International Journal of Networking and Computing – www.ijnc.org
ISSN 2185-2839 (print) ISSN 2185-2847 (online)
Volume 6, Number 2, pages 181–194, July 2016

Web-based Volunteer Computing for Solving the Elliptic Curve Discrete Logarithm Problem

Shoma Kajitani
Okayama University
Okayama, Japan

Yasuyuki Nogami
Okayama University
Okayama, Japan

Shunsuke Miyoshi
Okayama University
Okayama, Japan

Thomas Austin
San Jose University
San Jose, California, United States of America

Khandaker Md. Al-Amin
Okayama University
Okayama, Japan

Nasima Begum
Okayama University
Okayama, Japan

Sylvain Duquesne
University of Rennes 1, Campus Beaulieu
Rennes, France

Received: February 15, 2016
Revised: April 21, 2016
Accepted: July 4, 2016
Communicated by Toru Nakanishi

Abstract

Elliptic curve discrete logarithm problem (ECDLP) is the basis of security of elliptic curve cryptography (ECC). The security evaluation of ECC has been studied by solving an ECDLP. We need a large amount of computational resources for the evaluation. This paper proposes a new system collecting computational resources with Web-based volunteer computing (Web-based VC). In the system, web applications are allocated to volunteer participants (workers) as jobs. Web applications are built by utilizing technologies called Native Client (NaCl) and Portable NaCl (PNaCl). This paper evaluates the performance of the system and solves 70-bit ECDLP with Web-based VC. The performance of the web application utilizing NaCl is approximately 6.4 times higher than that of the web application written in JavaScript. Also, the performance of the web application utilizing PNaCl is approximately 4.2 times higher. In the case of NaCl, 70-bit ECDLP is solved in only 1389 seconds. If we collect 100,000 PCs by Web-based VC, 114-bit ECDLP will be solved in only approximately 1 day.

Keywords: Elliptic curve cryptography, Web-based VC, Native Client

1 Introduction

Elliptic curve cryptography (ECC) is one of the public key cryptographies. Elliptic curve discrete logarithm problem (ECDLP) is the basis of security of ECC. The strength of the security has been evaluated by solving a reasonable size of ECDLP. In order to solve ECDLP, we need a large amount of computational resources. For example, in 2009, 112-bit ECDLP was solved by 215 PlayStation 3 within around half a year [1]. It is difficult to collect more computational resources because the cost is too expensive.

This paper proposes a new system collecting a large amount of resources with volunteer computing (VC). VC is an internet-based parallel computing system. VC gathers volunteer participants (workers) through the Internet and uses the idle computational resources of the worker's personal computers (PC). Thus, VC realizes a high performance computing system at low cost. In the conventional VC system with BOINC [2], workers need some steps to participate in a VC project, e.g. installing dedicated client software to their computers and registering their e-mail addresses. These steps are troublesome for workers.

Web-based VC system is a new approach for VC system. In Web-based VC system, workers participate in a VC project by accessing a specific web page. In Web-based VC system, it is expected to increase the number of workers, because it is easy for workers to participate in a VC project compared with the conventional VC system. However, the computational performance of each worker is relatively low because workers execute a web application on a web browser. In order to improve the performance, this paper developed such web applications by utilizing technologies called Native Client (NaCl) and Portable NaCl (PNaCl) [3].

This paper implements a computing system with Web-based VC for solving ECDLP. First, this paper evaluates the performance of the web application utilizing NaCl and PNaCl. The performance of the web application utilizing NaCl is approximately 6.4 times higher than that of the web applicaton written in JavaScript. Also, the performance of the web application utilizing PNaCl is approximately 4.2 times higher than that of the web application written in JavaScript. Then, 70-bit ECDLP is solved with Web-based VC. In the case of NaCl, 70-bit ECDLP is solved in 1389 seconds.

Moreover, we estimated the time to solve 114-bit ECDLP from these exeperimental results. If we collect 100,000 PCs by Web-based VC, 114-bit ECDLP will be solved in only approximately 1 day.

2 Preliminaries

This section introduces ECDLP. The difficulty of solving ECDLP guarantees the security of elliptic curve cryptography (ECC). Then, this section introduces Pollard's rho method.

