Efficient Hardware Architecture of η_T Pairing Accelerator Over Characteristic Three

Szu-Chi Chung, *Student Member, IEEE*, Jing-Yu Wu, Hsing-Ping Fu, Jen-Wei Lee, *Student Member, IEEE*,
Hsie-Chia Chang, *Member, IEEE*, and Chen-Yi Lee, *Member, IEEE*

***Abstract*— To support emerging pairing-based protocols related to cloud computing, an efficient algorithm/hardware codesign methodology of η_T pairing over characteristic three is presented. By mathematical manipulation and hardware scheduling, a single Miller's loop can be executed within 17 clock cycles. Furthermore, we employ torus representation and exploit the Frobenius map to lower the computation cost of final exponentiation. Pipelining and parallelization datapath are also exploited to shorten the critical path delay. Finally, by choosing suitable multiplier architecture and selecting an appropriate number of multipliers, Miller's loop and final exponentiation can be computed in a fully pipelined manner. With these schemes, a test chip for the proposed pairing accelerator has been fabricated in 90-nm CMOS 1P9M technology with a core area of 1.52 x 0.97 mm^2. It performs a bilinear pairing computation over $F(3^{97})$ in 4.76 μs under 1.0 V supply and achieves 178% improvement to relative works in terms of area–time (AT) product. To support higher level of security, a 126-bit secure pairing accelerator that can complete a bilinear pairing computation over $F(3^{709})$ in 36.2 μs is implemented and this result is at least 31% better than relative works in terms of AT product.**

***Index Terms*— Application-specific integrated circuit (ASIC) implementation, elliptic curve, η_T pairing.**

I. Introduction

In 2000, Mitsunari *et al.*, Sakai *et al.*, and Joux independently discovered constructive properties of bilinear pairing [1]. One year later, Boneh and Franklin [2] solved a long lasting problem of identity-based cryptography based on bilinear pairing. Since then, an ever increasing number of protocols based on the bilinear pairing have appeared in the literature.

In recent years, cloud computing becomes a promising alternative to traditional local services. However, security and privacy issues may prevent wide acceptance in practice since the data no longer store on personal devices. To provide privacy and enhance security for users, myriad of cryptography protocols based on bilinear pairing have been presented to resolve this problem, such as [3] and [4]. Specifically, Sahai and Waters [3] introduced a protocol realizing the confidentiality and fine-grained access control of data based on the attribute-based encryption. On the other hand, Boneh *et al.* [4] facilitate the data owner efficiently search the files stored by cloud servers while prevents cloud servers from learning both the data file contents and user query information. Furthermore, the properties of bilinear pairing allows ID-based authentication, ID-based encryption, and hence certificateless key management becomes practical [5]. In 2008, IEEE established the draft standard for pairing-based cryptography [6].

Such protocols rely critically on efficient algorithms and implementations of pairing primitives. According to [1], when dealing with general curves providing common levels of security, the Tate pairing is more efficiently computable than the Weil pairing. Significant improvements were independently proposed in [7] and [8]. Barreto *et al.* [9] and Hess *et al.* [10] introduced the η_T pairing and Ate pairing, respectively, which further shortens the loop of Miller's algorithm. We choose η_T pairing in this paper since it is symmetric pairing, which can support more protocols. Moreover, the η_T pairing is defined on supersingular curve, which can achieve substantially computation reduction by choosing suitable distortion map and using ternary field arithmetic.

The η_T pairing contains two major steps [1]: 1) Miller's loop and 2) final exponentiation. To enhance throughput, the hardware of Miller's loop and final exponentiation can work independently, as the former data are completed with the computation of Miller's algorithm and follows by the final exponentiation, the latter data can be activated to start the computation of Miller's algorithm simultaneously. In [11]–[13], however, the computation time of Miller's algorithm and final exponentiation are far from balanced, which is hard to employ fully pipeline techniques to enhance throughput.

In this paper, a new η_T pairing accelerator for high-speed pairing-based protocols is proposed. To reduce the execution cycles of Miller's loop, algorithm selection and hardware scheduling are exploited and analyzed. Moreover, the torus representation and mathematical manipulation are used to shorten the final exponentiation time. In addition, the number and architecture of multipliers are selected and scheduled so that these two major steps can be computed in a fully pipelined manner. The major contributions of this paper are highlighted here.

