7 Security Fundamental Technologies
7-1 Recent Developments in Security Evaluation Technologies of Cryptosystems

Naoyuki SHINOHARA, Yoshinori AONO, and Takuya HAYASHI

Public key cryptosystems are important base technologies in information society and are actually used as in internet banking. The secure key length of a cryptosystem is estimated from the data of the world-top record of cryptanalysis, such as computational time. This paper explains developments of security evaluation technologies of pairing-based cryptosystems with η_T - pairing and lattice-based cryptosystems, respectively.

1 Introduction

Public key cryptosystems are fundamental technology widely used for supporting the information society. Typical examples of public key cryptosystems are RSA cryptography and elliptic curve cryptography. The security of public key cryptosystems is based on the difficulty to solve certain mathematical problems. For example, in elliptic curve cryptography, a cyclic group assigned by an elliptic curve is used, and if an elliptic curve discrete logarithm problem (ECDLP) over the cyclic group is solved, the elliptic curve cryptography can be deciphered. Therefore, for public key cryptosystems to operate safely, secure cryptography parameters (key length, etc.) must be estimated through deciphering experiments. Research institutes across the world are conducting research on the various mathematical problems associated with the security of public key cryptosystems. This paper describes the security evaluation of pairing-based and lattice-based cryptography.

Pairing-based cryptography is one of the public key cryptosystems currently being studied for practical application. High-performance cryptographic techniques that are difficult to implement in RSA cryptography and elliptic curve cryptography can be implemented using pairing-based cryptography. As an example of pairing-based cryptography, searchable encryption is explained in simple terms. In searchable encryption, the data and keywords can be searched while they are encrypted, so it is suitable for saving enciphered data on a server. It is suitable as a countermeasure against information leakage, and hence it is expected that pairing-based cryptography can be practically applied as a cryptosystem suitable for privacy protection. For pairing-based cryptography, the security is based on the difficulty of the calculations involved for solving ECDLP for an elliptic curve, and discrete logarithm problems over finite fields, so the size of the finite field used is an important element for determining the difficulty of the calculation. Also, the characteristics of the finite field are important factors for determining the security and processing speed of pairing-based cryptography. This paper describes the case where the characteristic is 3, which has been most reported in the research findings of high speed implementation. The technique for solving discrete logarithm problems over finite field GF ($3^{6\cdot 97}$) and its numerical experiment[1][2] are explained in concrete terms in Section **2**.

This paragraph outlines the security estimation of the lattice-based cryptography. From the viewpoint of long-term usage of a system, implementing quantum-resilient cryptosystems, that is, cryptography and practical systems which can keep secure against attacks by quantum computers, has been attracted. As mentioned above, RSA and elliptic curve cryptography, which are widely used around the world, can be broken by using quantum computers. On the other hand, lattice-based cryptography is a candidate for quantum-resilient cryptography. Furthermore, this kind of cryptography has many applications, represented by the realization of fully homomorphic encryption. This paper provides an overview of lattice-based attacks, which is a standard method for evaluating the security of lattice-based cryptography, and describes the evaluation of the LWE problem[3].

2 Security evaluation of pairing-based cryptography

In pairing-based cryptography that uses η_T pairing, a finite field GF (3^{6n}) where n is assumed to be a prime number is used. Function Field Sieve (FFS) is known as an algorithm for efficiently solving discrete logarithm problems over such finite fields of small characteristics. This paper focuses on discrete logarithm problems over GF ($3^{6\cdot 97}$), and describes an improved FFS that is suitable for solving such discrete logarithm problems, and the numerical experiment that used FFS.

2.1 Summary of function field sieve

It is known that the Function Field Sieve (JL06-FFS)[4] proposed by Joux and Lercier in 2006 is an efficient logarithm for solving discrete logarithm problems over GF(3^{6n}), where the extension degree n is 509 or less[5]. This section provides a summary when computing the solution X = logT of the discrete logarithm problem $T = g^X$ over a finite field GF (3^{6n}) using JL06-FFS. This function field sieve is comprised of the following four computation phases: polynomial selection phase, relation finding phase, linear algebra phase, and individual discrete logarithm phase.

