Evolutionary Cryptography Theory-Based Generating Method for Secure ECs

Chao Wang
the Key Laboratory of Specialty Fiber Optics and Optical Access Networks, School of Communication and Information Engineering, Shanghai University, Shanghai 200072, China. the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA.

Feng Hu
the Key Laboratory of Specialty Fiber Optics and Optical Access Networks, School of Communication and Information Engineering, Shanghai University, Shanghai 200072, China.

Huanguo Zhang
the Key Lab of Aerospace Information Security and Trusted Computing, Wuhan University, Wuhan 430072, China.

Jie Wu
the Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122, USA.

Follow this and additional works at: https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology

Part of the Computer Sciences Commons, and the Electrical and Computer Engineering Commons

Recommended Citation
Chao Wang, Feng Hu, Huanguo Zhang et al. Evolutionary Cryptography Theory-Based Generating Method for Secure ECs. Tsinghua Science and Technology 2017, 22(5): 499-510.
Evolutionary Cryptography Theory-Based Generating Method for Secure ECs

Chao Wang, Feng Hu*, Huanguo Zhang, and Jie Wu

Abstract: Ant Colony Optimization (ACO) has the character of positive feedback, distributed searching, and greedy searching. It is applicable to optimization grouping problems. Traditional cryptographic research is mainly based on pure mathematical methods which have complicated theories and algorithm. It seems that there is no relationship between cryptography and ACO. Actually, some problems in cryptography are due to optimization grouping problems that could be improved using an evolutionary algorithm. Therefore, this paper presents a new method of solving secure curve selection problems using ACO. We improved Complex Multiplication (CM) by combining Evolutionary Cryptography Theory with Weber polynomial solutions. We found that ACO makes full use of valid information generated from factorization and allocates computing resource reasonably. It greatly increases the performance of Weber polynomial solutions. Compared with traditional CM, which can only search one root once time, our new method searches all roots of the polynomial once, and the average time needed to search for one root reduces rapidly. The more roots are searched, the more ECs are obtained.

Key words: ECs; Weber polynomial; evolutionary algorithm; ant algorithm; complex multiplication

1 Introduction

Traditional cryptographers are always skeptical of evolutionary algorithms that can solve cryptographic issues. They insist that cryptographic issues built on the basis of mathematical theory should be solved using pure mathematics.

The introduction of evolutionary thought in cryptography is not to be used instead of whole current cryptography system. It supports and promotes the improvement of the cryptography parts produced due to combinatorial optimization problem or search problem.

Zhang et al.[1, 2] proposed the concept and design methods for evolutionary cryptography theory by introducing the law of natural biological evolution. This is different from traditional cryptography methods and has become a principal framework for automatic cryptography design and cryptanalysis.

Elliptic curve cryptography results in the relatively high security performance of current passwords. In 2014, Pontie et al.[3] proposed a new scalar multiplication algorithm of the elliptic curve in an embedded system to resist side channel attacks. Then, He and Zeadally[4] gave an RFID scheme based on ECC in medical environments in 2015. And in the work of Thiranant et al.[5], EC was first applied to the QA authentication code of smart phones. ECC currently plays an increasing important role in this field[6–8].

Based on evolutionary cryptography theory, we propose two new methods for selecting secure Koblitz ECs. We tested the first method by selecting secure Koblitz ECs over the field $F(2^{800})$. The secure Koblitz
curve we obtained went beyond 700 bit, and its base field and base point went beyond the parameter range for Koblitz curves recommended by NIST. This is the first successful practice of evolutionary cryptography theory in public cryptography research. Then we tested the second method over the field $F(2^{2000})$, which is beyond the parameter range recommended by NIST for Koblitz curves. This experiment and the theoretical analysis also show that evolutionary cryptography theory can successfully design public key cryptosystems.

By reference to existing applications of Evolutionary Cryptography theory, we successfully applied Ant Colony Optimization (ACO) into Complex Multiplication (CM) to improve the solution efficiency of class polynomials.

The CM algorithm was used by Atkin and Morain for the construction of ECs with good properties in the context of primality proving. The method was also adapted to obtain curves with good security properties by Spallek and Lay and Zimmer independently. Now, it is a mature algorithm. Its detailed description has been issued by many international organizations for standardization, such as ANSI, IEEE, ISO, and NIST.

In this paper, we first discuss evolutionary thought in cryptography and design an ACO model to solve class polynomials in CM for practical application. In Section 2, we carry out a detailed analysis of traditional CM and point-out the shortcomings in its class polynomial parts. In Section 3, we devise a suitable Ant Colony Model to overcome the above shortcomings. The corresponding explanations of parameter settings, resource allocation, and information updating in this model are also shown in this section. The experimental results and corresponding analysis are shown in Section 4. We conclude with a simple summary and outlook.

2 Analysis for CM

In this section, we offer a detailed analysis of the CM implementation process by reference to the literature. Our introductions are shown in the following order:

- Choose a discriminant $D$;
- Construct the Weber polynomial using the discriminant $D$;
- Solve the Weber polynomial;
- Transform the root of the Weber polynomial to the corresponding root of the Hilbert polynomial;
- Construct elliptic curves using the root of the Hilbert polynomial.

