Parallel GPU Implementation of η_T Pairing over Fields of Characteristic Two

M. Ishii, A. Inomata, and K. Fujikawa

***Abstract*—Pairings on hyperelliptic curves have been applied to many cryptographic schemes, and it is important to exploit methods that increase the speed of various pairings and their curves. Additionally, multiple pairings should be performed efficiently in some cryptographic application such as attribute-based encryption or functional encryption. We propose an efficient extension field construction method that defines a curve and its η_T pairing. We also implemented the parallel arithmetic on extension fields and multiple η_T pairings in parallel and reported experimental timing results. We achieved timing of 12.7ms and 52.0ms per pairing when computed 1248 pairings by using GPU Tesla K20c. We took the extension degree of base field $m = 487$ which is greater than the parameter $m = 367, 439$ that was appropriate for the η_T pairing at the 128-bit security level. By normalization of experimental result, we achieved a certain level of speeding up of the η_T pairing compared to the state-of-the-art CPU implementation. In addition, we achieved scalability with the extension degree of base field in our parallel implementation by performing Karatsuba multiplications between multiple elements of extension field in parallel.**

***Index Terms*—η_T pairing, multiple pairings, GPU implementation, CUDA, karatsuba method, DLP in finite field of small characteristic, security level.**

I. Introduction

Koblitz [1] suggested a hyperelliptic cryptosystem us- ing Jacobians of hyperelliptic curves as arithmetic generalizations on groups of elliptic curves. Arithmetic on Jacobians of hyperelliptic curves is more complex than on elliptic curve groups. Alternatively, we can use smaller finite fields; i.e., we can employ smaller size keys by using higher genus curves to achieve the same level of security.

Pairings on elliptic curves or higher genus curves have attracted significant attention and have been applied to many cryptographic schemes, such as ID-based cryptography. Generally, calculation methods for pairings are complex and the cost of pairings is considerably higher than that of arithmetic on curves. In addition, the cost is significantly higher when using algebraic curves of higher genus.

Modern graphics processing unit (GPU) technology for general purposes, based on GPU computation has advanced significantly, while the use thereof in high level cryptography implementations has increased rapidly. There has been much research on increasing the speed of multiple-precision arithmetic or arithmetic on finite fields using GPUs, which is explored further in Section II.

In this study, we consider the parallelization of arithmetic on extension fields. The pairing algorithm is suitable for our parallel algorithm. As there are many parameters for pairings, by implementing pairings on a GPU, we can exploit parallelization methods to extend the program code flexibly. In the case that the field characteristic defining the curve and pairing is large, we can compute elements of the field in parallel as modular arithmetic on prime fields using a GPU [2]–[5]. On the other hand, if the characteristic is small, we can implement arithmetic on the field efficiently using a GPU and polynomial bases. Y. Katoh, Y. Huang, C. Cheng, and T. Takagi [6] implemented arithmetic on the $\mathbb{F}_{3^m}$ and η_T pairing defined on $\mathbb{F}_{3^m}$ using a GPU. They succeeded in accelerating the process significantly by computing multiple η_T pairings in a bit-sliced fashion. As the field characteristic is small, the degrees of the polynomials calculated as elements of the extension field are large; therefore, we can use the power of a GPU effectively within the context of parallelization.

Having focused on a parallel implementation of arithmetic on (extension) fields using polynomial bases, we have developed a practical and efficient method for parallelizing arithmetic on extension fields and a method for their construction that is suitable for our parallel algorithm. Indeed, we implemented parallel η_T pairing on a supersingular genus-two curve. We then used basis conversion to compute the η_T pairing to change the extension field construction making it suitable for parallel arithmetic on fields. We also achieved speedup of the η_T pairing using a different extension field construction method based on [7].

The remainder of this paper is organized as follows. We describe work related to the state-of-the-art software implementation of pairings and a GPU implementation for parallel modular arithmetic and pairings in Section II. In Section III, we recall η_T pairing on a genus-two curve over a binary field and its algorithm. We then describe recent research on the discrete logarithm algorithm in a finite field of small characteristic and the security level for the η_T pairing over binary fields. Section IV presents the detailed methodology of our parallel algorithm for arithmetic on extension fields and η_T pairing. We then report experimental timing results of the η_T pairing implementation on a GPU in Section V. Finally, we present our conclusions and suggestions for future work in Section VI.

II. Related Work

Here, we summarize state-of-the-art work related to a software implementation of pairings. First, we describe some

Manuscript received December 30, 2013; revised February 27, 2014.

M. Ishii is with Nara Institute of Science and Technology, Nara, Japan (e-mail: masahiro-i@is.naist.jp).

A. Inomata is with Initiative Center, Nara Institute of Science and Technology, Nara, Japan (e-mail: atsuo@itc.naist.jp).

K. Fujikawa is with Information Initiative Center, Nara Institute of Science and Technology, Nara, Japan (e-mail: fujikawa@itc.naist.jp).

DOI: 10.7763/IJCCE.2014.V3.318

work concerning the efficient implementation of pairings on a CPU. We then describe some research on implementing multiple-precision arithmetic on finite fields, modular arithmetic, and pairings, or other cryptographic applications using GPUs.