Polynomial selection phase: First, $k \in \{1,2,3,6\}$ is selected, and the bivariate polynomial $H(x, y) \in \mathrm{GF}(3^k)[x, y]$ that satisfies the eight conditions[6] proposed by Adleman is determined. However, for the assigned integer d_H, we set $\deg_y H = d_H$. Also, for the assigned natural number d_m, the polynomial $m \in \mathrm{GF}(3^k)[x]$ with degree d_m is generated randomly, and a monic and irreducible polynomial $f \in \mathrm{GF}(3^k)[x]$ that satisfies the following conditions is calculated:

$$H(x, m) \equiv 0(\mathrm{mod}\, f), \deg f = 6n/k.$$

Here, the finite field GF(3^{6n}) is expressed by $\mathrm{GF}(3^k)[x]/(f)$ and there is a surjective homomorphism ξ from $\mathrm{GF}(3^k)[x, y]/(H)$ to $\mathrm{GF}(3^k)[x]/(f)$, such that $\xi(y) = m$. Next, for the assigned natural number B, the two factor bases $F_R(B)$ and $F_A(B)$ are determined as follows:

$$F_R(B) = \{\rho \in \mathrm{GF}(3^k)[x]: \deg \rho \leq B, \rho \text{ is monic and irreducible}\},$$

$$F_A(B) = \{< \rho, y - t >\in Div(GF(3^k)[x, y] \,/\, (H)): \rho \in F_R(B),\ H(x, t) \equiv 0(\mathrm{mod}\ \rho)\}.$$

However, Div (GF(3^k)[x, y]/(H)) is considered as the divisor group of $GF(3^k)[x, y]/(H)$, and $<p, y\text{-}t>$ is the divisor generated by p and y-t.

In this way, the initial values of the function field sieve in the polynomial selection phase are set. The computation time is so short that it can be ignored.

Relation finding phase: For the two assigned natural numbers R and S, we compute a sufficient number of pairs $(r, s) \in (\mathrm{GF}(3^k)[x])^2$ satisfying the following conditions:

$$\deg r \leq \mathrm{R}, \deg s \leq \mathrm{S}, \gcd(r, s) = 1, \tag{1}$$

$$rm + s = \textstyle\prod_{\rho_i \in F_R(B)} {\rho_j}^{a_i}, \tag{2}$$

$$(-r)^{d_H} H(x, -s/r) = \textstyle\prod_{\langle\rho_j, y-t_j\rangle \in F_A(B)} {\rho_j}^{b_j}. \tag{3}$$

However, a_i, b_j shall be non-negative integers. Let h be the class number of $\mathrm{GF}(3^k)(x)[y]/(H)$, and we assume that h is coprime to $(3^{6n} - 1)(3^k - 1)$. At this time, the following congruence expression holds for (r,s) that satisfies the conditions from (1) to (3):

$$\textstyle\sum_{\rho_i \in F_R(B)} a_i \log \rho_i \equiv \sum_{\langle\rho_j, y-t_j\rangle \in F_A(B)} b_j \log \sigma_j \pmod{(3^{6n} - 1)/(3^k - 1)}. \tag{4}$$

However, it shall be $\sigma_i = \xi(t_j)^{1/h}, \langle t_j\rangle = h\langle\rho_j, y - t_j\rangle$. The congruence expression (4) is called a relation.