Through the analysis, we can easily understand the shortcomings in CM. This section paves the way for the introduction of our Ant Colony Model in the next section.

2.1 Choose a discriminant $D$

Hasses theorem implies that $Z = 4p - (p + 1 - m)^2$ is positive. This in turn implies that there is a unique factorization $Z = Dv^2$, where $D$ is a square free positive integer. Consequently,

$$4p = u^2 + Dv^2$$

for some integer $u$ satisfying

$$m = p + 1 ± u$$

$D$ is actually a fundamental discriminant which is used by the CM method to determine a j-invariant and construct an EC of order $p + 1 - u$ or $p + 1 + u$.

To solve Eq. (1) when a suitable order is required, we can use two alternative approaches. The most straightforward one is to randomly generate pairs $(u, v)$, then check if the number $p$ that is constructed is a prime number. If it is, then we move to the next step of the CM method checking the two possible orders $m$ for suitability. Otherwise, we generate another pair $(u, v)$ and continue the same process. The second approach is to use Cornacchias algorithm. This algorithm solves a slightly different equation, $p = x^2 + Dy^2$, having as inputs a prime $p$ and the discriminant $D$. If a solution is found for the equation, then we proceed to the next step of the CM method setting $u = 2x$. Otherwise, another prime $p$ is chosen and we again apply Cornacchias algorithm.

2.2 Generate the Weber polynomial

A major problem in the earlier CM method is the construction of the Hilbert polynomials, as these require high precision floating points and complex arithmetic, which makes their computation very expensive. The root of the Hilbert polynomial is the j-invariant we are seeking.

To overcome this problem, cryptographic scholars have proposed two improvements:

- As the class polynomials depend only on $D$ (and not on $p$), they can be constructed in a preprocessing phase and stored for later use.
- Employ the Weber polynomial instead of the Hilbert polynomial. Their main differences are: (1) The coefficients of Hilbert polynomials grow excessively large as the discriminant $D$ increases, while for the same discriminant the Weber polynomials have much
smaller coefficients and thus are easier and faster to construct; (2) The roots of the Hilbert polynomial directly construct the EC, while the roots of the Weber polynomial have to be transformed to the roots of the corresponding Hilbert polynomial in order to construct the EC. This is the method we used.

The process of construction contains two parts: calculating the class invariants and realizing series multiplication. A detailed description is given in ANSI X9.63-2001 E.2.3. In addition, the process of constructions requires a large number of complex operations. We can reduce computing complexity or enhance computing speed using some clever skills.

Computing precision is another important factor when constructing polynomials. Different polynomials require different computing precision. Inadequate precision generates error coefficients, while higher precision slows the computing speed. Harald introduced a number of computing skills. In our design, we draw on these skills to improve the operation speed.

2.3 Find the roots of Weber polynomials

As mentioned above, the construction of the Weber polynomials can be preprocessed, but polynomials should be solved in real-time. Each root may correspond to an EC, so how to efficiently solve the Weber polynomial directly impacts the performance of CM.

This problem can be reduced to the solution of the arbitrary power of congruence equation. The most common method is to use a factorization algorithm to break down the high congruence equation into several linear factors in the form of series multiplication. Each linear factor corresponds to a root. ANSI X9.63-2001 E.1.4 lists such a factorization algorithm. The detail is shown in Algorithm 1.

Suppose we have a 5-power congruence equation based on the field of 2003 to solve.

\[ f(t) = t^5 - 167t^4 + 1646t^3 - 344t^2 + 1446t - 579 \mod 2003. \]

For such a simple polynomial, we can easily get its five roots using the brute force method. These five roots are \{1, 17, 29, 49, 71\}. Next, we solve this polynomial and show the detailed process using the above factorization algorithm for deep analysis. Through such an analysis, we found the existing shortcomings.

Step 2 in the factorization algorithm is the core. Its goal is to find a smaller power factor equation of \( g(t) \).

Algorithm 1 Factorization algorithm

\begin{itemize}
  \item \textbf{Input:} A prime \( p > 2 \), a positive integer \( d \), and a polynomial \( f(t) \), which factors modulo \( p \) into distinct irreducible polynomials of degree \( d \).
  \item \textbf{Output:} A random degree-\( d \) factor of \( f(t) \).
  \end{itemize}

\begin{enumerate}
  \item Set \( g(t) = f(t) \).
  \item While \( \deg(g) > d \):
    \item Choose \( u(t) \) random monic polynomial of degree \( 2d - 1 \);
    \item Compute \( c(t) = u(t)^{\frac{d}{2}} \mod g(t) \);
    \item Compute \( h(t) = \gcd(c(t) - 1, g(t)) \);
    \item If \( f(t) \) is constant or \( \deg(h) = \deg(g) \) then go to Step 2.1;
    \item If \( 2\deg(h) > \deg(g) \), then set \( g(t) = g(t)/h(t) \), else \( g(t) = h(t) \).
  \item Output \( g(t) \).
\end{enumerate}

As to the randomness of \( u(t) \) in Step 2.1, even for the same goal, the result may be different in a different run. For example, when \( g(t) = f(t) \), we present some cases based on different \( u(t) \).