J. Beuchat, E. López-Trejo, L. Martínez-Ramos, S. Mitsunari, and F. Rodríguez-Henríquez [8] implemented a reduced modified Tate pairing ($\eta_T$pairing) on supersingular elliptic curves and designed a fast multi-core library using single-instruction multiple-data instructions. They reported a calculation time of just 1.87ms on Intel Core i7 architectures for Tate pairing at the 128-bit security level.

In [9], Beuchat *et al.* described the design of a fast software library for the computation of optimal Ate pairing on a Barreto–Naehrig elliptic curve. They re- ported that optimal Ate pairing at the 126-bit security level took 2.33 million clock cycles on a single core of an Intel Core i7 2.8 GHz processor.

D. F. Aranha, J. López, and D. Hankerson [10] implemented η_T pairing over bi- nary supersingular curves at the 128-bit security level in parallel (using two types of parallelism: vector instructions and multiprocessing). They reported parallel timings 66% faster than the result of [8]. S. Chatterjee, D. Hankerson, and A. Menezes [11] reported preliminary timings for Type 1 (symmetric) pairings on supersingular genus-2 curves of characteristic 2 at the 128-bit security level. The η_T pairing over $\mathbb{F}_{2^{439}}$ took 16.4 million clock cycles on an Intel Core 2 processor.

In [12], D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. López described efficient formulas for computing optimal Ate pairings on ordinary elliptic curves over prime fields. Their efficient techniques for computing pairings, for the first time allow a pairing to be obtained in under 2 million cycles on a 64-bit processor, improving the result of [9] by 28%–34%.

D. F. Aranha, J. Beuchat, J. Detrey, and N. Estibals [13] presented a novel optimal Eta pairing algorithm on supersingular genus-2 binary hyper-elliptic curves. According to their experimental results from a software implementation, an optimal Eta pairing on a genus-2 curve over $\mathbb{F}_{2^{367}}$ took 4.44 and 2.75 million clock cycles on an Intel Core 2 processor and Core i5 32 nm (Nehalem microarchitecture) processor, respectively.

Recently, Mitsunari [14] reported an efficient implementation of an optimal Ate pairing at the 126-bit security level (in [9], [12]) on an Intel Haswell processor. A mulx instruction supported by the Haswell processor was used to achieve the pairing in only 1.17 million clock cycles.

Efforts to speed up modular arithmetic using the power of GPUs are reported in [2], [5]. In [5], the authors implemented three modular arithmetic operations on a GPU: addition, subtraction, and multiplication. They used Montgomery's method for multiplication to avoid division. They also implemented arithmetic on elliptic curves using modular multiplication on a GPU.

Radix representation or a residue number system (RNS) can be used for modular multiplication in large finite fields. According to [3], [4], using a radix representation is superior to a RNS with regard to parallel implementation of modular multiplication on GPUs.

Katoh *et al.* [6] implemented arithmetic on $\mathbb{F}_{3^m}$ and an η_T pairing defined on $\mathbb{F}_{3^m}$ using a GPU. They succeeded in achieving significant speedups by computing multiple η_T pairings in a bit-sliced fashion. Essentially, they parallelized modular arithmetic represented by a polynomial basis by performing arithmetic on the basic field $\mathbb{F}_3$ in each thread. They implemented modular multiplication on a GPU by parallelizing the Comb method on a finite field [6, §3.2, Implementation II]. In addition, they implemented and evaluated parallel multiplication by computing 32 operations in a bit-sliced fashion. They reported that η_T pairing over $\mathbb{F}_{3^{509}}$ took 3.01ms on an NVIDIA GTX 480 and concluded that their GPU implementation for larger fields was slower than multi-core CPU implementations [8] owing to the limited fast on-die memory on the GPU. Y. Zhang, C. Jason-Xue, D. S. Wong, N. Mamoulis, and S. M. Yiu [15] were the first to present an evaluation of bilinear pairings over composite-order groups on graphics card hardware. They implemented parallelized base field operations via a RNS and performed multiple pairings to occupy the hardware resource. According to their experimental results, in 1024-bit base fields, the NVIDIA GTX 480 achieved a running time of 8.7ms per pairing, which is 19.6 times faster than the state-of-the-art CPU implementation.

III. η_T Pairing and Security Level

A. Algorithm of η_T Pairing

Here, we describe the η_T pairing [7] and discuss some of its properties. Barreto *et al.* exploited η_T pairing in [7] for supersingular curves as a generalization of the Duursma-Lee technique [16].

Suppose that $C/\mathbb{F}_q$ is a supersingular curve with embedding degree $k > 1$ and that a distortion map

$$\psi : C(\mathbb{F}_q) \to C(\mathbb{F}_{q^k})$$

is present. This allows denominator elimination, i.e., for $P \in C(\mathbb{F}_{q^k}), \psi(P) \in C(\mathbb{F}_{q^k})$ has an *x*-coordinate in $\mathbb{F}_{q^{k/2}}$.