Linear algebra phase: The linear equation is provided from a large number of the relations obtained in the relation finding phase. In the linear algebra phase, this linear equation is solved using the Lanczos method, etc., and the discrete logarithms of elements in factor base are obtained:

$$\log \rho_1, \ldots, \log \rho_{\#F_R(B)}, \log \sigma_1, \ldots, \log \sigma_{\#F_A(B)}.$$

Individual discrete logarithm phase: In this phase, the discrete logarithms of elements in factor base are correlated with the solution $\log T$ of the given discrete logarithm problem. In other words, the integers A_i, B_j that satisfy the following are derived using special-Q descent, etc.:

$$\log T \equiv \textstyle\sum_{\rho_i \in F_R(B)} A_i \log \rho_i + \sum_{\langle\rho_j, y-t_j\rangle \in F_A(B)} B_j \log \sigma_j \pmod{(3^{6n} - 1)/(3^k - 1)}.$$

2.2 Problem setting and parameter setting of FFS

From the point of view of evaluating the security of pairing-based cryptography, this paper discusses a multiplicative subgroup of GF($3^{6\cdot 97}$) wherein the order is 151-bit prime factor P_{151}. Also, the value reported in[7] is introduced for the initial value of the parameter when such discrete algorithm problems are solved by the function field sieve based on JL06-FFS. However, regarding the value of k, in our implementations, the respective calculation experiments were conducted with k=3, 6, and k=3 was selected from the result as it was considered suitable. As a result, the following was set for the parameters described in Subsection **2.1**:

$$(k, d_H, d_m, B, R, \mathrm{S}) = (3,6,33,6,6,6).$$

2.3 Improvements for function field sieve

This section describes the improvements for the function field sieve, introduced for solving discrete logarithm problems over $\mathrm{GF}(3^{6\cdot 97})$. For more details, see[1][2].

SIMD implementation of sieving: In order to check whether a certain (*r, s*) satisfies equations (2) and (3), naively, the polynomials of the left-hand side of equations (2) and (3) should be factorized. However, since polynomial factorization is relatively expensive, pre-processing called "sieving" is employed to reduce the overall cost of the relation finding phase. In this case, sieving is done for all (*r, s*), then for the remaining (*r, s*), whose polynomials are expected to satisfy equations (2) and (3) with a high probability, polynomial factorization is done to obtain the relations. To describe the sieving, let us take equation (2) as an example. When $rm + s$ is divisible by ρ_j, then $rm + s \equiv 0 \pmod{\rho_j}$. Hence, for fixed r, $s = s_0 + \kappa\rho_j$ $(\kappa \in \mathrm{GF}(3^3)[x])$, where $s_0 \equiv -rm \pmod{\rho_j}$, it satisfies $rm + s \equiv 0 \pmod{\rho_j}$. Due to this fact, doing the above calculation for a sufficient number of $\rho_j \in F_R(B)$ to collect information about the divisibility of $rm + s$ by ρ_j, one can check whether equation (2) holds for (*r, s*) without polynomial factorization. This can also be done for equation (3).

Based on the parameter $(B, R, S) = (6,6,6)$, the degree of polynomial $\mathrm{GF}(3^3)[x]$ appearing in sieving is at most 6. Also, the calculation of $-rm \pmod{\rho_j}$ can be done by computing $\bmod\ \rho_j$ after multiplication by x in $\mathrm{GF}(3^3)[x]$ progressively. Therefore, for sieving, it is sufficient to represent degree-7 polynomials in $\mathrm{GF}(3^3)[x]$. The Single Instruction Multiple Data (SIMD) approach is suitable for many such small targets. Figure 1 shows the data representation in our SIMD sieving. In this representation, GF(3) is represented by 2 bits $(h, l) \in \mathrm{GF}(2)^2$, and GF(3) operations can be described by bitwise operations (at least six operations are necessary[8]). $\mathrm{GF}(3^3)$ is represented as $\mathrm{GF}(3)[\omega]/(\omega^3 - \omega - 1)$, and the multiplication by x in $\mathrm{GF}(3^3)[x]$ can be described by left logical shift, and division by x in $\mathrm{GF}(3^3)[x]$ can be described by right logical shift. By this representation, at most, 16 polynomials in $\mathrm{GF}(3^3)[x]$ can be processed simultaneously.