(1) When \( u(t) = t + 3 \)

According to Step 2.2,

\[ c(t) = u(t)^{\frac{d}{2}} \mod g(t) = (t + 3)^{1001} \mod g(t) = 1376t^4 + 336t^3 + 370t^2 + 1142t + 783. \]

According to Step 2.3,

\[ c(t) - 1 = t^4 + 862t^3 + 830t^2 + 1046t + 1267 \mod 2003, \]

\[ h(t) = \gcd(c(t) - 1, g(t)) = t^3 - 121t^2 + 1596t - 1476. \]

It is composed of \( (t - 1), (t - 49), \) and \( (t - 71) \). In the actual process of computing, we do not know these factors in advance. Here, we list them for easy analysis. Two types of low power equations are generated after each factorization. They are \( h(t) \) and \( g(t)/h(t) \). In Step 2.5, we choose just the lower power factor \( g(t)/h(t) = t^2 - 46t + 493 \) as the next \( g(t) \).

(2) When \( u(t) = t + 452 \)

\[ h(t) = \gcd(c(t) - 1, g(t)) = t^3 - 137t^2 + 1513t - 1056. \]

It is composed of \( (t - 17), (t - 49), \) and \( (t - 71) \). We choose the lower power factor \( g(t)/h(t) = t^2 - 46t + 493 \) as the next \( g(t) \).

(3) When \( u(t) = t + 4 \)

\[ h(t) = \gcd(c(t) - 1, g(t)) = t^2 - 120t + 1476. \]

It is composed of \( (t - 49) \) and \( (t - 71) \). We choose the lower power factor \( g(t)/h(t) = t^2 - 120t + 1476 \) as the next \( g(t) \).

(4) When \( u(t) = t + 1321 \)

No lower power factor was found. As can be seen from the above analysis, there are three outcomes:

- The results of factorization may be the same, while choosing different \( u(t) \).
The results of factorization may be different, while choosing different \( u(t) \).

Sometimes there is no result. The factorization algorithm is a continuous cyclical process. As to the existence of the above three cases, the algorithm displays a lot of randomness. We express the factorization process as the following pattern:

As shown in Fig. 1, the three dark solutions in different layers and corresponding lines compose a search path. This contains two times factorization. First, we factor the solution set \{1, 17, 29, 49, 71\}, and find a two-power factor \{29, 71\}. Second, we factor the solution set \{29, 71\} and find a one-power factor \{71\}, which is one root of \( f(t) \).

Line \( \odot \) indicates that, using factorization algorithm, one solution set in the lower layer comes from a solution set in the upper layer. Starting from the top-level solution set, a path is composed of one solution in each layer and the lines between adjacent layers. We call this a search path. There are many search paths in Fig. 1. At the end of each search path, we find one root of the polynomial.

We reached the conclusion that the biggest drawback in the traditional factorization algorithm is the lack of information integration. If the EC is constructed by the j-invariant corresponding to 71, it does not satisfy our secure guidelines and we have to look for another root of \( f(t) \) by repeating above factorization algorithm. This process is a waste of time and full of uncertainties. As we can see from Fig. 1, we can easily get another root \{29\} after we get root \{71\}. If the traditional factorization algorithm could be combined with some intelligent evolutionary algorithm, such obvious effective information would not be wasted. In Section 3, we will discuss how to combine the traditional factorization algorithm with ACO to improve the solution efficiency.

2.4 Transform Weber roots to its corresponding Hilbert roots

The roots of the Weber polynomial are not equal to j-invariants. Therefore, how to rapidly transform them is a key issue.

In our research, the class invariants \( D \neq 0 \mod 3 \), \( D = 1, 2, 7 \mod 8 \), belong to relatively simple cases. In these cases, the degree of the Weber polynomial is the same as the Hilberts. Their roots are a one-to-one transformation relationship.

An interesting case is where \( D = 3 \mod 8 \). Here, the degree of the Weber polynomial is three times larger than that of the corresponding Hilbert polynomial. If the number of distinct roots of the Weber polynomial is exactly three times the number of the roots of the corresponding Hilbert polynomial, the transformations map three roots of the Weber polynomial into the same root of the Hilbert polynomial. Obviously, when the number of distinct roots of the Weber polynomial is equal to the number of distinct roots of the Hilbert polynomial, one root of the Weber polynomial is mapped to exactly one root of the Hilbert polynomial.