Then, for $T \in \mathbb{Z}$, *Eta pairing* (η_T pairing) is given by

$$\eta_T : \mathrm{Jac}(\mathbb{F}_q)[r] \times \mathrm{Jac}(\mathbb{F}_q)[r] \to \mu_r \subset \mathbb{F}_{q^k}^{\times}$$

$$(D, E) \mapsto f_{(T,D)}(\psi(E))^{(q^k-1)/r}$$

Barreto *et al.* generalized the Duursma–Lee techniques, including effective calculation of divisors and using a Frobenius map, directly in Miller's algorithm. They succeeded in generalizing a loop shortening idea in many other cases. They described η_T pairing on a supersingular genus-two curve in detail in [7, §7]. We use this curve and consider η_T pairing under the same conditions.

We consider the supersingular curve

$$C : y^2 + y = x^5 + x^3$$

over $\mathbb{F}_{2^m}$ an embedding degree of 12; therefore, we have to perform arithmetic on the extension field $\mathbb{F}_{2^{12m}}$. In this study, we implemented η_T pairing using two methods to construct an extension field.

In the first method, we construct $\mathbb{F}_{2^{12m}}$ according to [7, §7.1], starting with a sixth degree extension. We then construct a quadratic extension as follows:

$$\mathbb{F}_{2^{6m}} \simeq \mathbb{F}_{2^m}[x]/(x^6 + x^5 + x^3 + x^2 + 1),$$
$$\mathbb{F}_{2^{12m}} \simeq \mathbb{F}_{2^{6m}}[y]/(y^2 + y + w^5 + w^3),$$
$$w^6 + w^5 + w^3 + w^2 + 1 = 0.$$

Let s_0 be a root of $y^2 + y + w^5 + w^3$. We call the following polynomial basis s_0w-basis,

$$\{1, w, w^2, w^3, w^4, w^5, s_0, s_0w, s_0w^2, s_0w^3, s_0w^4, s_0w^5\}.$$

The second method constructs $\mathbb{F}_{2^{12m}}$ by starting with a cubic extension of $\mathbb{F}_{2^m}$ for parallel arithmetic on the extension field. We define $\mathbb{F}_{2^{3m}}$ using the irreducible polynomial $x^3 + x + 1$ over $\mathbb{F}_{2^m}$. Letting w be one of the roots of $x^3 + x + 1$, we can represent elements of $\mathbb{F}_{2^{3m}}$ with basis $\{1, w, w^2\}$ over $\mathbb{F}_{2^m}$. Similarly, we consider the irreducible polynomial $y^2 + y + w + 1$ over $\mathbb{F}_{2^m}$ and let s be one of the roots of the polynomial. Hence, $\mathbb{F}_{2^{6m}}$ has the basis $\{1, w, w^2, s, sw, sw^2\}$. Finally, for the irreducible polynomial $z^2 + z + s + sw^2$ over $\mathbb{F}_{2^{6m}}$, let t be one of the roots of the polynomial. We can thus represent elements of $\mathbb{F}_{2^{12m}}$ with the basis

$$\{1, w, w^2, s, sw, sw^2, t, tw, tw^2, st, stw, stw^2\}.$$

We refer to this polynomial basis as stw-basis. In this paper, we examine a specific algorithm but do not describe the functions and algorithm for the η_T pairing in detail in each case. According to [7, §7], when we use the s_0w-basis, we can compute the η_T pairing as [7, §7.3, Algorithm 4]. We also present the η_T pairing algorithm using the stw-basis as Algorithm 1 using the same notation as in [7, §7.3, Algorithm 4].

As shown in Algorithm 1, we can calculate pairings in the same manner irrespective of which polynomial basis is used. However, using stw-basis reduces the cost of multiplication of $\alpha\beta$ in line 20. Therefore, we can implement $\alpha\beta$ efficiently in parallel. We need to perform 13 multiplications on base field by using Karatsuba method in order to compute $\alpha\beta$ with s0w-basis. We can reduce number of multiplication to 9 by constructing the 12-th extension field of with stw-basis.

B. Security Level for η_T Pairing

Security parameters for the pairings were chosen assuming Coppersmith's algorithm [17] or a generalization thereof [18] with heuristic complexity

$$L_Q\left(\frac{1}{3}, \left(\frac{32}{9}\right)^{\frac{1}{3}}\right)$$

where

$$L_Q(\alpha, c) = \exp\left(\left(c + o(1)\right)(\log Q)^{\alpha}(\log\log Q)^{1-\alpha}\right)$$

Therefore, the η_T pairing over $C/\mathbb{F}_{2^m}$ at the 128-bit security level was implemented by choosing $m = 367, 439$ since $\mathbb{F}_{2^{12\cdot 367}}$, $\mathbb{F}_{2^{12\cdot 439}}$ were assumed to be 128-bit security against Coppersmith attacks.