Reducing variables by Frobenius map and Montgomery multiplication: In the linear algebra phase, the algorithms for solving the linear equations, e.g., the Lanczos method, requires $O(N^\epsilon)(2 < \epsilon \leq 3)$ modular multiplications for N variables. Hence, the complexity can be reduced by reducing the variables. Using the Frobenius map is an approach to reduce the variables[4][5]. For example, if an element ρ_i of factor base is mapped to another element ρ_i of factor base by the Frobenius map ϕ, then $\rho_j = \rho_i^{3^{97^2}} = \phi(\rho_i)$. Therefore,

$$\log \rho_j \equiv 3^{97^2} \log(\rho_i) \pmod{P_{151}}$$

and so $\log \rho_j$ can be removed from the linear equations. The variables can be reduced by this approach, but the coefficient will increase to approximately P_{151} (originally, all the coefficients of equation (4) are exponents of the right-hand side of equations (2) and (3), so the size is at most several tens), by multiplying by $3^{97^2} \pmod{P_{151}}$. Therefore, the complexity of modular multiplication will increase significantly. To reduce this increasing complexity, we introduce Montgomery multiplication for modular multiplication.

For an integer $A = 2^k$ (k is commonly selected as the word length of CPU) which is coprime to the prime number P_{151}, map each coefficient of linear equations to $A\,\mathbb{Z}_{P_{151}}$ by the map

$$\begin{aligned} \mathbb{Z}_{P_{151}} &\to A\mathbb{Z}_{P_{151}} \\ a &\mapsto Aa \pmod{P_{151}}, \end{aligned}$$

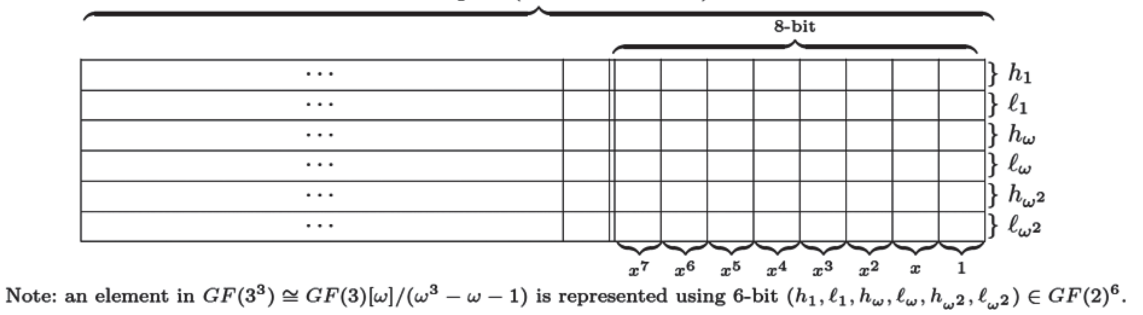

128-bit register (8-bit × 16 elements)
8-bit
...
h_1
ℓ_1
h_ω
ℓ_ω
h_{ω^2}
ℓ_{ω^2}
x^7 x^6 x^5 x^4 x^3 x^2 x 1
Note: an element in $GF(3^3) \cong GF(3)[\omega]/(\omega^3 - \omega - 1)$ is represented using 6-bit $(h_1, \ell_1, h_\omega, \ell_\omega, h_{\omega^2}, \ell_{\omega^2}) \in GF(2)^6$.

Fig. 1 SIMD representation for sieving

then the modular multiplication is performed by Montgomery multiplication $Aa \otimes Ab = A(ab) \pmod{P_{151}}$.

In Montgomery multiplication, since the modulo operation can be replaced by multiplications and logical shifts, the modular multiplication can be computed efficiently.

2.4 Experiment results

The experiment results are explained for each phase.

Relation finding phase: There were in all 187,602,242 relations (used factor bases were 134,697,663) obtained from sieving. Among these, 33,786,299 are free relations, which is a trivial relation obtained without sieving. These computations can be done in 53.1 days using 212 CPU cores, but the actual time taken was 118 days (including interruptions due to planned power outages and code improvements). The computation started on May 14, 2011, and finished on September 9, 2011.