2.1 ECDLP

Let $\mathbb{F}_p$ be a prime field. An elliptic curve E is defined over $\mathbb{F}_p$ by the equation as follows:

$$E : y^2 = x^3 + ax + b, \quad \text{with } a, b \in \mathbb{F}_p, \tag{1}$$

$E(\mathbb{F}_p)$ is the set of rational points on the curve E defined over $\mathbb{F}_p$. Let $\#E(\mathbb{F}_p)$ be the order of rational points including the infinity $\mathcal{O}$. Then, let r be the order of a cyclic group in $E(\mathbb{F}_p)$. For simplicity, this paper deal with the case of $r = \#E(\mathbb{F}_p)$. The relation between P and Q is shown in Eq.(2), where P and Q are rational points on the curve, and s is a scalar such that.

$$Q = [s]P. \tag{2}$$

From Eq.(2), we can say that it is easy to obtain Q from P and s. When the number of rational points is huge, then its inverse problem becomes too difficult to obtain s from P and Q. This problem is called elliptic curve discrete logarithm problem (ECDLP), and it is the basis of the security of ECC.

2.2 Pollard's Rho Method

Pollard's rho method [4] is well known as an efficient technique for solving ECDLP. This method mainly consists of 2 parts as follows.

1. Generating a lot of random rational points,
2. Detecting a collision among the generated points.

Rational points T_i are iteratively obtained by calculating Eq.(3) with random numbers a_i, b_i, where $0 < a_i, b_i < r$.

$$T_i = [a_i]P + [b_i]Q. \tag{3}$$

When a collision is detected such as $T_i = T_j (i \neq j)$, ECDLP is solved. In other words, the scalar value s of Eq.(2) is obtained.

The original rho method is shown in Algorithm. 1. In this paper, we have generated a table which consists of n rational points for generating random rational points. n is an arbitrary positive integer and it is the size of the table. Let W_i be a rational point in the table. The rational points T_i is generated from T_{i-1} and W_l , where l is an index of the table. The function $\eta(T_{i-1})$ returns the index l corresponding to T_{i-1}. According to the birthday paradox, a collision occurs with a probability of 50% when $\sqrt{\pi r/12}$ rational points have been generated. Thus, the original rho method averagely needs to store $\sqrt{\pi r/12}$ rational points in order to detect a collision. The number of stored points can be reduced by the distinguished point method [5]. Let θ be a parameter for the distinguished point method. The number of stored points is reduced by $1/\theta$ because the method stores only the distinguished points. When θ is too large, any collision doe not occur among the stored data. It is sometimes said that θ should be less than 25% of the ECDLP size.

In our previous work, an improvement of rho method has been proposed [6]. For solving ECDLP efficiently, this paper focuses on how to apply Web-based VC for solving ECDLP together with the improvement.

Algorithm 1: Pollard's Rho Method

```
Input: $P$, $Q(= [s]P) \in E(\mathbb{F}_p)$ $(0 \leq s < r)$
Output: $s$
1  $n$ is an arbitary positive integer,
2  for $i = 0$ to $n - 1$ do
3      $a_i$, $b_i$ are random elements $(0 \leq a_i, b_i < r)$,
4      $W_i \leftarrow [a_i]P + [b_i]Q$.
5  $a_0$, $b_0$ are random elements $(0 \leq a_0, b_0 < r)$,
6  $T_0 \leftarrow [a_0]P + [b_0]Q$.
7  $i \leftarrow 1$
8  while $T_i \neq T_j (0 \leq j < i)$ do
9      $l \leftarrow \eta(T_{i-1})$,
10     $a_i \leftarrow a_{i-1} + a_l$, $b_i \leftarrow b_{i-1} + b_l$, $T_i \leftarrow T_{i-1} + W_l$,
11     $i \leftarrow i + 1$.
12 $s \leftarrow -(a_i - a_j)(b_i - b_j)^{-1} \pmod r$.
```

3 Previous Work

This section explains the previous work of some of the authors of this paper [6], [7].

3.1 Summary of the Computing System

The previous work solved ECDLP by a parallel computing system. The summary of the system is shown in Fig. 1. The system is composed of servers and many clients.