Recently, small theoretical and practical advancements of the efficient discrete logarithm problem (DLP) algorithm have been made [19]–[23]. G. Adj, A. Menezes, T. Oliveira, and F. Rodr ǵuez-Henr ǘquez [24] explained how the new algorithms by Joux [21] and R. Barbulescu, P. Gaudry, A. Joux, and E. Thom é[23] could be combined to solve the DLP in $\mathbb{F}_{3^{6\cdot 509}}$ faster than the Joux–Lercier algorithm [25]. They estimated the complexity (number of multiplications) to solve the DLP in $\mathbb{F}_{3^{6\cdot 509}}$ to be only $2^{73.7}$ compared with $2^{102.69}$ estimated by N. Shinohara, T. Shimoyama, T. Hayashi, and T. Takagi [26] using the Joux–Lercier algorithm and Joux's pin-pointing technique [19]. They also noted the case of characteristic 2 [24, Appendix A] and estimated that the complexity to solve the DLP in $\mathbb{F}_{2^{12\cdot 367}}$ was $2^{94.6}$, which is greater than $2^{91.6}$ estimated by them using the Joux–Lercier method with pinpointing. In addition, with access to a massive number of processors (233 processors), the bottleneck of the new algorithm would be a linear algebra computation with complexity 2^{60}.

Algorithm 1: η_T pairing ($m \equiv 7 \pmod 8$) using stw-basis

```
INPUT: $P = (x_P, y_P), Q = (x_Q, y_Q) \in \mathrm{Jac}(\mathbb{F}_{2^m})$
OUTPUT: $f \in \mathbb{F}_{2^{12m}}$
1: $x_1[i] \leftarrow x_P^{2^i}, y_1[i] \leftarrow y_P^{2^i}, x_2[i] \leftarrow x_Q^{2^i}, y_2[i] \leftarrow y_Q^{2^i}$ $(0 \le i \le m-1)$
2: $f \leftarrow 1$
3: for $i = 0$ to $(m-3)/2$ do
4:     Consider the following $k_j \pmod m$
5:     $k_1 \leftarrow (3m - 9 - 6i)/2, k_2 \leftarrow k_1 + 1, k_3 \leftarrow k_2 + 1$
6:     $k_4 \leftarrow (3m - 3 - 6i)/2, k_5 \leftarrow k_4 + 1, k_6 \leftarrow k_5 + 1$
7:
8:     Compute $\alpha \leftarrow a + bw + cw^2ds + t$
9:     $c \leftarrow x_1[k_4] + x_1[k_5]$
10:    $a \leftarrow y_2[k_2] + (c+1)x_2[k_3] + (x_1[k_4] + 1 + x_2[k_3])x_2[k_2] +$
           $y_1[k_4] + x_1[k_4] + \gamma_1(i) + 1$
11     $b \leftarrow x_2[k_3] + x_1[k_5] + \gamma_1(i) + 1$
12:    $d \leftarrow x_2[k_3] + x_1[k_5] + 1$
13:
14:    Compute $\beta \leftarrow e + f_2w + gw^2ds + t$
15:    $f_2 \leftarrow x_2[k_2] + x_1[k_6] + \gamma_1(i) + 1$
16:    $e \leftarrow y_2[k_1] + (x_1[k_6] + 1 + x_1[k_5])x_2[k_1] + y_1[k_5] +$
$dx_1[k_6] +$                $x_1[k_5] + \gamma_1(i)$
17:    $g \leftarrow x_2[k_2] + x_2[k_1] + \gamma_1$
18:
19:
20:    $f \leftarrow f(\alpha\beta)$
21: end for
22:
23: $x_P \leftarrow x_1[6(m-1)/2] + \gamma_1((m-1)1)/2)$
           $= x_1[m-3] + 1$ $(6(m-1)/2 \equiv m - 3 \pmod m)$
24: $y_P \leftarrow y_1[6(m-1)/2] + x_1[6(m-1)/2 + 1] + \gamma_3((m - 1/2)$
                   $= y_1[m-3] + x_1[m-2]$
25:
26: Perform the final doublings / addition
27: $u \leftarrow y_2[0] + x_2[1](1 + x_2[0] + x_P^8 + x_P^4) + x_P^4x_2[0] + y_P^4 + x_P^8 +$
       $x_P^4 + 1$
28:$f \leftarrow f^4(u, x_2[1] + x_2[0] + x_P^8 + x_P^4 + 1, x_2[1] + x_2[0], x_P^8 +$
       $x_2[0], 1,0,1,0,0,0,0,0)$
29:
30: Perform the final exponentiation
31: $f \leftarrow f^{(2^{6m}-1)(2^{3m}-2^{4m}2^{(m+1)/2}-1)}$
32: return $f$
```

In Fig. 1, we compare the heuristic complexity to solve the DLP in $\mathbb{F}_{2^{12\cdot n}}$ with Coppersmith algorithm, Joux–Lercier algorithm and Joux's pinpointing method. For implementation of the η_T pairing over $C/\mathbb{F}_{2^m}$ at the 128-bit security level, we take two extension degrees $m = 487, 967$.

Since Adj *et al.* estimated the DLP algorithm using Joux–Lercier pinpointing method in $\mathbb{F}_{2^{12\cdot 367}}$ with $2^{91.6}$, we should take m which is greater than at least $m = 439$. Although it needs concrete analysis for the complexity of DLP algorithm in $\mathbb{F}_{2^{12\cdot m}}$ as shown in [24, Appendix A], we evaluate parallel implementation of the η_T pairing over $C/\mathbb{F}_{2^{487}}$. In the second case, we take extension degree $m = 967$ which seems adequate parameter at the 128-bit security level and perform an experimental simulation of pairing computation using GPU.