Because of the complication of \( D = 3 \mod 8 \), we do not consider this case. For the rest, transformation formulas are listed in Ref. [17].

\[
R_H = (R_W^{24} - 16)^3 / R_W^{24} \quad (D = 7 \mod 8, \; D \neq 0 \mod 3),
\]

\[
R_H = (2^{12} R_W^{-24} - 16)^3 / 2^{12} R_W^{-24} \quad (D = 3 \mod 8, \; D \neq 0 \mod 3),
\]

\[
R_H = (2^6 R_W^{12} + 16)^3 / 2^6 R_W^{12} \quad (D = 2, \; 6 \mod 8, \; D \neq 0 \mod 3),
\]

\[
R_H = (2^6 R_W^{-12} - 16)^3 / 2^6 R_W^{-12} \quad (D = 1 \mod 8, \; D \neq 0 \mod 3),
\]

\[
R_H = (2^6 R_W^6 - 16)^3 / 2^6 R_W^6 \quad (D = 7 \mod 8, \; D \neq 0 \mod 3).
\]

Here, \( R_W \) expresses the root of the Weber polynomial. \( R_H \) is the corresponding root of the Hilbert polynomial. In practice, we chose a different transformation formula according to the type of \( D \).

\( D = 1, \; 2, \; 3, \; 6, \; 7 \mod 8 \) and \( D \neq 0 \mod 3 \).
2.5 Security analysis

The standard security guidelines quote from ANSI X9.63:
- The order $E(F_q)$ is divisible by a large prime $n > 2^{160}$.
- The MOV condition holds.
- The anomalous condition holds.
- Furthermore, to guard against possible future attacks against special classes of non-supersingular curves, it is prudent to select an EC at random.

The ECs generated according to security guidelines above have the ability to resist current ECC attacks, such as the exhaust algorithm and attachs such as the Pohlig-Hellman\cite{18}, Baby-Step Giant-Step, Pollards Rho\cite{19}, SSAS (Smart2Satoh2Araki2 attack), MOV\cite{20}, and FR.

The side-channel attack has been a hot research topic in recent years. This kind of attack utilizes some deficiencies in ECC algorithms to obtain the key indirectly through statistical methods, and has nothing to do with the security of ECs parameters. Therefore, we did not consider such attacks in this research.

To the best of our knowledge, sometimes a security requirement regarding the degree $h_D$ of the class field polynomial exists. Such requirement is only posed by the German Information Security Agency, which requires that $h_D$ should be greater than 200. The reason is that there are few ECs generated from class field polynomials with smaller degrees amenable to specific attacks. However, no such attacks are known to date and this requirement does not seem to be part of the security standard. Despite this fact, we have taken into consideration such large values of $h_D$ in our study.

3 Introduction of Evolutionary Algorithms

3.1 Evolutionary computing and ant colony optimization

Evolutionary thought has resulted in great achievements when solving combined optimization problems such as the Traveling Salesman Problem (TSP), 0/1 knapsack problem, packing problem, and production scheduling optimization problem. The analysis and design used in cryptography can come down to search and optimize NP-hard problems. Therefore, it is reasonable to assume that cipher analysis and design can be generated automatically using evolutionary computing.

Evolutionary thought overcomes the limitations of random testing and exhaustive searching. It can search by guidance under the objective function. In the traditional method, to quickly design or decipher a cryptosystem, we need to understand the mathematical properties inside the cipher algorithm in depth or employ exhaustive searching. Compared with traditional mathematical cryptography design and analysis, the application of evolutionary thought can lead to automatic cryptography design and analysis.

Based on evolutionary thought, Zhang and Tan\cite{2} proposed evolutionary cryptography thought in 1999. He successfully applied evolutionary cryptography thought to construct a high security level S-box for DES\cite{1}. In addition, he also achieved good results with the Bent function\cite{21} and DES cryptanalysis\cite{22, 23}. Meanwhile, Clark et al.\cite{24} proposed the introduction of an intelligent algorithm into cryptography and obtained some research achievements in S-box design, Boolean function\cite{25}, and cryptanalysis\cite{26}. In Refs. [24–26], the authors considered the possibility of fully automating the decrypting procedure. A straightforward implementation turns out to be incapable of decrypting harder cryptograms due to random variation in the bigram heuristic. His work shows that the pheromone feedback mechanism of an Ant Colony System is capable of overcoming some random variation and decrypting a wider variety of messages. So, it is an effective and meaningful method of obtaining a highly secure level cipher using evolutionary thought. In this paper, we mainly use ACO to achieve our goal.

ACO is one of the most classical evolutionary computing algorithms. It takes inspiration from the foraging behavior of some ant species. These ants deposit pheromones on the ground to mark some favorable path that should be followed by other members of the colony. ACO exploits a similar mechanism for solving optimization problems.

Since 1991, different ACO algorithms have been proposed, such as the Ant Colony System\cite{27}, Ant-Q System\cite{28}, MAX-MIN Ant System\cite{29}, and Best-Worst Ant System\cite{30}. Here, we choose the Ant Colony System algorithm, whose characteristics include local and global updates of the pheromone, and a pseudo-random-proportional of state transition rule. It can effectively reduce the probability of producing the same solutions generated by some ants through the same iteration. In this paper, this reduces the probability of repeated solutions after aggregation of data generated
in the same iteration, thereby increasing the efficiency of the target solution searching. The use of a tabu list is more suitable for storing the results of invalid factorization.