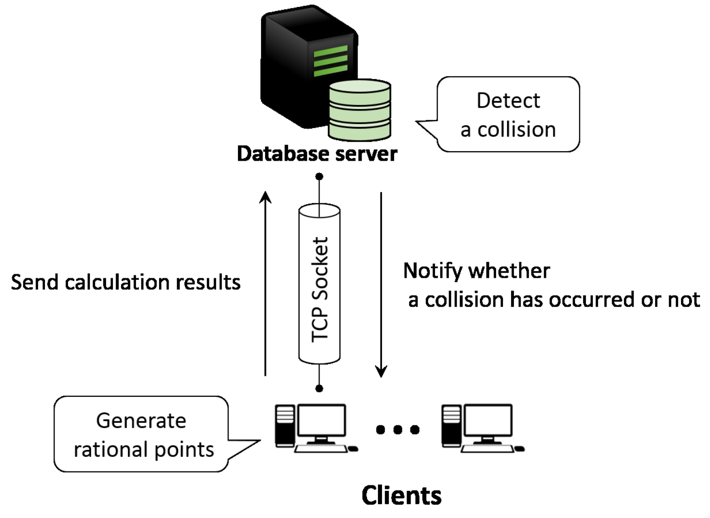


Figure 1: Summary of the computing system

Pollard's rho method is used in the previous work for solving ECDLP. This method consists of 2 parts as described in Sec. 2.2. The first part generates many random rational points. The second part detects a collision among the generated points. In the computing system, the clients generate random rational points and the servers detect a collision.

At first, the clients generate many random rational points. Then the clients execute an application generating rational points. This application is developed using C++ language with g++ compiler. When the client generates the specified number of rational points, the client sends coordinates of the generated points to the server. The server and a client communicate by using a TCP socket. The clients continue to generate rational points until a collision is occurred.

In the 2nd step of rho method, the server detects a collision. The server receives coordinates of the generated points and stores them in a MySQL database. When the collision is occurred, the server notifies the clients.

3.2 Experimental Results of Previous Work

In the previous work [7], 94-bit ECDLP was solved. This work used 2 servers and 69 clients in Okayama university. The machines of these clients were used for about 2 days and 94-bit ECDLP was solved. The detailed comparison with the proposed system is discussed in Sec. 6.4.

4 Volunteer Computing

This paper proposes a new computing system for solving ECDLP. The proposed system uses volunteer computing (VC) to collect a large amount of calculation resources. This section introduces VC systems, specially Web-based VC system.

4.1 Summary of VC Systems

Volunteer computing (VC) is a large-scale Internet-based parallel computing system. VC collects many volunteer participants (workers) through the Internet and uses their calculation resources. VC has realized a high performance system such as a supercomputer at SETI@HOME [8] project. The greatest advantage of VC is that a high performance system is realized at a very low cost.

In VC system, jobs are allocated from a server to workers through the Internet. Workers calculate it and return a calculation result to the server.

4.2 Conventional VC System

In the conventional VC system with the software BOINC, workers need some procedure to participate in the VC system which are as follows:

1. Downloading BOINC client software.
2. Installing the client software and rebooting PC.
3. Starting the client software.
4. Choosing VC projects to participate.
5. Registering worker information by e-mail address.

These procedures are sometimes troublesome for workers. In the conventional VC system, the job is made for each computing platform. Although developing for each platform needs labor or cost, but the job is optimized.

4.3 Web-based VC System

As a new approach for VC system, Web-based VC system [9] has been proposed. In Web-based VC system, workers use a Web browser, and participate in the VC project by the following procedure.

1. Starting a web browser.
2. Accessing a specific web page.

In Web-based VC system, all the worker needs to access a specific web page from a web browser. This is very easy for workers. Therefore, Web-based system is expected to collect much more workers compared to the conventional VC system. In this paper, Web-based VC is used for solving ECDLP.

However, the computational performance of each worker is actually lower than the conventional system. In the conventional VC system, the job is optimized for a certain platform. On the other hand, in Web-based VC system, the job is not optimized because it is executed on a web browser as a web application. Therefore, the computational performance of each worker becomes lower than the conventional system.

5 Proposed Computing System

This paper proposes a computing system using Web-based VC for solving ECDLP. This section shows an implementation of the proposed system. First, the proposed system is briefly introduced in Sec. 5.1. Second, Sec. 5.2 shows how to build web applications. Then, the behavior of a web server is explained in Sec. 5.3. Next, a web page of our system is shown in Sec. 5.4 Then, the parallelization method and the reduction of the data sent to the server is explained in Sec. 5.5.

5.1 Summary of the Proposed System

This section shows a summary of the proposed system. (*see* Fig. 2)