Liner algebra phase: For a linear system with 187,602,242 equations and 134,697,663 variables, firstly the Frobenius map approach was taken to reduce the variables, and the number of variables was reduced to 45,059,572. Then, pre-processing called "filtering" was carried out, and the number of equations and variables were reduced to 6,141,443 and 6,121,440, respectively. This linear system was solved using the parallel Lanczos method. This computation can be done in 80.1 days using 252 CPU cores, but the actual time taken was 90 days. The computation started on January 16, 2012, and finished on April 14, 2012.

Individual logarithm phase: In order to obtain a linear relation between the given discrete logarithm problem and discrete logarithms of factor bases, the computation was done using 168 CPU cores, and the actual time taken was 15 days. The computation started on February 3, 2012, and finished on February 28, 2012. The computation of this phase can be done independently of the linear algebra phase, so the computations were done using another server during the linear algebra phase.

After the linear algebra phase was completed, the computation of the discrete logarithm problem and its verification were completed on April 24, 2012. That is, the computation of the discrete logarithm in the subgroup of $\mathrm{GF}(3^{6\cdot 97})$ where the order is P_{151} was successful. For the actual solution and the script for its verification, refer to[2].

3 Security estimation of lattice-based cryptography

The security of lattice-based cryptography is based on some types of lattice point search problems of lattice vectors. For example, the shortest vector problem to find a non-zero lattice point closest to the origin, the closest vector problem to find a lattice point closest to the target point, and the Learning With Errors (LWE) problem, have been investigated. In order to use such cryptography in the real world, appropriate setting of parameters such as key length is necessary. For this purpose, the relationship between the cryptographic parameter and the attacking time needs to be known to find secret information. Moreover, the cryptosystems used in the real world must have robust parameters, which means that finding secret data without a legitimate key is impossible in realistic time and equipment. In order to propose such parameters, not only experiments for attacking in small parameters but also simulation for large parameters are necessary. This section gives a brief outline on lattice-based attacks, which are a standard technique for analyzing lattice vector search problems, and as an example, gives an analysis for the LWE problem.

3.1 Outline of lattice-based attacks

In lattice-based cryptography, a problem that recovers secret information from the public data without a legitimate key can be converted into a lattice point search problem in general. That is, there is a lattice basis $L = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ that corresponds to the public information and a specific lattice point derives the secret information. The lattice-based attack has two steps which consist of employing a lattice basis reduction algorithm and a lattice vector search algorithm. In the former lattice basis reduction step, we use a single lattice basis reduction algorithm or combination of such algorithms, such as the LLL algorithm, the BKZ algorithm, etc. By lattice basis reduction, the computational time of the latter step is reduced. Hence, there is a trade-off relation of computational time between the two steps.

Lattice point search algorithm: As a simple example, we introduce the ENUM algorithm[9] for searching for the shortest vector in a lattice. The algorithm computes the Gram-Schmidt basis $(\boldsymbol{b}_1^*, \ldots, \boldsymbol{b}_n^*)$ of the input basis $(\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$, and performs depth-first search for the following tree:

- Each node is labeled by a lattice vector. Particularly, the root node has the zero vector.

- Each depth-k node has the vector $\boldsymbol{v} = \sum_{i=n-k+1}^{n} a_i \boldsymbol{b}_i$ ($\forall i, a_i \in \mathbf{Z}$) and its children are labeled by vectors of the form $\boldsymbol{v} + a_{n-k}\boldsymbol{b}_{n-k}$ ($a_{n-k} \in \mathbf{Z}$).

The depth of the tree is the same as the dimension of lattice n, and each leaf corresponds to a lattice point. In order to limit the searching range, in each depth k, the algorithm prunes the nodes when the projective length $|\pi_{n-k+1}(\boldsymbol{v})|$ of the corresponding vector v is larger than a threshold c. Here, c is the parameter to set the searching radius. This algorithm can enumerate all lattice vectors shorter than c.

When one wants to find the shortest vector of a given lattice, set $c = \min(|\boldsymbol{b}_1|, \dots, |\boldsymbol{b}_n|)$ and execute the above algorithm. If vectors are found, the solution is the shortest among the found vectors. Otherwise, that is, if no vector has been found, the shortest vector among the lattice basis $(\boldsymbol{b}_1, \dots, \boldsymbol{b}_n)$ is the desired vector.