1) Considering the recent attack proposed in [14], we suggest to use larger field to achieve 126-bit security level. To mitigate the corresponding increase of computation

Manuscript received May 25, 2013; revised November 23, 2013; accepted January 14, 2014. Date of publication February 19, 2014; date of current version January 16, 2015. This work was supported by the National Science Council, Ministry of Economic Affairs of Taiwan, under Grant 100-2221-E-009-044-MY3 and Grant 102-2220-E-009-044-MY3.

The authors are with the Department of Electronics Engineering and Institute of Electronics, National Chiao Tung University, Hsinchu 30010, Taiwan (email: steve2003121.eecs96@g2.nctu.edu.tw; justbluei@hotmail.com; irretta@gmail.com; jenweilee@gmail.com; hcchang@mail.nctu.edu.tw; cylee@si2lab.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2014.2303489

overhead, we carefully choose algorithm parameters including reduction polynomial, field element encoding method, paring algorithm, and so on.

2) This paper reducing the computation cost of both Miller's loop and final exponentiation by applying several state-of-the art optimization.
3) By designing suitable multiplier architecture and employing pipelining and parallelization, the implementation result outperforms relative works in terms of speed and hardware efficiency.

The rest of this paper is organized as follows. The algorithm of η_T pairing as well as its parameter selections are presented in Section II. The improved pairing arithmetic including Miller's loop and final exponentiation are proposed and analyzed in Section III. Moreover, Section IV reports the hardware architecture of our proposed pairing accelerator as well. The measurement results of a 90-nm test chip and 126-bit security level accelerator as well as the comparisons against relative works are given in Section V. Finally, Section VI concludes this paper.

II. Background

A. Symmetric Pairings

In most of the protocols, symmetric pairing is often selected as it allows simpler and briefer mathematical statements and definitions. Let $G1$ be an abelian group with additive identity element O and r is a positive integer. Suppose $G1$ has order r, which means $[r]P = O$ for all $P \in G1$. Suppose $G2$ is a cyclic group of the same order r with multiplicative identity element one. A symmetric pairing is a map

$$e : G1 \times G1 \rightarrow G2.$$

Note e should be feasibly computable, bilinear in both group, and nondegenerate, which enables the construction of novel and efficient cryptography protocols.

Let n be an integer, then the most crucial function in pairing is Miller's function $f_{n,P}$, and the divisor of Miller's function is defined as

$$(f_{n,P}) = n(P) - ([n]P) - (n-1)(O).$$

Miller's functions are at the root of most pairing proposed for cryptographic purpose. We refer the reader to [1] for the mathematical details of divisor. To compute the function, one can use Miller's algorithm [15], which shows that a Miller's function satisfies the following observation up to a nonzero factor in F_q:

$$f_{m+n,P} = f_{m,P} \cdot f_{n,P} \cdot l_{[n]P,[m]P}/v_{[n+m]P}$$

where $l_{[n]P,[m]P}$ denotes the line through $[n]P$ and $[m]P$, and $v_{[n+m]P}$ is the vertical line through $[n+m]P$. These two functions are so called line function in the literature.

An efficient algorithm can be derived from the above formula, since with this relation, the line function and hence Miller functions can be computed using usual chord-and-tangent method on elliptic curve.

Algorithm 1 Miller's Loop Without Cube Roots of η_T Pairing

```
Input: $P, Q \in E(F_{3^m})[r], t, u \in F_{3^m}$, and $F, G \in F_{3^{6m}}$.
Output: $\eta_T(P, Q) \in F^*_{3^{6m}}$
$x_P = x_P + b$
$y_P = -y_P$
$x_Q = {x_Q}^3; y_Q = {y_Q}^3$
$t = x_P + x_Q$
$F = (-y_P t + y_Q\sigma + y_P\rho) \cdot (-t^2 + y_P y_Q \sigma - t\rho - \rho^2)$
For $i$ from 1 to $(m-1)/2$ do
    $F = F^3$
    $x_Q = {x_Q}^9 - b; y_Q = -{y_Q}^9$
    $t = x_P + x_Q$
    $u = y_P y_Q$
    $G = \sigma u - t^2 - \rho t - \rho^2$
    $F = FG$
EndFor
Return $F$.
```