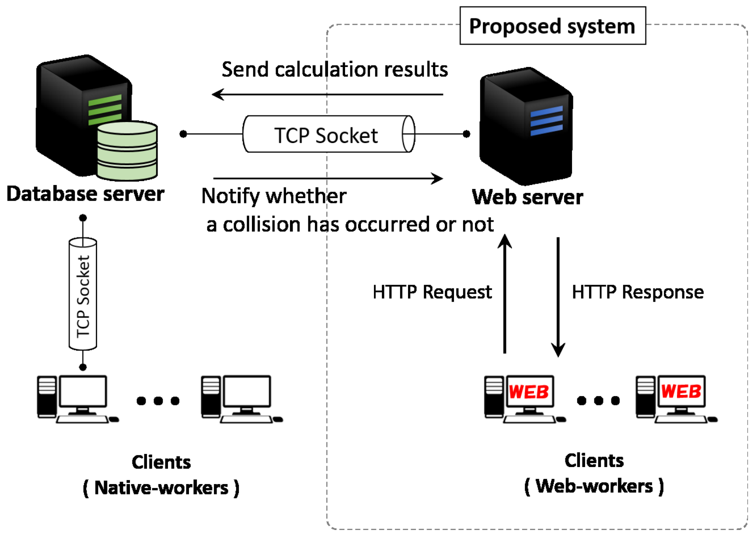


Figure 2: Summary of the proposed system

The most typical feature of the proposed system is that the clients of the proposed system use a web browser. The client of the proposed system (hereinafter, referred to as “web-worker”) can join the system only by accessing a website for solving ECDLP. When a web-worker accesses the website, a web server sends the web application to the web-worker. The web-worker executes the web application and generates many rational points. After that, the coordinates of the generated rational points are sent to the web server. The web server and the database server in the previous work communicate by using a TCP socket. When the web server receives a calculation result from a web-worker, the web server also sends it to the database server.

5.2 Web Application

5.2.1 Web application and its problem

A Web application generating random rational points needs to be built for web-workers. In the previous work, the client (hereinafter, referred to as “native-worker”) executes a native application. A native application is an application built for a certain platform. Specifically, the native application was built by compiling C++ codes with g++ compiler in the previous work. On the other hand, web-workers execute the web application on a web browser. Therefore, the web application needs to be built for web-workers.

In this paper, web applications are built by utilizing Native Client (NaCl) and Portable Native Client (PNaCl). In general, the native application works more efficiently than the web application because the native application is built for a certain platform. In order to improve the performance of the web application, we use technologies called NaCl and PNaCl in this paper.

5.2.2 Native Client and Portable Native Client

Native Client (NaCl) is a sandboxing technology for running the compiled C and C++ code on the web browser efficiently and securely. NaCl allows safely running the compiled code on a web browser as a web application and it runs at near native speeds. The C and C++ code are compiled with NaCl toolchain contained in the dedicated SDK (NaCl SDK). NaCl toolchain is based on GCC. The compiled code is independent of user's operating system (OS).

Portable Native Client (PNaCl) extends NaCl with architecture independence. A C and C++ code is compiled with PNaCl toolchain contained in NaCl SDK. PNaCl toolchain is based on Clang which is a compiler for C and C++ languages. The compiled code can be executed on a web browser without depending on user's OS and architecture. Therefore, the web application runs as a 32-bit application even if the worker's PC is 64-bit OS.

These technologies are used in Folding@home [10] which is a famous VC project. Folding@home uses not only the conventional VC system but also Web-based VC system utilizing PNaCl [11].

5.2.3 Developing a web application

In this paper, the web application generating rational points which is made by compiling C++ codes with NaCl SDK. The application performs about 100-bit integer arithmetic. In order to perform such arithmetic efficiently, 128-bit integer was used in the previous work. 128-bit integer is a kind of GCC extension. However, the C++ code using 128-bit integer can not be compiled with PNaCl toolchain because PNaCl toolchain is not based on GCC. Moreover, 128-bit integer is available only on 64-bit architecture. In other words, the application for 32-bit architecture needs to be built without 128-bit integer.

Therefore, two types of applications are developed in this paper. One application is developed by using 128-bit integer. This application is made by compiling with NaCl toolchain. It runs only on 64-bit architecture. The other application is developed using GMP (GNU Multiple-Precision) library [12]. GMP is a free library for multi-precision arithmetic. This application is built by compiling with PNaCl toolchain. Although the calculation efficiency is lower than the application using 128-bit integer, the application using GMP does not depends on architectures.

5.2.4 Security and Pepper API

Generally, applications embedded in a web page are executed automatically while opening the web page. Thus, there are some risks such as viral infections. In the proposed system, the web-worker executed the web application safely because of NaCl and PNaCl. NaCl and PNaCl are sandboxing technologies for running the web application in a sandbox. The communication with other processes is restricted in a sandbox. Thus, the web application cannot operate other applications or some important data even if it contains virus codes.

NaCl and PNaCl module communicate with other processes by using Pepper API. In the proposed system, NaCl and PNaCl module communicate with JavaScript. The summary of communications between NaCl module and JavaScript is shown in Fig. 3. JavaScript sends the web application to NaCl module by using Pepper API after receiving it from the web server. The web application runs in the NaCl module and rational points are generated. Then, NaCl module sends the coordinates of generated rational points to JavaScript by using Pepper API.