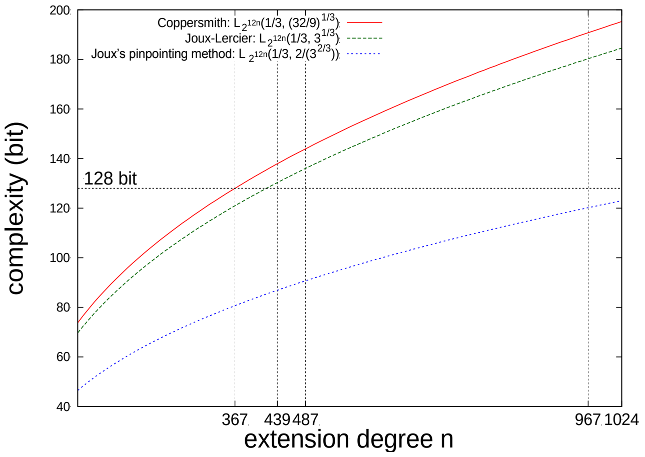

Fig. 1. Comparison of the complexity to solve the DLP in $\mathbb{F}_{2^{12\cdot n}}$.

IV. Methodology

In this section, we present the proposed method for parallelization of arithmetic on base fields and extension fields. Katoh *et al.* [6] implemented η_T pairing on a GPU. Similarly, we parallelized arithmetic on extension fields constructed by irreducible polynomials. We parallelized the arithmetic in a straightforward manner to allow flexible computation of extension field elements. In addition, we propose an approach for combining Karatsuba method parallelization and parallel arithmetic on extension fields.

A. Parallel Computation of Multiplication on Base Fields and Extension Fields

First, we introduce how to implement operations in base fields $\mathbb{F}_{2^m}$. We implemented elements of base fields as polynomial represented by uint64_t array. We then adopted left-to-right comb window method [27], therefore we took the word size of 64-bit and the window size of 4 with experimental results.

Katoh *et al.* implemented η_T pairing on a GPU by parallelizing the Comb method on finite fields [6, §3.2, Implementation III] in a bit-sliced fashion. We implemented straightforward parallelization of arithmetic of polynomials as arithmetic on base field. Indeed, for elements of base field

$$a(x) = \sum_{i=0}^{m} a_i\, x^i, \qquad b(x) = \sum_{i=0}^{m} b_i\, x^i \in \mathbb{F}_{2^m}$$

We compute the coefficients of $a(x) + b(x)$ by using XOR operation $a_1 \oplus b_1$ in parallel, and compute each c_i where

$$\sum_{i=0}^{2m} c_i\, x^i = c(x) = a(x)b(x)$$

in parallel with comb window method. In comb win-dow method, we add polynomials as elements of base field to $c(x)$ in m threads and we implement other calculation in serial.

In this study, we implemented arithmetic on extension fields using only the Karatsuba method. The Karatsuba method can be generalized for polynomials of arbitrary degree [28, §3.2, Algorithm 2]. We consider two polynomials of degree d,

$$A(x) = \sum_{i=0}^{d} a_i\, x^i, \qquad B(x) = \sum_{i=0}^{d} b_i\, x^i$$

For each $i = 0, \dots, d$, we compute

$$V_i \coloneqq a_i b_i \tag{1}$$

and for $0 \le s < t \le d$,

$$V_{s,t} \coloneqq (a_s + a_t)(b_s + b_t) \tag{2}$$

We can then compute $a(x)b(x) = \sum_{i=0}^{2d} c_i x^i$ as follows

$$c_0 = V_0,\ c_{2d} = V_d,$$

$$c_i = \begin{cases} \sum_{s+t=i} V_{s,t} - \sum_{s+t=i}(V_s + V_t) & i\text{: odd,} \\ \sum_{s+t=i} V_{s,t} - \sum_{s+t=i}(V_s + V_t) + V_{i/2} & i\text{: even,} \\ \text{for } 0 \le s < t \le d, 0 < i < 2d. \end{cases} \tag{3}$$

Thus, we can use the Karatusba method for multiplication in extension field $\mathbb{F}_{q^k}$ and reduce multiplication to 3, 6, 21, or 78 operations for $\mathbb{F}_q$ if $k = 2, 3, 6$ or 12, respectively.

In addition to the above parallel method for operations in base field, we consider the Karatsuba parallelization method for arithmetic on extension fields. We parallelize the precomputation phase of the Karatsuba method as follows. First, we start to compute V_i (1) in parallel. At the same time we compute $(a_s + a_t), (b_s + b_t)$ (2) and $V_{s,t}$ (2) in parallel. After that, we compute c_i (3) in serial.

B. Implementation of η_T Pairing on GPU

We implemented our parallel algorithm on a GPU, the NVIDIA Tesla K20c, and used the Compute Unified Device Architecture (CUDA) programming model [29]. The experimental environment is presented in Table I.

TABLE I: Experimental Environment.