Complexity of lattice point search algorithm: It is known that the computational complexity of the above searching algorithm is accurately approximated by the following formula[10]:

$$\mathrm{N}=\sum_{i=1}^{n} \frac{V_i(c)}{\prod_{j=n-i+1}^{n} |\boldsymbol{b}_i^*|}. \quad (5)$$

Here, $V_i(c)$ is the volume of an i dimensional sphere of radius c. Hence, the complexity of lattice point search can be predicted by only the searching radius and the lengths $(|b_1^*|, \dots, |b_n^*|)$ of the Gram-Schmidt basis. By the shape of the formula, decreasing $|\boldsymbol{b}_i^*|$ of the latter indexes, the complexity can decrease. Note that this corresponds to reducing $|\boldsymbol{b}_i^*|$ of the first indexes since the lattice volume $\prod_{i=1}^{n} |\boldsymbol{b}_i^*|$ is an invariant.

Lattice basis reduction algorithm: Performing a lattice reduction algorithm, the computational cost of lattice point search (5) can be decreased. In this section, we introduce the BKZ algorithm which is usually used for analyzing lattice-based cryptography.

The algorithm has the blocksize parameter β besides the input lattice basis B and performs the following basic operations for i=1,2,… successively: for the projective sublattice $L_{[i:i+\beta-1]} := \pi_i(\boldsymbol{b}_i, \dots, \boldsymbol{b}_{i+\beta-1})$ of dimension β, find the shortest vector and update the basis by using it. Remark that if i+β-1 exceeds n, replace the blocksize β as the suitable one, i.e., β=n-i+1. The vector found by the ENUM subroutine satisfies $|\pi_i(\boldsymbol{v})| \le |\pi_i(b_i)| = |\boldsymbol{b}_i^*|$. Thus, the updated $|\boldsymbol{b}_i^*|$ at index i is smaller than or equal to the previous one, which means that the latter elements become large and the complexity (5) can be smaller than before. After the process for i=n-1 is finished, return the index i to one. Repeating this reduction process, the complexity (5) is decreasing little by little. The algorithm terminates when $\boldsymbol{b}_i^*$ is the shortest vector of $L_{[i:i+\beta-1]}$ for all i. Since it calls the ENUM algorithm a "subroutine," a reasonable complexity estimation is possible.

This is an outline of the original version of the BKZ algorithm by Schnorr and Euchner[11], and several improving techniques are proposed. For example, in order to reduce the computational cost, abort the algorithm when the first vector is short enough for a considered problem[12], or use the searching radius c as the Gaussian-Heuristic which is known as the expected length of the shortest vector[13].

3.2 Simulation of lattice-based attacks

In order to estimate the complexity of attacking lattice based cryptography, the computational time of both the lattice basis reduction and lattice point search need to be simulated. As mentioned above, the complexity of lattice point search can be predicted by using the Gram-Schmidt basis lengths $(|b_1^*|, \dots, |b_n^*|)$ of the output of lattice basis reduction. Thus, our goal is to give a precise simulator to give the computation time and Gram-Schmidt lengths of the output of the lattice basis reduction algorithm with several parameters. The security estimation under this lattice-based model can be obtained by the lowest of the sum of simulated time for the lattice basis reduction and lattice point search among the parameters.

The simulation for output $(|\boldsymbol{b}_1^*|, \dots, |\boldsymbol{b}_n^*|)$ is performed by the BKZ simulator[13]. Below, the simulated value of $|\boldsymbol{b}_i^*|$ is denoted by ℓ_i, and in each index i of the BKZ algorithm. It is known that the length of the shortest vector of projective sublattice $L_{[i:i+\beta-1]}$ found by the ENUM subroutine can be simulated by the Gaussian-Heuristic

$$\mathrm{GH}(L_{[i:i+\beta-1]}) := V_\beta(1)^{-\frac{1}{\beta}} \cdot \mathrm{vol}(L_{[i:i+\beta-1]})^{\frac{1}{\beta}}.$$