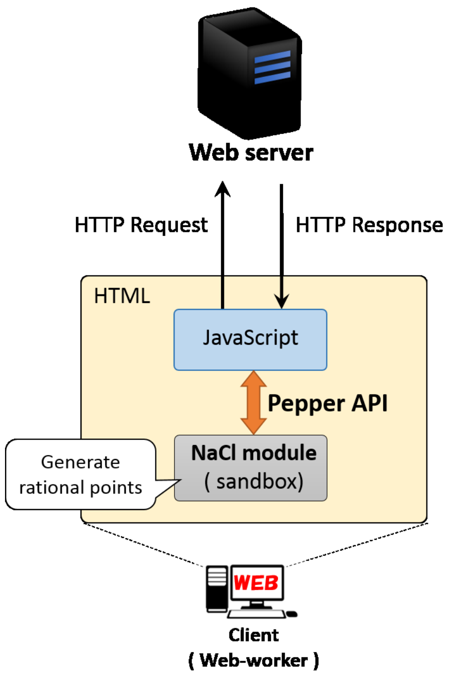


Figure 3: Communication between NaCl module and JavaScript

5.3 Web Server

The web-worker sends a request message to the web server on accessing the website. Then, the web server replies to it with the web application including HTML files and JavaScript files. JavaScript files are used to communicate with the web server, NaCl module, and PNaCl module as described in Sec. 5.2.4.

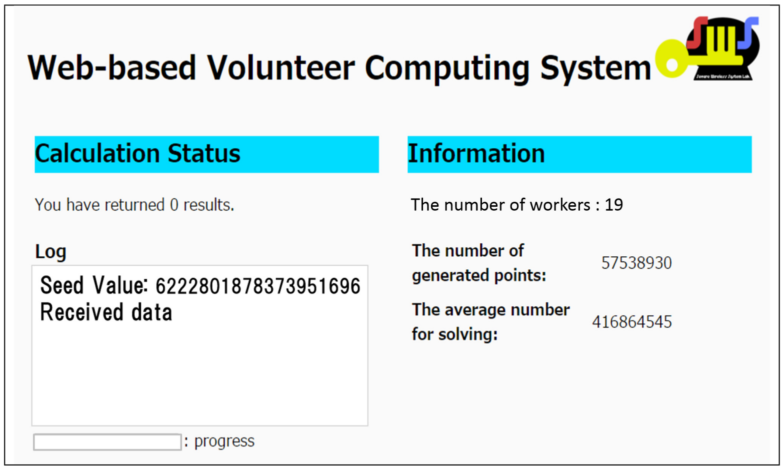
The web server also sends a seed value to the web-worker. The seed value is used to generate random numbers, and random numbers are used to generate random rational points. If some different web-worker receives the same seed value, the same random rational points are generated. In order to avoid this problem, the seed value depends on the access date and web-worker's IP address in the proposed system.

Besides, the web server sends calculation results to the database server on receiving them from web-workers. The coordinates of generated rational points are sent from web-workers to the web server as a calculation result when the web-workers generate the specified number of rational points. Then, the web server sends calculation results to a database server by using a TCP socket. After that, the database server notifies the web server whether a collision is occurred or not. If a collision has been occurred, the web server also notifies the web-workers.

5.4 Web Page

The web page shown in Fig. 4 is opened after the web-worker receives the web application and so on. Also, the web application automatically runs on opening the web page. When the specified number of rational points is generated, the web-worker sends them to the web server as a calculation result. This is because if a rational point is sent whenever it is generated, the server communicate with web-workers far too frequently. In this work, the web-worker returns the calculation result once about half an hour. The web-worker continues to generate rational points while the page is opened.

The web page displays some values as follows:


Web-based Volunteer Computing System

Calculation Status

You have returned 0 results.

Log

```
Seed Value: 6222801878373951696
Received data
```

: progress

Information

The number of workers : 19

The number of generated points:	57538930
The average number for solving:	416864545

Figure 4: Web page of the proposed system

- Number of returned calculation results
- Seed value
- Progress of a calculation
- Number of web-workers
- Number of generated rational points (stored in the database)
- Average number of rational points for solving ECDLP

The number of web-workers and generated rational points are sent from the web server. The number of web-workers is updated when a web-worker accesses or leaves the web page. The number of generated rational points is updated on returning a calculation result.