OS	Fedora 19
CPU	Intel Core i7-4770K, 3.50GHz, 4 Cores
Memory	DDR3-1333, 32GB
GPU	Tesla K20c, 2496 CUDA Cores Graphics Clock: 706 MHz
Compiler	GCC-4.8.2, NVCC-5.5
Compute Capability	3.5

The extension degree directly affects the time for parallel arithmetic computations on the extension field. We implemented and evaluated η_T pairing for extension degrees $m = 487$ and $m = 967$ as described in previous section about security level. Since elements of base field are represented uint64_t array, we need to take the length of 8 and 16 respectively for the extension degree $m = 487$ and $m = 967$. Therefore, we implemented that 8 or 16 threads handle each operation in base field using CUDA programming model. In addition, we computed multiple pairings in order to use GPU resource effectively. We implemented arithmetic between multiple elements in base fields and extension fields in parallel by using blocks in CUDA programming. Each block handles calculation with independent elements of fields in parallel using multiple threads, therefore we could compute multiple η_T pairings independently.

Basically, we implemented operations in fields by starting to copy data to global memory which is an off-chip memory device that can be accessed by any thread and the device's access time is higher than that of other memory operations. We then stored intermediate variable to shared memory and after that finished to calculate on GPU, we copied the results to main memory. Indeed, we used shared memory to store precomputation table in window method, and the limit of the window size when $m = 967$ was 8 since the size of shared memory on Tesla K20c was 48KB. We implemented parallel Karatsuba method of the degree $d = 2, 5$ respectively for extension degree of 3, 6 and combined the Karatsuba method in order to do multiplication on 12-th extension field for each construction using s_0w -basis, stw -basis. We computed additions on fields except per- formed in Karatsuba method on CPU since it is enough fast to perform additions between multiple elements of fields.

V. Evaluation

In this section, we showed the experimental result of the implementation of η_T pairing on GPU. First, we reported comparison of the total time to compute multiple η_T pairings on CPU and GPU. In Fig. 2 and Fig. 3, we describe total time of CPU and GPU implementation of the multiple η_T pairings over $\mathbb{F}_{2^{487}}, \mathbb{F}_{2^{967}}$ respectively, with s_0w -basis, stw -basis. CPU implementation means that we compute multiple pairings in serial with same parallel algorithm for GPU implementation.

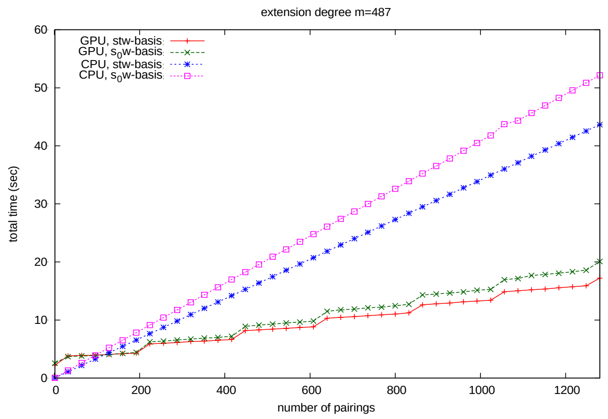

Fig. 2. Total time of multiple η_T pairings over $\mathbb{F}_{2^{487}}$.

As shown in Fig. 2 and Fig. 3, timings of GPU implementation are slower than CPU implementation with small number of pairings since GPU resource is not used sufficiently. We can see that the total time in the case of $\mathbb{F}_{2^{967}}$ is seemed to have more effect on parallelization compared to the case of $\mathbb{F}_{2^{487}}$. The construction for 12-th extension field using stw -basis have an impact on timing of CPU implementation over $\mathbb{F}_{2^{967}}$ since the cost of multiplication is bigger than the case of $m = 487$.

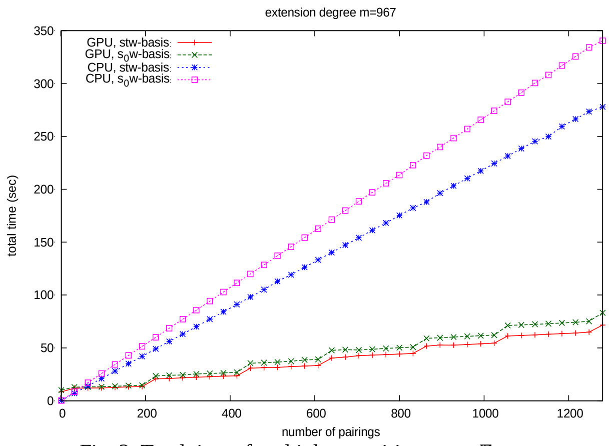

Fig. 3. Total time of multiple η_T pairings over $\mathbb{F}_{2^{967}}$.