Substituting the simulating value

$$\mathrm{vol}(L_{[i:i+\beta-1]}) = \prod_{j=i}^{i+\beta-1} \ell_j$$

of the volume of projective sublattice, the updated value of $|\boldsymbol{b}_i^*|$ can be simulated. Write the new value as ℓ_{i-new}. Since the volume of the sublattice is invariant during this process, the updated $|b_{i+1}^*|$ can also be replaced by $\ell_i \cdot \frac{\ell_i}{\ell_{i-new}}$. This is a simulation of the process for one index i.

As the real BKZ algorithm, executing the process for $i = 1, \dots, n-1$, the Gram-Schmidt lengths of output basis can be simulated, and the complexity of the algorithm is the sum of computational time of the ENUM algorithm overall.

3.3 Simulation of lattice-based attacks for the LWE problem

The LWE problem with the parameter set (n,m,q,s) is formulated as follows. From the input random matrix $A \in \mathbf{Z}_q^{n\times m}$ and the vector **b**, the goal is to find the error vector **e** and the secret vector **s** computed by

$$\mathbf{b} = A\mathbf{s} + \mathbf{e} \pmod{q}. \tag{6}$$

Here, s is randomly sampled from $\boldsymbol{Z}_q^n$, and each coordinate of **e** is independently sampled from the discrete Gaussian distribution of variance $\boldsymbol{s}^2$. In order to estimate the complexity of this problem, convert the (A,**b**) to a lattice vector search problem. Rewriting formula (6) by using an integer vector **w** as

$$\mathbf{b} = A\mathbf{s} + \mathbf{e} + q\mathbf{w} = [A \quad qI]\begin{bmatrix}\mathbf{s}\\ \mathbf{w}\end{bmatrix} + \mathbf{e}.$$

Hence, the secret vector **s** can be derived from the closest vector of **b** in the lattice defined by the columns of the matrix $[A \ qI]$. As mentioned in the above sections, the attacking time can be predicted by simulating the Gram-Schmidt lengths ($|\boldsymbol{b}_1^*|, \dots, |\boldsymbol{b}_n^*|$) after lattice basis reduction for the lattice and simulating the time of lattice point search.

3.4 Improvements of the lattice point search

In order to estimate the complexity of the LWE problem[3], we proposed the improvement of the lattice point search algorithm as follows. By using the property of discrete Gaussian distribution, the condition for alive in depth k is changed as

$$L_k \le |\pi_{n-k+1}(\boldsymbol{v} - \boldsymbol{b})| \le R_k\,. \tag{7}$$

In other words, the node that does not satisfy this is pruned. To reduce the total complexity, we tried to narrow the range$[L_k, R_k]$. The computational complexity of lattice point search in this setting is approximated by

$$\mathrm{N} = \sum_{i=1}^{n} \frac{\mathrm{Vol}C_k}{\prod_{j=n-i+1}^{n} |\boldsymbol{b}_i^*|}\,.$$

Here, C_k is the k dimensional object

$$\{(x_1, \dots, x_k) : L_k \le x_1^2 + \cdots + x_k^2 \le R_k\}$$

defined by the searching condition (7). Refer to [3] for details on the algorithm. Figure 2 shows a graph of bit security for typical parameters.

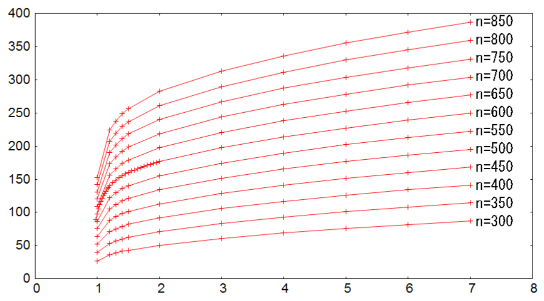


Fig. 2 bit-security=$\log_2$ (computation time [seconds]) for q=32749 and various s (horizontal axis)