5.5 Parallelization method and reduction of the data sent to the server

When the web page for joining the ECDLP attack is opened, the web-worker downloads the attack program on the web browser and then begins to generate random rational points. Then, the web-worker sends the data to the server as a specified number of rational points has been generated. Since a lot of web-workers join and then send the data, it is not practical for the server to receive and store the huge amount of data. In addition, the server needs to find a collision from the huge data. In order to reduce the number of sent and stored rational points, the distinguished point method is efficiently utilized as described in Sec. 2.2.

As an example, let us consider the case that $\theta = 2^{10}$. The web-worker iteratively generates rational points. Among them, the web-worker distinguishes and stores rational points such that its x–coordinate is divisible by $\theta = 2^{10}$ only[1]. When the number of rational points stored at the web-worker has reached to the specified number, the web-worker sends them to the server. Let N be the number of generated rational points, N/θ rational points on average is sent to the server as shown in Fig. 5. Thus, the parameter θ is able to efficiently reduce the data size sent to and stored in the server. However, the size of θ should not be too large as described in Sec. 2.2

[1]In this case, it is easily distinguished because the binary representation of the x–coordinate has 10 zeros from LSB such as $(1010\cdots110000000000)_2$.

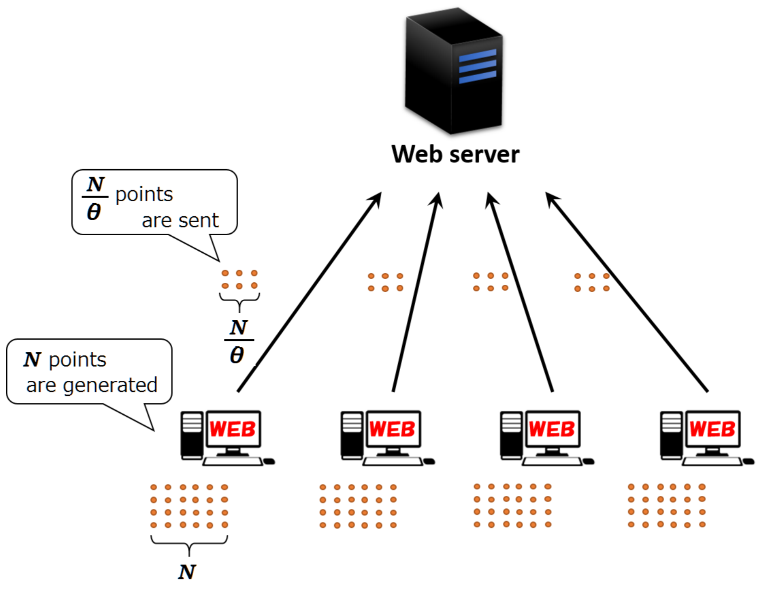


Figure 5: Reduction of the data sent to the server with distinguished point method

6 Experiments

This paper evaluated the proposed system by solving 70-bit ECDLP. First, the performance of NaCl and PNaCl were evaluated by comparing the computational performance of web applications with that of native applications. Then, 70-bit ECDLP was solved with the proposed system. In the experiment, the web server allocated web applications for solving 70-bit ECDLP as jobs to web-workers.

6.1 Performance Evaluation of NaCl

Table 1: Worker's PC environment for NaCl comparison

OS	Windows 7 Professional 64bit
CPU	Intel(R) Core i7-870 2.93GHz
Memory	8GB
Web browser	Google chrome ver.43.0.2357.130

Table 2: Performance evaluation result of NaCl

Type of applications	web application		native application
	NaCl	JavaScript	
ECDLP size	70-bit		
The number of tabs	1		
The number of generated points	32,768,000		
Computation time [sec]	46.21	294.98	12.92

This experiment evaluated the computational performance of NaCl. The solving 70-bit ECDLP

Table 3: Worker's PC environment for PNaCl comparison

OS	Ubuntu 14.04 LTS 32bit
CPU	Intel(R) Core i7-870 2.93GHz
Memory	8GB
Web browser	Google chrome ver.44.0.2403.125

Table 4: Performance evaluation result of PNaCl

Type of applications	web application		native application
	PNaCl	JavaScript	
ECDLP size	70-bit		
The number of tabs	1		
The number of generated points	32,768,000		
Computation time [sec]	88.08	368.78	33.56

application using 128-bit integer was executed on NaCl as a web application and on a terminal as a native application. Also, the web application written in JavaScript was executed. The native application is a executable file which is compiled C++ codes with g++ compiler. This experiment measured the computation time of the web applications and the native application. Then, the computational performance was evaluated by comparing them. In this experiment, the application generated the specified number of rational points for easily comparing. The experimental environment is shown in Table 1 and the results are shown in Table 2.