We then report the timing results for computing per pairing on GPU in Fig. 4

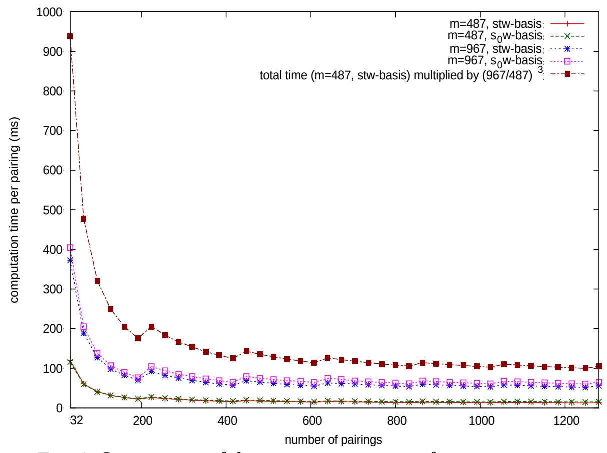

Fig. 4. Comparison of the computation time of per η_T pairing.

In this experimental result, we achieved fast timings of 12.7ms and 52.0ms per pairing when computed 1248 parings. We can consider that the timing per pairing goes down as increasing number of pairings. In addition, although the order of growth in algorithm for the η_T pairing is roughly $O(m^3)$ where m is extension degree, timing per η_T pairing over $\mathbb{F}_{2^{967}}$ is less than the estimation which is showed by the uppermost line in Fig. 4.

We achieved timing per pairing of 12.7ms per pairing using GPU Tesla K20c which the core clock is 706 MHz. In regard to security level for new DLP algorithm, we took the extension degree of base field $m = 487$ that is greater than $m = 367, 439$. As future works, we tackle effective management and use of memories in particular registers on GPU. We believe that can achieve significant speed up compared to state-of-the-art result of CPU or GPU implementation by optimizing our approach.

VI. Conclusions

In this study, we implemented the parallel arithmetic on extension fields and multiple η_T pairings in parallel. In addition, we used effective construction of extension field so that we could reduce the cost of the η_T pairing. We achieved timing of 12.7ms and 52.0ms per pairing when computed 1248 pairings by using GPU Tesla K20c. We took the extension degree of base field $m = 487$ which is greater than the parameter $m = 367, 439$ that was appropriate for the η_T pairing at the 128-bit security level. By normalization of experimental result, we achieved a certain level of speeding up of the η_T pairing compared to the state-of-the-art CPU implementation.

As shown in the Section III and Section II, it is adequately considered that the extension degree $m = 487$ is not appropriate for the η_T pairing at the 128-bit security in the fields of characteristic 2. However, the parallelize method of Karatsuba multiplication on extension field we proposed can be basically apply to performing modular multiplication in the case of large characteristic. In addition, we achieved scalability with the extension degree of base field in our parallel implementation and that also held in the case of large characteristic.

References

[1] N. Koblitz, "Hyperelliptic cryptosystems," *Journal of Cryptography*, vol. 1, pp. 139–150, 1989.

[2] S. Fleissner, "Gpu-accelerated montgomery exponentiation," in *Proc. International Conference on Computational Science 2007*, 2007, vol. 4487, pp. 213–220.

[3] R. Szerwinski and T. Güneysu, "Exploiting the power of GPUs for asymmetric cryptography. In cryptographic hardware and embedded systems," in *Proc. CHES 2008, 10th International Workshop*, 2008, vol. 5154, pp. 79–99.

[4] O. Harrison and J. Waldron, "Public key cryptography on modern graphics hardware," presented at the Eurocrypt 2009, Cologne, Germany, April 26-30, 2009.

[5] P. Giorgi, T. Izard, and A. Tisserand, "Comparison of modular arithmetic algorithms on GPUs," in *Proc. ParCo'09, International Conference on Parallel Computing*, pp. 119–133, France, 2009.

[6] Y. Katoh, Y. Huang, C. Cheng, and T. Takagi, "Efficient implementation of the η_T pairing on GPU," in *Proc. 9th International Conference on Applied Cryptography and Network Security, ACNS 2011, Industrial Track*, 2011, pp. 119–133.

[7] P. S. L. M. Barreto, S. Galbraith, C. Ó hÉigeartaigh, and M. Scott, "Efficient pairing computation on supersingular abelian varieties," *Designs, Codes and Cryptography*, vol. 42, pp. 239–271, 2007.

[8] J. Beuchat, E. López-Trejo, L. Martínez-Ramos, S. Mitsunari, and F. Rodríguez-Henríquez, "Multi-core implementation of the tate pairing over supersingular elliptic curves," *Lecture Notes in Computer Science*, vol. 5888, pp. 413–432, 2009.

[9] J. Beuchat, J. E. González-Díaz, S. Mitsunari, E. Okamoto, F. Rodríguez-Henríquez, and T. Teruya, "High-speed software implementation of the optimal ate pairing over Barreto-Naehrig curves," *Lecture Notes in Computer Science*, vol. 6487, pp. 21–39, 2010.

[10] D. F. Aranha, J. López, and D. Hankerson, "High-speed parallel software implementation of the η_T pairing," *Lecture Notes in Computer Science*, vol. 5985, pp. 89–105, 2010.

[11] S. Chatterjee, D. Hankerson, and A. Menezes, "On the efficiency and security of pairing-based protocols in the type 1 and type 4 settings," *Lecture Notes in Computer Science*, vol. 6087, pp. 114–134. 2010.