The ant colony algorithm has the characteristics of distributed computing, non-central control, and distributed indirect communication between individuals. It can be easily combined with other optimization algorithms. ACO has been widely applied to function optimization, system identification, data mining, network routing, and so on.

In this paper, we use ACO, with its characteristics of information integration and information feedback, to solve a cryptographic problem. ACO does not involve complex mathematical operations, so is suitable for solving complicated cryptographic problems.

3.2 Ant colony model design

Here, we combined the traditional factorization algorithm with ACO to find roots more efficiently. Compared with the TSP, we designed our model with no complicated theory. We chose the most effective model after trying a number of possibilities.

In Fig. 2, we show the corresponding flow chart for this model. A detailed text description is given below.

Step 1 Initialization. The Ant Colony Model allocates a Weber polynomial to an ant for the first iteration.

Step 2 Search. Ant finds a factor polynomial Poly₁ by implementing Algorithm 1.

Step 3 Local Update. Local Update occurs after an ant finishes its mission. Then having the Search Target divided by Poly₁, we get the other factor polynomial Poly₂.

Step 4 Global Update. Global Update occurs at the end of the iteration. According to the statistics of Poly₁ and Poly₂, Global Update generates some necessary parameters including search target and the number of ants for next iteration.

Step 5 Information feedback, as shown in Fig. 3. If the number of ants for next iteration is equal to 0, the next step goes to Step 6 or else goes to Step 2.

Step 6 Re-initialization. Allocate Search Targets to different ant for the new iteration.

Step 7 Summary.

3.3 Implementation process in detail

We assumed that the Weber polynomial was composed of seven roots \{A B C D E F G\}. Then we tried to solve it using Ant Colony Model. The search path may be different in different tests. We show the detailed search process for each step in Fig. 4. Another possible search path is shown in Fig. 5. It did not matter that search paths were different, we always got all roots.

Solution Set: We call each cycle a Solution Set, which is composed of some roots denoted by letters. Sometimes it could be a search target, while the next time it would be a search result or Local Update result.

Search Line: We call the line connecting two contiguous Solution Sets a Search Line. Each ant finishes its search mission on the Line.

Ant: Global Update allocates goal polynomials to different ants as their missions. In our model, each ant knows the factorization algorithm. They break down goal polynomials using this algorithm.

Search Result: This kind of set is used to store the search results for each ant. After the Global Update, the

Fig. 2 Model of the ant colony designed for solving the Weber polynomial.

Fig. 3 Flow chart corresponding to Fig. 2
Search Result is transformed to the goal polynomial for the next iteration.

**Local Update:** The Search Result is the factor of the goal polynomial, so we can get another factor by dividing the goal polynomial by the Search Result. There is no overlap between these two results. For example, with the target equation \( \{A \ D \ C \ D \ E \ F \ G\} \) divided by the search result \( \{B \ E \ G \ F\} \), we get another result \( \{A \ C \ D\} \). Obviously, \( \{B \ E \ G \ F\} \) and \( \{A \ C \ D\} \) are mutually exclusive.

**Local Update result:** The other factor of the goal polynomial, generated using the Local Update.

**Global Update:** Through the Global Update, the model generates some necessary parameters including the Search Target and the number of ants for the next iteration. In this sense, the model has the ability of autonomy to properly allocate the system resources and reduce unnecessary human intervention.

As shown in Fig. 3, the feedback plays a very important role in our model. It is used to allocate system resources for the next iteration according to the last. Actually, it belongs to the Global Update.

This type of the search in Fig. 4 contains three iterations and costs six ants for factorization. At the end of the search, we find all seven roots.

### 3.4 Evolutionary CM

Based on above experimental verification, we combined the Ant Colony Model with traditional CM to construct a complete Evolutionary CM algorithm. The
description is shown below.

**Step 1** Determine the suitable scope of $D$ according to current experiment hardware configuration. Then, construct all Weber polynomials according to ANSI X9.63-2001 E1.4. Record all Weber polynomials in DataBase.

**Step 2** Randomly select a square-free positive integer as the discriminant $D$. $D$ should satisfy the congruence conditions listed in ANSI X9.63-2001 E3.2.1. Or else, randomly select another value.

**Step 3** Check if $D$ is the discriminant of base field $p$ according to ANSI X9.63-2001 E3.2.2. If not, go to Step 2.

**Step 4** Give the scope of cofactor $k$. Calculate the curve order $u$ and the base point order $r$ with the algorithm ANSI X9.63-2001 E3.2.3. The relationship between these two orders is $u = kr$.

**Step 5** Query and load the Weber polynomial from DataBase according to $D$.

**Step 6** Send the Weber polynomial loaded in Step 5 to above Ant Colony Model to find all roots.

**Step 7** Construct EC $y^2 = x^3 + ax + b$ with the algorithm ANSI X9.63-2001 E3.4.1. Each root could construct an EC.

Check if the ECs constructed in Step 7 satisfy the security guidelines listed in ANSI X9.63-2001 H.1.

4 Experimental Procedure and Results Analysis

4.1 Device configuration

CPU: AMD sempron 2800+; memory: 256 MB; hard disk: 80 GB.