According to the experimental results, the web application on NaCl required 46.21 seconds to generate 32,768,000 rational points. In the case of the web application written in JavaScript, 294.98 seconds is required to generate them. On the other hand, the native application required 12.92 seconds to generate them. It confirms that the performance of NaCl is approximately 6.4 times higher than that of JavaScript, though it is approximately 3.6 times lower than the performance of the native application.

6.2 Performance Evaluation of PNaCl

This experiment evaluated the computational performance of PNaCl. The solving 70-bit ECDLP application using GMP library was executed on PNaCl as a web application and on a terminal as a native application. Also, the web application written in JavaScript was executed. Then, their computation time were measured. The experimental environment is shown in Table 3 and the result is shown in Table 4. In this experiment, the applications were executed on a 32-bit OS PC because the web application runs on PNaCl as a 32-bit application even if worker's PC is 64-bit OS.

According to the experimental results, the web application on PNaCl required 88.08 seconds to generate 32,768,000 rational points. In the case of the web application written in JavaScript, 368.78 seconds is required to generate them. On the other hand, the native application required 33.56 seconds to generate them. It confirms that the performance of PNaCl is approximately 4.2 times higher than that of JavaScript, though it is approximately 2.6 times lower than the performance of the native application.

6.3 Solving 70-bit ECDLP

This experiment solved 70-bit ECDLP with proposed system and compared NaCl with PNaCl. In this experiment, the web server has the database for detecting a collision because the scalability of the experiment is small. The server PC environment is shown in Table 5, and a worker's PC environment is shown in Table 1. The experimental results are shown in Table 6.

Table 5: Server PC environment

OS	Ubuntu 14.04 LTS 64bit
CPU	Intel(R) Core i7-3770 3.40GHz
Memory	4GB
Web server	Apache ver. 2.4.7
Database	MySQL ver. 5.5.41

Table 6: Experimental results of solving 70-bit ECDLP

Type of application	NaCl	PNaCl
ECDLP size	70-bit	
The number of tabs	8	
Average number for a collision	13,185,766,025	
θ for distinguished point	2^{15}	
The number of generated points	9,238,478,848	10,293,411,840
The number of stored points	281,936	314,130
Time of solving ECDLP	3086 sec	6583 sec
The number of genearated points per second	$\approx$ 2,993,674	$\approx$ 1,563,635

The average number of generated rational points for detecting one collision was $\sqrt{\pi r/12} \approx 13185766025$, where $r = 664113186492198813709$ (70-bit). The parameter θ for the distinguished point was 2^{15}. If θ is too large, any collision does not occur among the stored data. Thus, 70-bit ECDLP was averagely solved when $\sqrt{\pi r/12}/2^{15}$ rational points are stored in the database.

In the case of NaCl, 70-bit ECDLP was solved in 3086 seconds with a PC. On the other hand, in the case of PNaCl, 70-bit ECDLP was solved in 6583 seconds. The number of generating points per second is shown in Table 6. According to this results, it is shown that the computational performance of NaCl is approximately 2 times higher than that of PNaCl.

The following point is given as a reason for the experimental results. As described in Sec. 5.2.2, an application runs on PNaCl as a 32-bit application even if worker's PC is 64-bit OS. On the other hand, an application runs on NaCl as a 64-bit application. In this experiment, solving ECDLP applications are executed on a 64-bit OS PC, thus the computational performance of NaCl is higher than that of PNaCl.

6.4 Comparison with Previous Work

In this section, the proposed system particularly with NaCl is compared with the previous system which is described in Sec. 3. The comparison is shown in Table 7. In the case of the proposed system with NaCl, approximately 3 million rational points are generated per second. In the case of the previous system, approximately 24 million rational points are generated per second. According to this results, the performance of the proposed system is approximately 8 times lower than that of previous system. However, in Sec. 6.1, it is shown that the performance of the web application on NaCl is approximately 3.6 times lower than that of the native application. We assumed that the data transmission processing of JavaScript made the difference between these results.

Next, the parameter θ for distinguished point method is set to 2^{20}. The data transmission frequency is more reduced by increasing the parameter θ. Table 8 shows the experimental results of $\theta = 2^{20}$. According to this results, the performance of the proposed system has been improved. It is approximately 3.7 times lower than that of previous system. Therefore, it is observed that the data transmission makes a large overhead in the proposed system when θ is not enough large.