[12] D. F. Aranha, K. Karabina, P. Longa, C. H. Gebotys, and J. López, "Faster explicit formulas for computing pairings over ordinary curves," *Lecture Notes in Computer Science*, vol. 6632, pp. 48–68. 2011.

[13] D. F. Aranha, J. Beuchat, J. Detrey, and N. Estibals, "Optimal eta pairing on supersingular genus-2 binary hyperelliptic curves," *Lecture Notes in Computer Science*, vol. 7178, pp. 98–115, 2012.

[14] S. Mitsunari. (2013). A fast implementation of the optimal ate pairing over BN curve on intel haswell processor. *Cryptology ePrint Archive Report 2013/362*. [Online]. Available: http://eprint.iacr.org/2013/362, 2013.

[15] Y. Zhang, C. Jason-Xue, D. S. Wong, N. Mamoulis, and S. M. Yiu, "Acceleration of composite order bilinear pairing on graphics hardware," in *Proc. the 14th International Conference on Information and Communications Security*, 2012, pp. 341–348.

[16] I. Duursma and H. Lee, "Tate pairing implementation for hyperelliptic curves $y^2 = x^p - x + d$," *Lecture Notes in Computer Science*, vol. 2894, pp. 111–123, 2003.

[17] D. Coppersmith, "Fast evaluation of logarithms in fields of characteristic two," *IEEE Transactions on Information Theory*, vol. 30, pp. 587–594, 1984.

[18] L. Adleman and M. Huang, "Function field sieve method for discrete logarithms over finite fields," *Information and Computation*, vol. 151, pp. 5–16, 1999.

[19] A. Joux, "Faster index calculus for the medium prime case application to 1175-bit and 1425-bit finite fields," *Lecture Notes in Computer Science*, vol. 7881, pp. 177–193, 2013.

[20] F. Göloğlu, R. Granger, G. McGuire, and J. Zumbrägel. (2013). On the function field sieve and the impact of higher splitting probabilities: application to discrete logarithms in $\mathbb{F}_{2^{1971}}$ and $\mathbb{F}_{2^{3164}}$. *Cryptology ePrint Archive, Report 2013/074*. [Online]. Available: http://eprint.iacr.org/2013/074

[21] A. Joux. A new index calculus algorithm with complexity $L(1/4 + o1$ in very small characteristic. *Cryptology ePrint Archive, Report 2013/095*. [Online]. Available: http://eprint.iacr.org/2013/095

[22] F. Göloğlu, R. Granger, G. McGuire, and J. Zumbrägel. Solving a 6120-bit dlp on a desktop computer. *Cryptology ePrint Archive, Report 2013/306*. [Online]. Available: http://eprint.iacr.org/2013/306

[23] R. Barbulescu, P. Gaudry, A. Joux, and E. Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. *Cryptology ePrint Archive, Report 2013/400*. [Online]. Available: http://eprint.iacr.org/2013/400

[24] G. Adj, A. Menezes, T. Oliveira, and F. Rodríguez-Henríquez. Weakness of $\mathbb{F}_{3^{6\cdot 509}}$ for discrete logarithm cryptography. *Cryptology ePrint Archive, Report 2013/446*. [Online]. Available: http://eprint.iacr.org/ 2013/446

[25] A. Joux and R. Lercier, "The function field sieve in the medium prime case," *Lecture Notes in Computer Science*, vol. 4004, pp. 254–270, 2006.

[26] N. Shinohara, T. Shimoyama, T. Hayashi, and T. Takagi, "Key length estimation of pairing-based cryptosystems using η_T pairing," *Lecture Notes in Computer Science*, vol. 7232, pp. 228–244, 2012.

[27] D. Hankerson, A. Menezes, and S. Vanstone, *Guide to Elliptic Cryptography*. Springer-Verlag, 2004, ch. 2, pp. 48–51.

[28] A. Weimerskirch and C. Paar, "Generalization of the karatsuba algorithm for efficient implementations," Cryptology ePrint Archive, Report 2006/224, [Online]. Available: http://eprint.iacr.org/2006/224

[29] Cuda zone. [Online]. Available: http://developer.nvidia.com/ category/zone/cuda-zone.

Masahiro Ishii received M.S. degree from Nagoya University in 2011 and received M.E. degree from Nara Institute of Science and Technology in 2013 and is currently a Ph.D. candidate in Graduate School of Information Science at Nara Institute of Science and Technology. His research interest is pairing-based cryptography, efficient software implementation of finite field arithmetic including GPU programming, and algorithm for discrete logarithm problem in finite fields.

Atsuo Inomata received M.E. degree in information science from 1997 to 1999 and a Ph.D. in information science from Japan Advanced Institute of Science and Technology in 2002. Currently, he is an associate professor in Graduate School of Information science, Nara Institute of Science and Technology. His research focuses on cryptography, information security. He is a member of IEICE, IPSJ, and JSISE.

Kazutoshi Fujikawa received M.E. and Ph.D. degrees in information and computer sciences from Osaka University in 1990 and 1993, respectively. Currently, he is a professor of Information Initiative Center, Nara Institute of Science and Technology. His research focuses on multimedia communication systems, digital libraries, ubiquitous computing, and mobile net-works. He is a member of ACM, IEEE, and IPSJ.