4 Summary and future directions

The security of pairing-based cryptography is based on the difficulty to solve discrete logarithm problems for finite fields, and we have proposed a method for efficiently solving these problems. Using this method, we succeeded in solving discrete logarithm problems over a finite field GF($3^{6\cdot97}$), by implementing and performing experiments. Also, by using the properties of discrete Gaussian distributions which are not used in previous works, we improved the lattice point search and sped up the attack for LWE. These results are used to propose secure cryptographic parameters. In this laboratory, we continue working to improve the computation method for solving problems related to discrete logarithm problems and lattices, and are continuing with research and development on other mathematical problems related to the security evaluation of cryptography.

References

1 T. Hayashi, T. Shimoyama, N. Shinohara, and T. Takagi, "Breaking Pairing-Based Cryptosystems Using ηT Pairing over GF(3^{97})," ASIACRYPT 2012, LNCS 7658, pp.43–60, (2012).

2 T. Hayashi, T. Shimoyama, N. Shinohara, and T. Takagi, "Breaking Pairing-Based Cryptosystems Using ηT Pairing over GF(3^{97})," IACR Cryptology ePrint Archive 2012:345, (2012).

3 Y. Aono, X. Boyen, L. T. Phong, and L.Wang, "Key-private proxy re-encryption under LWE," INDOCRYPT 2013, LNCS 8250, pp.1–18, (2013).

4 A. Joux and R. Lercier, "The function field sieve in the medium prime case," EUROCRYPT 2006, LNCS 4004, pp.254–270, (2006).

5 T. Hayashi, N. Shinohara, L. Wang, S. Matsuo, M. Shirase, and T. Takagi, "Solving a 676-bit discrete logarithm problem in GF(3^{6n})," PKC 2010, LNCS 6056, pp.351–367, (2010).

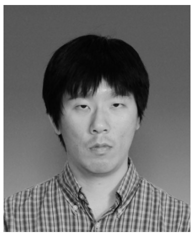
6 L. M. Adleman, "The Function Field Sieve," ANTS-I, LNCS 877, pp.108–121, (1994)

7 N. Shinohara, T. Shimoyama, T. Hayashi, and T. Takagi, "Key length estimation of pairing-based cryptosystems using ηT pairing over GF(3^n)," IEICE Transactions, vol.97-A, no.1, pp.236–244, (2014).

8 Y. Kawahara, K. Aoki, and T. Takagi, "Faster Implementation of eta-T Pairing over GF(3^m) Using Minimum Number of Logical Instructions for GF(3)-

Addition," Pairing 2008, LNCS 5209, pp.282–296, (2008).
9 R. Kannan, "Improved algorithms for integer programming and related lattice Problems", STOC 1983, pp.193–206, (1983).
10 N. Gama, P. Q. Nguyen, and O. Regev, "Lattice enumeration using extreme pruning", EUROCRYPT 2010, LNCS, vol. 6110, pp.257–278, (2010).
11 C. P. Schnorr and M. Euchner, "Lattice basis reduction: improved practical algorithms and solving subset sum problems", Math. Program., vol. 66, no. 1-3, pp.181–199,(1994).
12 G. Hanrot, X. Pujol, and D. Stehle,"Analyzing blockwise lattice algorithms using dynamical systems," CRYPTO 2011, LNCS, vol.6841, pp.447–464, (2011).
13 Y. Chen and P. Q. Nguyen, "BKZ 2.0: Better lattice security estimates," ASIACRYPT 2011, LNCS, vol. 7073, pp.1–20, (2011).

Naoyuki SHINOHARA, Ph.D.
Researcher, Security Fundamentals Laboratory, Cybersecurity Research Institute
Cryptanalysis, Computational Number Theory



Yoshinori AONO, Ph.D.
Researcher, Security Fundamentals Laboratory, Cybersecurity Research Institute
Cryptanalysis, Lattice Theory

Takuya HAYASHI, Ph.D.
Researcher, Security Fundamentals Laboratory, Cybersecurity Research Institute
Cryptanalysis, Efficient Implementation