On the strength comparison of ECC and RSA

Masaya Yasuda, Takeshi Shimoyama, Tetsuya Izu, Jun Kogure
(FUJITSU LABORATORIES LTD.)

A part of this research is financially supported by the National Institute of Information and Communications Technology (NICT), Japan.
Outline

- Background and our motivation
- The hardness of the ECDLP
  - ECDLP = Elliptic Curve Discrete Logarithm Problem
- The hardness of the IFP
  - IFP = Integer Factorization Problem
- Comparison of the hardness of the ECDLP and the IFP
- Conclusions and future work
Background and our motivation (1/2)

Table: The comparison of security strengths of ECC and RSA with 80-bit security

<table>
<thead>
<tr>
<th>Report</th>
<th>ECC</th>
<th>RSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIST</td>
<td>160</td>
<td>1024</td>
</tr>
<tr>
<td>Lenstra-Verheul</td>
<td>160</td>
<td>1300</td>
</tr>
<tr>
<td>RSA Labs.</td>
<td>160</td>
<td>760</td>
</tr>
<tr>
<td>NESSIE</td>
<td>160</td>
<td>1536</td>
</tr>
<tr>
<td>IETF</td>
<td>-</td>
<td>1228</td>
</tr>
<tr>
<td>ECRYPT II</td>
<td>160</td>
<td>1248</td>
</tr>
</tbody>
</table>

There are a lot of previous works that analyze the security of ECC and RSA. But the comparison of strengths varies depending on analysis.

Our motivation is once again to compare the security strengths, considering state-of-the-art of theory and experiments.
To compare the strengths of ECC and RSA, we estimate the computing power required to solve the ECDLP and the IFP in a year, respectively.

- The security of ECC and RSA \( \approx \) the hardness of the ECDLP and the IFP, respectively.

In this talk, we focus on the hardness of the ECDLP and the IFP only from the viewpoint of software implementation.

- Using special-purpose hardware for solving the ECDLP or the IFP is a theme of great interest at this conference.
- But these platforms and architectures vary, and it is difficult to make an analysis on the cost performance. \( \Rightarrow \) This is a future work.
**Definition of the ECDLP:**
- Given an elliptic curve $E/GF(q)$ and two points $S$, $T$ of large prime order $n$, find $k$ with $T = kS$.

**Pollard’s rho method:**
- It is the fastest known algorithm for solving the ECDLP.
- First generate $\{X_i\}$ successively by an iteration function $f$. Then find a collision $X_i = X_j$.

**Our policy to estimate the complexity of the rho method:**
- Since the rho method is probabilistic, we estimate the complexity required to solve the ECDLP with 99% probability in order to unify the success probability of solving the IFP.
- Furthermore, we focus on three types in the ECDLP:
  - prime fields, binary fields, and Koblitz curves types.
Suitable iterations for the rho method (1/2)

- **Choice of iterations**
  - The complexity heavily depends on choice of iteration functions.
    - An iteration function \( f \) is the most suitable if \( f \) is a random walk.
  - To analyze the randomness of \( f \), we define
    
    \[
    \delta := \text{the average of } \#\{ \text{iterations with } f \} / \text{Exp.}
    \]
    ('Exp' = the expected number of iterations with random walks)
    
    If \( \delta \) is close to 1, \( f \) has the enough randomness.

- **Prime and binary fields cases:**
  - Teske’s L-adding walk: \( f_{TA}(X) = X + M_j \).
    - \( M_i \): pre-computed points (\( i = 1, \ldots, L \)).
    - \( L \) : partition number, which affects the randomness.
  - From experiments, \( f_{TA} \) with \( L \geq 20 \) has enough randomness on average.

![Graph showing \( \delta \) with \( f_{TA} \) vs partition number L]
Suitable iterations for the rho method (2/2)

**Koblitz curves case:**

- Gallant et al.’s function: \( f_{GLV}([X]) = [g(X)], g(X) = X + \phi^j(X) \)
  - \( \phi \): the Frobenius map, \( j = \text{hash}_m([X]) \)
  - \([X]\): the equivalent class containing \( X \)
- This function is well-defined on \( E/\sim \) and suitable for the speed-up with \( \phi \) and the negation map.
  - \( E/\sim \): the set of the equivalent classes.
- Our experimental result on \( \delta \) with \( f_{GLV} \) on Koblitz curves \( E(GF(2^m)) \):

<table>
<thead>
<tr>
<th>( m )</th>
<th>( 41 )</th>
<th>( 53 )</th>
<th>( 83 )</th>
<th>( 89 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \delta )</td>
<td>1.06</td>
<td>1.10</td>
<td>1.03</td>
<td>1.01</td>
</tr>
</tbody>
</table>

- From experiments, \( f_{GLV} \) has enough randomness on average.
Our method to estimate the complexity

To estimate the complexity, we fix $f_{TA}$ with $L \geq 20$ and $f_{GLV}$, which have enough randomness on average.

