_{m+1} > 0$, then we also have

$$\frac{v'\lambda - w'n}{r} - \frac{\sqrt{n}}{r} < \alpha < \frac{v'\lambda - w'n}{r} + \frac{\sqrt{n}}{r}. \tag{9}$$

Therefore α has to be an integer in the intersection of (8) and (9). From Lemma 2, in order to seek α for the second vector v_2 of a GLV generator, it is sufficient to test only four integers at most since one of $|r|, |t|$ is greater than $\frac{1}{2}\sqrt{n}$. Now we give our algorithm of finding the second vector $v_2 = (u, v)$.

Algorithm 1 *Find a GLV generator $v_1 = (r, t), v_2 = (u, v)$, for given n and λ as above*

Input: n, λ
Output: v_1, v_2

Step 1. *Compute $v_1 = (r_{m+1}, -t_{m+1})$ such that $s_{m+1}n + t_{m+1}\lambda = r_{m+1}$ and $|r_{m+1}|, |t_{m+1}| < \sqrt{n}$ using the extended Euclidean algorithm to find the greatest common divisor of n and λ. (Gallant et al.'s algorithm)*

Step 2. *Check if each components of either $(r_m, -t_m)$ or $(r_{m+2}, -t_{m+2})$ is bounded by $\sqrt{n}$, stop and set the shorter of $(r_m, -t_m)$ and $(r_{m+2}, -t_{m+2})$ as the second vector v_2. Otherwise, go to step 3.*

Step 3. *Find any v' and w' such that*

$$s_{m+1}v' - t_{m+1}w' = 1.$$

For example, v' and w' are obtained from the extended Euclidean algorithm since s_{m+1} is relatively prime to $-t_{m+1}$.

Step 4. *Compute*

$$I_{11} = -\frac{v'}{t} - \frac{\sqrt{n}}{t}, \qquad I_{12} = -\frac{v'}{t} + \frac{\sqrt{n}}{t}.$$

Step 5. *Let* $I_1 = [I_{11}, I_{12}]$, *if* $t > 0$, *and* $I_1 = [I_{12}, I_{11}]$ *if* $t < 0$.
Step 6. *Compute*

$$I_{21} = \frac{v'\lambda - w'n}{r} - \frac{\sqrt{n}}{r}, \qquad I_{22} = \frac{v'\lambda - w'n}{r} + \frac{\sqrt{n}}{r}.$$

Step 7. *Let* $I_2 = [I_{21}, I_{22}]$.
Step 8. *Find all integers in the intersection of* I_1 *and* I_2 *and define them by* α. *Note that the number of* α'*s is at most 4. If there does not exist any such integer, stop.*
Step 9. *Set* $v_2 = (u, v)$, *where*

$$u = w'n - v'\lambda + \alpha r, \qquad v = v' + \alpha t.$$

One can easily verify that $v_2 = (u, v)$ is in $ker\, f$ and $|u|$, $|v| < \sqrt{n}$. Therefore $\{v_1, v_2\}$ is a GLV generator.

3.3 Comparison with the Method of Gallant *et al.*

To compare our algorithm with Gallant *et al.*'s algorithm, consider the following example,

$$n = 1319399, \quad \lambda = 344894, \quad \sqrt{n} \approx 1148.65.$$

We have from the extended Euclidean algorithm,

$$(r_m, -t_m) = (1812, -329), \quad (r_{m+1}, -t_{m+1}) = (871, 570),$$
$$(r_{m+2}, -t_{m+2}) = (70, -1469).$$

Therefore the two vectors from Gallant *et al.*'s algorithms are $v_1 = (871, 570)$ and $v_2 = (70, -1469)$. To proceed our algorithm, let $\tilde{v}_1 = (r, t) = (871, 570)$. We have $v' = -241$, $w' = -63$ since $s_{m+1} = (r_{m+1} - t_{m+1}\lambda)/n = 149$ and $-t_{m+1} = 570$. Then

$$I_{11} \approx -1.59237,\ I_{12} \approx 2.43798, \quad I_{21} \approx 1.76159,\ I_{22} \approx 4.39914.$$

The integer intersection of I_1 and I_2 is $\{2\}$. Therefore we have $\tilde{v}_2 = (-941, 899)$ from Step 9. We note that each component of $\tilde{v}_2$ in our algorithm is bounded by $\sqrt{n}$ while the second component of vector v_2 from Gallant *et al.*'s algorithm exceeds $\sqrt{n}$.

From the above example, we see that Gallant *et al.*'s algorithm does not give a GLV generator even if there is a GLV generator. Since our proposed algorithm checks all the possibility of the existence of a GLV generator with the first vector v_1, we never miss a GLV generator if there is any. Our proposed algorithm needs only a few more real number arithmetic after one gets the first vector v_1.

4 Conclusion

We presented an algorithm to facilitate the use of Gallant *et al.*'s idea to speed up the scalar multiplication of elliptic curves over a prime field in a more concrete way. Their method gives the desired efficiency if there is a GLV generator which we named in this paper. The efficiency improvement of their method is still good if one can guarantee that each component of the two linearly independent vectors is bounded by a small constant multiple of $\sqrt{n}$, i.e., $c\sqrt{n}$ when c is small. Analyzing cases with $c > 1$ is more complicated than the case with $c = 1$ and can be an further interesting work.

Acknowledgment

We appreciate the anonymous referees of the SAC 2002 for their comments and suggestions.

References

1. D. Bailey and C. Paar : *'Optimal extention fields for fast arithmetic in public-key algorithms'*, Advances in Cryptology-Crypto'98, Lecture Notes in Computer Science, Vol 1462, 1998, pp.472–485.
2. H. Cohen, A. Miyaji, and T. Ono : *'Efficient Elliptic Curve Exponentiation using Mixed Coordinates'*, Advances in Cryptology-Asiacrypt'98, Lecture Notes in Computer Science, Vol 1514, 1998, pp.51–65.
3. V. Miller : *'Use of Elliptic Curves in Cryptography'*, Advances in Cryptology-Crypto'85, Lecture Notes in Computer Science, Vol 263, 1986, pp.417–426.
4. R. Gallant, R. Lambert, and L. Vanstone : *'Faster Point Multiplication on Elliptic Curves with Efficient Endomorphism'*, Advances in Cryptology-Crypto'2001, Lecture Notes in Computer Science, Vol 2139, 2001, pp.190–201.
5. N. Koblitz : *'CM-curves with Good Cryptographic Properties'*, Advances in Cryptology-Crypto'91, 1992, 48, pp.279–287.
6. N. Koblitz : *'Elliptic Curve Cryptosystems'*, Mathematics of Computation, 1987, 48, pp.203–209.
7. C. Lim and P. Lee : *'More Flexible Exponentiation with Precomputation'*, Advances in Cryptology-Crypto'94, Lecture Notes in Computer Science, Vol 839, 1994, pp.95–107.
8. J. Solinas : *'An Improved Algorithm for Arithmetic on a Family of Elliptic Curves'*, Advances in Cryptology-Crypto'97, Lecture Notes in Computer Science, Vol 1294, 1997, pp.357–371.
9. J. Solinas : *'Efficient Arithmetic on Koblitz Curves'*, Design, Codes and Crytography, 2000, 19, pp.195–249.
10. V. Müller : *'Fast Multiplication on Elliptic Curves over small fields of charactersitic two'*, J. of Cryptology, 1998, 11, pp.219–234.