4.2 Preprocessing of the Weber polynomial

We constructed Weber polynomials from 7 to 30000, which meet $D \equiv 1, 2, 5, 7 \mod 8$. Then we recorded them down for easily access. Because of its low efficiency, we did not use the case where $D \equiv 3 \mod 8$ in our research.

4.3 Solve the Weber polynomial

Let $W(D)$ denote the Weber polynomial. We find suitable pairs $p(D)$ first, then try to solve the Weber polynomial $W(D)$ mod $p$.

Experiment shows that there are two major factors impacting the speed of search: the degree of the polynomial and the magnitude of the field. We list three sets of data from our experiment based on three different magnitude of field (192 bits in Fig. 6, 256 bits in Fig. 7, and 384s bit in Fig. 8) using original factorization and our Ant Colony Model.

$D$ is the discriminant for CM. $hD$ is the degree of the Weber polynomial corresponding to $D$.

TT-Ant is the abbreviation of the string total time of our Ant Colony Model.

![Fig. 6 Three fitting curves based on 192 bits.](image)

![Fig. 7 Three fitting curves based on 256 bits.](image)

![Fig. 8 Three fitting curves based on 384 bits.](image)
AT-Ant is the abbreviation of the string average time for each root of our Ant Colony Model.

TT-Original is the abbreviation of the string total time of original factorization algorithm.

The biggest difference between the original factorization algorithm and our Ant Colony Model is that the former only gets one root each time, whereas our model gets all the roots once.

As can be seen from Figs. 6–8, the TT-Ant fitting curve is about 2.1–2.5 higher than the TT-Original. On the surface, our model would consume more computing time than the original factorization algorithm. But our model would get all the roots, while the original algorithm just gets one root. Therefore, in the view of average time for each root, our model consumes much less than the original algorithm. The comparison made between the AT-Ant fitting curve and the TT-Original fitting curve is most straightforward and meaningful.

Theoretically, the original factorization algorithm takes, on average, $hD/2$ times factorization to get a root, while our Ant Colony Model takes a fixed $hD$ times factorization to find all roots. One root costs one factorization, so our model has an efficiency $hD/2$ times than that of the traditional factorization algorithm. Usually, $hD \gg 2$, so $hD/2 \gg 1$. Obviously, the solution efficiency is greatly improved after combination with Ant Colony Model.

### 4.4 Experiment for Evolutionary CM

We described the detailed Evolutionary CM algorithm Section 3.4. In practice, we made some improvements. In Fig. 9, we show the model of Evolutionary CM used in our experiment.

In Table 1, we list ECs parameters (such as $p, a,$ and $b$) and few important immediate parameters (such as Weber roots and Hilbert roots) in our experiment. Limited by space, we have listed 3 ECs (of total 125 ECs) generated by using the Evolutionary CM in Fig. 9.

There are three main parts to construct this model: Thread One, Thread Two, and pre-computation. An important feature of the Ant Colony Model introduced in Section 3 is that TT-Ant is larger than TT-Original. So, the waiting time before constructing ECs is too long when $hD$ is large. To overcome this shortcoming, we employ the two-thread technology in our experiment.

Thread One is used to find the root, while Thread Two is used to construct ECs. Cornacchia’s algorithm module, security guidelines module, and the Ant Colony Model module are embedded in Thread One. The transformation module and construction module are embedded in Thread Two. Once a root is found in Thread One, it is immediately sent to Thread Two. The stack in Thread Two is used to temporarily store the roots that cannot be handled at that time.

### 5 Conclusion

We completed research on the searching roots of the Weber polynomial based on evolutionary thought, and successfully combined the ACO with traditional CM. From the mathematical point of view, our research could be used to solve high degree power congruence polynomials. Experiments show that the introduction of an Ant Colony Model makes the full use of the last iteration results and automatically deploys resources for the next iteration. The efficiency of CM is improved and is directly reflected in the average time consumption. This means that we can generate more security ECs than the original CM in the amount of time.

### Acknowledgment

This research was supported by the National Natural Science Foundation of China (Nos. 61332019, 61572304, 61272056, and 60970006), the Innovation Grant of Shanghai Municipal Education Commission (No. 14ZZ089), and Shanghai Key Laboratory of Specialty Fiber Optics and Optical Access Networks (No. SKLSFO2014-06).

### References

[1] H. G. Zhang, X. T. Feng, Z. P. Tan, and Y. Z. Liu, Evolutionary cryptosystems and evolutionary design for DES, (in Chinese), *Journal of China Institute of Communications*, vol. 23, no. 5, pp. 57–64, 2002.