- **The complexity of the rho method is determined by**
  
  ① the number of iterations, and
  
  We estimate the number of iterations required to solve the ECDLP with 99% probability, by considering the distribution of the frequencies.

  ② the processing performance of iterations.

  We estimate the processing performance of iterations by using the previously known best records.
① The number of iterations

- Distribution of the frequencies of the number of iterations:
  - We solved the ECDLP over prime and binary fields of 40-bit for 10,000 times. We also used $f_{TA}$ with $L = 20$.
    - We did not use the speed-up with the negation map.
  - As the numbers of iterations can be modeled as waiting times, it is reasonable to approximate the graph by $\Gamma$-distribution.
  - With this approximation, we estimate that we can solve the ECDLP with 99% probability if we compute iterations by 3 times of the expected number.

![Diagram showing distribution of frequencies and approximation curve]

※The Horizontal axis shows the number of iterations (set 1 as the expected number)
The processing performance of iterations

- We use the latest data:
  - Prime fields: 306.08 cycles / iteration (112-bit) [1]
  - Binary fields: 1047 cycles / iteration (131-bit) [2]
  - Koblitz curves: $1.62 \times 1047$ cycles / iteration (131-bit)
    - $1.62$ : overhead of transforming bases estimated by our implementation.

- $t(f)$: the processing performance on an elliptic curve of $N$-bit.
- *We assume that $t(f)$ scales proportionally to $N^{1.585}$, due to the Karatsuba method.*

$$
t(f) = \begin{cases} 
306.08 \times (N/112)^{1.585} & \text{(prime fields case),} \\
1047 \times (N/131)^{1.585} & \text{(binary fields case),} \\
1.62 \times 1047 \times (N/131)^{1.585} & \text{(Koblitz curves case).} 
\end{cases}
$$

Estimation formula

The complexity of the rho method = \#\{iterations\} \times t(f).

- **Estimation formula of the complexity of the rho method:**
  - \( T = \) the computing power to solve the ECDLP of N-bit with 99% probability
    - FLOPS is the unit of \( T \).
    - Consider \( 2^N \) as the order of the point \( S \). Set \( Y = 365 \cdot 24 \cdot 60 \cdot 60 \) (seconds)
  
  \[
  T = \begin{cases} 
    3 \cdot \sqrt{\pi} 2^N / 2 \times 306.08 \times (N / 112)^{1.585} / Y \ (prime \ fields \ case), \\
    3 \cdot \sqrt{\pi} 2^N / 2 \times 1047 \times (N / 131)^{1.585} / Y \ (binary \ fields \ case), \\
    3 \cdot \sqrt{\pi} 2^N / N / 2 \times 1.62 \times 1047 \times (N / 131)^{1.585} / Y \ (Koblitz \ curves \ case). 
  \end{cases}
  \]

  The number of iterations with 99% success probability. Assume the efficiency of the speed-up with automorphisms.
  The processing performance of iterations. Assume that \( t(f) \) scales proportionally to \( N^{1.585} \).
Computing power to solve the ECDLP

By our estimation formula, we estimate the computing power required to solve the ECDLP of N-bit in a year as follows:

- The vertical axis shows $\log_{10}(T)$.

For example, we have $T \approx 10^{19}$ FLOPS in case $N = 160$-bit.
[IFP] GNFS

GNFS (= general number field sieve)
- It is the most efficient known algorithm for solving the IFP of large composite integers.
- Heuristically, the complexity of the GNFS for factoring $N$ is given by

$$L_N \left( \frac{1}{3}, \frac{3\sqrt{64}}{9} \right) \text{ as } N \to \infty$$

where

$$L_N(s, c) = \exp((c + o(1))(\log N)^s (\log \log N)^{1-s}).$$

Our method to estimate the complexity of the GNFS:

1. We analyze the $o(1)$-value of $L_N(s, c)$ experimentally.
2. By using the previously known result, we set
   - the $o(1)$-value of $L_N(s, c)$, and
   - the leading coefficient $\ell$ of $L_N(s, c)$. 
Analysis of the o(1)-value of $L_N(s, c)$

The o(1)-value of $L_N(s, c)$ is closely related to the o(1)-value of $\Phi(x, z)$.

For two integers $x$ and $z$, let $\Phi(x, z)$ denote the probability that an arbitrary integer in $[1, x]$ is $z$-smooth.

It is known that we have $\Phi(x, z) = (\log_z x)^{(-1+o(1)) \log x}$ as $x \to \infty$.

Experimental investigation:

We expect that the o(1)-value of $\Phi(x, z)$ is included in $[0, 0.2]$ as $x \to \infty$.