Table 7: Comparison with the previous system ($\theta = 2^{15}$)

Type of application	NaCl	Native
ECDLP size	70-bit	
The number of tabs (threads)	8	
Average number for a collision	13,185,766,025	
θ for distinguished point	2^{15}	
The number of generated points	9,238,478,848	14,653,456,384
The number of stored points	281,936	447,188
Time of solving ECDLP	3086 sec	601 sec
The number of genearated points per second	$\approx$ 2,993,674	$\approx$ 24,381,791

Table 8: Comparison with the previous system ($\theta = 2^{20}$)

Type of application	NaCl	Native
ECDLP size	70-bit	
The number of tabs(threads)	8	
Average number for a collision	13,185,766,025	
θ for distinguished point	2^{20}	
The number of generated points	9,762,242,560	14,587,789,312
The number of stored points	9,310	13,912
Time of solving ECDLP	1389	563 sec sec
The number of genearated points per second	$\approx$ 7,028,252	$\approx$ 25,910,815

6.5 Estimation about Solving 114-bit ECDLP

We estimated the time for solving 114-bit ECDLP with web-based VC. This estimation is shown in Table 9.

Pollard's rho method averagely needs to generate $\sqrt{\pi r/12}$ random rational points in order to occur a collision as described in Sec. 2.2. In the case of 114-bit ECDLP, the rho method averagely needs to generate $\sqrt{\pi \times 2^{114}/12}$ ($\approx 7.37384604894 \times 10^{16}$) random rational points. All of them don't need to be stored in the database because rho method can reduce the number of stored points by the distinguished point method. According to the experimental result of solving 70-bit ECDLP, 7,028,252 rational points are generated per second with a PC. Therefore, it will take approximately 333 years to solve 114-bit ECDLP with a PC.

Existing VC projects collect many workers. For example, PrimeGrid [13] which is a famous VC project has about 100,000 active workers. If we collect 100,000 PCs, 114-bit ECDLP will be solved in only approximately 1 day.

Table 9: Estimation about solving 114-bit ECDLP

Type of application	NaCl	
ECDLP size	114-bit	
θ for distinguished point	2^{27}	
The number of PCs	1	100,000
Estimation time for solving	333 years	1.21 days

7 Conclusion

This paper solved 70-bit ECDLP with Web-based VC and evaluated the computational performance of NaCl and PNaCl. Solving ECDLP applications were executed on NaCl and PNaCl as web applications.

According to the experimental results, the performance of NaCl is approximately 6.4 times higher than that of JavaScript and the performance of PNaCl is approximately 4.2 times higher than that of JavaScript. Then, 70-bit ECDLP was solved in 1389 seconds with a PC when the application was executed on NaCl. If 100,000 PCs are collected by Web-based VC, 114-bit ECDLP will be solved in only approximately 1 day.

References

[1] Joppe W. Bos, Marcelo E. Kaihara, Thorsten Kleinjung, Arjen K. Lenstra, "Solving a 112-bit Prime Elliptic Curve Discrete Logarithm Problem on Game Consoles using Sloppy Reduction", 2009.

[2] BOINC: http://boinc.berkeley.edu.

[3] Native Client: https://developer.chrome.com/native-client.

[4] J. Pollard, "Monte Carlo Methods for Index Computation (mod p)", Math. Comp, vol. 32, pp. 918-924, 1978.

[5] H. Cohen and G. Frey ed., "Handbook of Elliptic and Hyperelliptic Curve Cryptography", Chapman & Hall, 2006.

[6] S. Miyoshi, Y. Nogami, "Collision Detection with DNS in Rho Method on BN Curve", 2014 IEEE International Conference on Consumer Electronics - Taiwan, pp. 49-50, Taiwan, 2014.

[7] S. Miyoshi, Y. Nogami, T.Kusaka, N.Yamai, "Solving 94-bit ECDLP with 70 Computers in Parallel", 17th International Conference on Pairing-Based Cryptography, pp. 2491-2494, France, 2015.

[8] SETI@home: http:setiathome.berkeley.edu.

[9] S. Takaki, K. Watanabe, M. Fukushi, N. Amano, N. Funabiki, T. Nakanishi, "A proposal of Web Browser-based Volunteer Computing Platform", IPSJ SIG Technical Report, 2014-HPC-143(29), pp. 1-8, 2014. (in Japanese)

[10] Folding@home: https://folding.stanford.edu/.

[11] Folding@home Chrome Client: http://folding.stanford.edu/nacl/.

[12] GMP: https://gmplib.org/.

[13] PrimeGrid: http://www.primegrid.com/.