[2] H. G. Zhang and Z. P. Tan, *Evolutionary Cryptosystem*, (in Chinese). Wuhan University Press, 2011.
Table 1  ECs generated by using our evolutionary CM (521 bits, 4 = D = 11519, hD = 125).

| Parameter | Value |
|-----------|-------|
| \( \rho \) | 3 683 491 264 995 834 878 569 767 231 831 807 980 983 083 030 084 688 216 682 264 551 399 991 735 855 787 842 733 678 706 371 522 681 823 240 394 763 578 095 929 267 964 632 623 921 961 580 563 300 651 061 |
| \( \#E_\rho \) | 9.20 872 816 248 958 719 642 441 807 957 951 955 245 770 757 712 422 054 170 566 137 849 997 933 963 946 098 706 035 149 799 064 370 311 299 942 894 092 690 565 312 038 831 947 931 637 554 246 328 503 511 485 763 |
| Weber roots | 2 237 209 703 838 095 851 972 454 675 046 771 308 497 674 565 774 075 422 128 152 331 961 333 264 123 033 600 033 943 023 041 834 490 497 137 841 513 553 854 457 572 791 028 993 733 942 038 111 055 042 500 799 |
| (Curve-1 to Curve-3) | 2 381 694 581 007 604 906 232 949 922 180 794 842 755 140 280 895 422 315 781 362 444 381 628 382 610 234 789 240 433 316 818 906 033 099 742 725 255 731 334 787 205 916 913 240 174 023 257 118 642 756 045 315 |
| Curve-2 | 1 256 030 102 032 038 558 505 389 091 466 160 150 941 303 076 845 910 871 769 204 223 286 228 493 191 530 114 253 600 999 415 457 379 240 240 548 547 812 079 647 397 743 087 800 124 584 433 975 821 099 759 674 |
| Hilbert roots | 4 000 944 510 583 589 816 633 196 682 579 603 139 782 959 853 357 116 583 352 496 345 437 951 355 158 832 150 984 815 264 386 303 672 753 903 527 725 973 602 306 320 820 010 303 167 182 201 774 525 868 239 903 191 271 124 741 310 |
| (Curve-1 to Curve-3) | 1 731 974 222 724 236 484 249 058 160 432 279 814 485 539 139 414 759 823 402 773 561 454 286 648 335 449 044 005 997 826 394 916 106 355 852 537 604 857 137 662 213 066 069 728 208 858 376 320 424 739 116 405 |

Notes: Gx and Gy are the base points of the ECs.

[3] S. Pontie, P. Maistri, and R. Leveuge, An elliptic curve crypto-processor secured by randomized windows, in *Proc. 17th Euromicro Conf. on Digital System Design (DSD)*, Verona, Italy, 2014, pp. 535–542.

[4] D. He and S. Zeadally, An analysis of RFID authentication schemes for internet of things in healthcare environment using elliptic curve cryptography, *IEEE Internet of Things Journal*, vol. 2, no. 1, pp. 72–83, 2015.

[5] N. Thiramant, Y. S. Lee, and H. Lee, Performance comparison between RSA and elliptic Curve cryptography-based QR code authentication, presented at the IEEE, International Conference on Advanced Information.
NETWORKING and Applications Workshops IEEE, Gwangju, South Korea, 2015.

[6] J. W. Xu, K. D. Wang, C. Wang, F. Hu, Z. H. Zhang, S. Y. Xu, and J. Wu, Byzantine fault-tolerant routing for large-scale wireless sensor networks based on fast ECDSA, Tsinghua Science and Technology, vol. 20, no. 6, pp. 627–633, 2015.

[7] H. Jia, C. Wang, and J. Gu, Error bit correction of ECC based on grover quantum intermediate encounter search algorithm, Netinfo Security, vol. 2, no. 6, pp. 28–34, 2016.

[8] Y. H. Chen, H. H. Jia, and L. Y. Jiang, ECC scanning attack based on grover algorithm, Netinfo Security, vol. 2, pp. 28–32, 2016.

[9] C. Wang, H. G. Zhang, and L. L. Liu, The experiment of Koblitz elliptic curves generating based on evolutionary cryptography theory and verifying the parameters recommended by NIST, China Communications, vol. 8, no. 4, pp. 41–49, 2016.

[10] C. Wang, H. G. Zhang, and L. L. Liu, Evolutionary cryptography theory based generating method for secure Koblitz EC and its improvement by HMM, Science China Information Sciences, vol. 55, no. 4, pp. 911–920, 2012.

[11] A. O. L. Atkin and F. Morain, Elliptic curves and primality proving, Mathematics of Computation, vol. 61, no. 203, pp. 29–68, 1993.

[12] A. M. Spallek, Konstruktion einer elliptischen Kurve über einem endlichen Körper zu gegebener Punktengruppe, Master Thesis, Universität GH Essen, Diman, the Netherland, 1992.

[13] G. J. Lay and H. Zimmer, Constructing elliptic curves with given group order over large finite fields, Lecture Notes in Computer Science, vol. 887, pp. 250–263, 1994.

[14] G. Cornacchia, u di un metodo per la risoluzione in numeri interi dell equazione, Giornale di Mathematice di Battaglini, vol. 46, no. 1, pp. 33–90, 1980.

[15] IEEE P1363/D13, Standard specifications for public key cryptography, draft ballot, http://www.groupie.ieee.org/groups/1363/tradPK/draft.html, 1999.