Since $\Phi(x, z)$ is used twice for deriving $L_N(s, c)$, we expect that the o(1)-value of $L_N(s, c)$ is included in $[0, 0.4]$ as $x \to \infty$. 

![Experimental result on the o(1)-value of $\Phi(x, z)$ with $x \leq 32$-bit.](image)
② Setting of \( o(1) \) and \( \ell \)

- Previously known results:
  - The result in [3] gives the computing power required to solve the IFP of N-bit in a year as follows:
    - Assume that the memory requirement is 2G Bytes.
    - \( T = \text{the computing power (FLOPS is the unit of } T) \)

<table>
<thead>
<tr>
<th>( N )</th>
<th>768</th>
<th>1024</th>
<th>1536</th>
<th>2048</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \log_{10} T )</td>
<td>12.6879</td>
<td>16.3395</td>
<td>22.2319</td>
<td>27.0413</td>
</tr>
</tbody>
</table>

- \( o(1) \) and \( \ell \):
  - The complexity of the GNFS is very close to each data of the above result if we set
    - \( o(1) = 0.348172 \), and (Note that \( o(1) \) is in \([0, 0.4]\))
    - \( \ell = 1.0373 \times 10^7 \) as the leading coefficient of \( L_N(s, c) \).
  - Estimation formula:
    \[
    T = \ell \cdot L_N \left( \frac{1}{3}, \frac{3}{\sqrt[3]{9}} \right) \quad \text{with } o(1) \text{ and } \ell \text{ as above}
    \]

By our estimation formula, we estimate the computing power required to solve the IFP of N-bit in a year as follows:

- The vertical axis shows $\log_{10}(T)$. 

![Graph showing the estimated computing power required to solve the IFP for different bit sizes. The graph plots the bit size of the IFP on the x-axis and $\log_{10}(T)$ on the y-axis. The data points are marked for bit sizes of 384, 768, 1024, 1536, 2048, and 3072. The GNFS CRYPTREC report 2006 is referenced.]

Copyright 2012 FUJITSU LABORATORIES LTD.
By comparing the computing power required to solve the ECDLP and the IFP, we estimate the bit sizes that provide the same level security:

<table>
<thead>
<tr>
<th>Bit size of the IFP</th>
<th>Bit size of the ECDLP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prime fields</td>
</tr>
<tr>
<td>512</td>
<td>86</td>
</tr>
<tr>
<td>696</td>
<td>107</td>
</tr>
<tr>
<td>768</td>
<td>115</td>
</tr>
<tr>
<td>851</td>
<td>123</td>
</tr>
<tr>
<td>1024</td>
<td>138</td>
</tr>
<tr>
<td>1219</td>
<td>154</td>
</tr>
<tr>
<td>1536</td>
<td>176</td>
</tr>
<tr>
<td>2048</td>
<td>207</td>
</tr>
<tr>
<td>2839</td>
<td>247</td>
</tr>
</tbody>
</table>

※If we use ordinary multiplication for elliptic operations, the ECDLP bit length can be reduced at most two bits.
In particular, we have

- **768-bit IFP ≅ 115-bit ECDLP over prime field, 112-bit over binary field, or 117-bit on Koblitz curves.**
  - The world records of 2011 for solving RSA and ECC are 768-bit RSA in 2010 [4] and 112-bit ECC over prime field in 2009 [5], respectively.
  - Since the times of these two records are close, we consider that these records indicate reasonability of our estimation.

- **1024-bit IFP ≅ 138-bit ECDLP over prime field, 136-bit over binary field, or 141-bit on Koblitz curves.**
  - Although it is often said that 1024-bit RSA corresponds to 160-bit ECC, our estimation indicates that shorter ECC key sizes provide the same level of security.

---

Conclusions and future work

**Conclusions:**

- **ECDLP (the rho method):**
  - By considering the distribution of the frequencies, we estimate \( \#\{\text{iterations}\} \) with 99% success probability.
  - We also estimate \( t(f) \) by using the previously known best records.

- **IFP (the GNFS):**
  - To estimate the complexity of the GNFS, we determine \( o(1) \) and \( \ell \) of \( L_N(s, c) \) by using our analysis of \( \Phi(x, z) \) and CRYPTREC result [3].
  - Our estimation showed that 1024-bit IFP \( \equiv 140\)-bit ECDLP.
    - If we say 160-bit ECC has 80-bit security, our estimation indicates that 1024-bit RSA does not reach this security.

**Future work:**

- We need to analyze the complexity of the rho method with the iteration function proposed by [6], which seems to be the best known function on Koblitz curves.

- Furthermore, we need to check if the processing performance of iterations scales proportionally to \( N^{1.585} \).

shaping tomorrow with you