[16] B. Harald, Efficient algorithms for generating elliptic curves over finite fields suitable for use in crytography, Ph.D. Dissertation, Dept. of Computer Science, Technical Univ. of Darmstadt, Darmstadt, German, 2002.

[17] V. Joachim, G. Zur, and P. Daniel, Factoring polynomials over finite fields: A survey, Journal of Symbolic Computation, vol. 31, nos. 1&2, pp. 3–17, 2001.

[18] S. C. Pohlig and M. E. Hellman, An improved algorithm for computing logarithms over GF(F) and the cryptographic significance, IEEE Transactions on Information Theory, vol. 24, no. 1, pp. 106–110, 1978.

[19] P. Oorschot and M. J. Wiener, Parallel collision search with cryptanalytic applications, Journal of Cryptology, vol. 12, no. 1, pp. 1–28, 1999.

[20] A. T. Menezes, T. Okamoto, and S. A Vanstone, Reducing elliptic curve logarithms in a finite field, IEEE Transactions on Information Theory, vol. 39, no. 5, pp. 1639–1646, 1993.

[21] Q. Shu, H. G. Zhang, and Z. Y. Wang, Evolutionary design for Bent function, Journal of Electronics, vol. 32, no. 11, pp. 1901–1903, 2004.

[22] J. Song, H. G. Zhang, and Q. S. Meng, Cryptoanalysis of four-round DES based on genetic algorithm, presented at the Proceedings of the International Conference on Wireless Communications, Networking and Mobile Computing, Shanghai, China, 2007.

[23] F. Yang, J. Song, and H. Zhang, Quantitative cryptoanalysis of six-round DES using evolutionary algorithms, presented at the Proceedings of the International Symposium on Intelligence Computation and Applications, Wuhan, China, 2008.

[24] J. A. Clark, J. L. Jacob, and S. Stepney, The design of Sboxes by simulated annealing, New Generation Computing, vol. 23, no. 5, pp. 219–231, 2005.

[25] J. A. Clark, J. L. Jacob, S. Stepney, S. Maritra, and W. Willan, Evolving Boolean functions satisfying multiple criteria, Lecture Notes in Computer Science, vol. 2551, pp. 246–259, 2002.

[26] A. Clark and E. Dawson, A parallel genetic algorithm for cryptanalysis of the polyalphabetic substitution cipher, Cryptologia, vol. 21, no. 2, pp. 129–138, 1997.

[27] L. M. Gambardella and M. Dorigo, Solving symmetric and asymmetric TSPs by ant colonization, in Proceedings of the IEEE Conference on Evolutionary Coomputation, Nagoya, Japan, 1996.

[28] L. M. Gambardella and M. Dorigo, Ant-Q: A reinforcement learning approach to the traveling salesman problem, in Proceedings of the ML-95, the 12th International Conference on Machine Learning, Tahoe City, CA, America, 1995.

[29] T. Stutzle and H. H. Hoos, MAX-MIN ant system, Future Generation Computer Systems Journal, vol. 16, no. 8, pp. 889–914, 2000.

[30] O. Cordon, I. D. Viana, F. Herrera, and L. Moreno, A new ACO model integrating evolutionarly computation concepts: The best-worst ant system, presented at the ANTS2000–From Ant Colonies to Artificial Ants, Brussels, Belgium, 2000.

Feng Hu received the BS degree from Xi’an University of Science and Technology in 2008. Currently, he is a PhD student at Electronic and Information Engineering Dept. of Shanghai University. His research interests include information security and quantum computing cryptography.

Huanguo Zhang is currently a professor and PhD supervisor at the School of Computer of Wuhan University. He graduated from Xidian University in 1970 with a bachelor degree. His research interests include information security and trusted computing.
Jie Wu is the Chair and the Laura H. Carnell Professor at the Department of Computer and Information Sciences at Temple University. Prior to joining Temple University, USA, he was a program director at the National Science Foundation and a distinguished professor at Florida Atlantic University. He received the PhD degree from Florida Atlantic University in 1989. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including *IEEE Transactions on Computers*, *IEEE Transactions on Service Computing*, and *Journal of Parallel and Distributed Computing*. He was the general co-chair/chair for IEEE MASS 2006 and IEEE IPDPS 2008 and program co-chair for IEEE INFOCOM 2011. Currently, he is serving as a general chair for IEEE ICDCS 2013 and ACM Mobi Hoc 2014, and program chair for CCF CNCC 2013. He was an IEEE Computer Society Distinguished Visitor, ACM Distinguished Speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). He is a CCF Distinguished Speaker and a fellow of the IEEE. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.

Chao Wang received the PhD degree from Tongji University in 1999. Currently, he is an IEEE senior member, council member of China Association of AI, Information Security Committee Vice Chair of China Electronics Institute, Committee member of the Sixth Shanghai Expert Committee for Informatization, Directorate of China Artificial Intelligence Institute, Committee member of CCF, IEEE Shanghai Section Secretary, IEEE Shanghai CAS Chapter Vice Chair, and IEEE Shanghai Computer Chapter Vice Chair. His research interests include wireless sensor network, network information security, and ECC quantum computing cryptography.