B. η_T Pairing

Let r be the largest prime factor of N, so that r^2 is not a factor of N, where N is the order of the curve. Then, we can write $N = i \cdot r$, where i is a small positive integer. The η_T pairing is a symmetric pairing that maps two points in r-torsion groups into an element of the group of rth roots of unity μ_r

$$E(F_{3^m})[r] \times E(F_{3^m})[r] \rightarrow \mu_r \subset F^*_{3^{km}}.$$

The embedding degree or security multiplier is the least positive integer k for which μ_r is contained in the multiplicative group $F^*_{3^{km}}$. The curve used in this paper has $k = 6$, which is the maximum value possible for supersingular elliptic curves. Moreover, let $P, Q \in E(F_{3^m})[r]$ and $T = 3^m - N$, construct the divisor $D = (Q) - (O)$, and $(f_{r,P}) = r(P) - r(O)$. Such that supports of D and $(f_{r,P})$ are disjoint, then η_T pairing function $\eta_T(P, Q)$ is defined by[1]

$$f_{T,P}(\psi(D)) = \left(\prod_{i=0}^{(m-1)/2} g_{3^i P}(\psi(D))^{3^{(m-1)/2-i}} \right) l_V(\psi(Q)).$$

The above formula contains cube root operations. However, cube root is substantially slower than calculations of cube in hardware, thus we adopt a cube-free implementation of η_T pairing, as shown in Algorithm 1 [11].

The result of η_T pairing is mapped to a coset. To ensure unique output value of η_T pairing, we have to raise it up to Mth power, where $M = (3^{6m} - 1)/N$. This additional step is known as final exponentiation. As suggested by [16], the final exponentiation can be further factored to $(3^{3m} - 1)$ $(3^m + 1)(3^m + 1 \pm 3^{(m+1)/2})$, which can lower the computation cost. To compute the final exponentiation, [6] suggested to

[1] $\psi(x, y) = (\rho - x, \sigma y)$ is the distortion map used in this paper, where $\sigma^2 = -1$ and $\rho^3 = \rho + b, \sigma, \rho \in F_{3^{6m}}$. Duursma and Lee integrate distortion map into Miller's algorithm and introduce the g_V function. g_V is a rational function defined over $E(F_{3^{6m}})[r]$, and having divisor $(g_V) = 3(V) + ([-3]V) - 4(O)$. For all $V = (x_V, y_V) \in E(F_{3^m})[r]$ and $(x, y) \in E(F_{3^{6m}})[r]$, it is defined as $g_V(x, y) = {y_V}^3 - ({x_V}^3 - x + b)^2$. l_V is the line equation corresponding to addition of $[3^{(m+1)/2}]V$ with $[b]V$.

Algorithm 2 Final Exponentiation of η_T Pairing

```
Input: $F = \eta_T(P, Q)$
Output: $F^M$
$U = F^{3^{3m}-1}$
$V = U^{3^m+1}$
$W = V$
$X = V^{3^m+1}$
For $i$ from $1$ to $(m+1)/2$ do
    $W = W^3$
EndFor
    $Y = W^{-b}$
Return $XY$.
```

Algorithm 3 Parallel–Serial Multiplication Over F_{3^m}

```
Input: $A = \sum_{i=0}^{m-1} a_i x^i, B = \sum_{i=0}^{m-1} b_i x^i, a_i, b_i \in F_3$
Outputs: $C = A \cdot B = \sum_{i=0}^{m-1} c_i x^i, c_i \in F_3$
$s(x) = 0$
$t = 0$
For $i$ from $\lceil m/D \rceil$ downto $-1$ do
    $t(x) = \sum_{j=0}^{D-1} a_{D_{i+j}} x^j b(x)$
    $s(x) = t(x) + x^D \cdot (s(x) \mod f(x))$
    EndFor
Return $s(x)/x^D$.
```

used composite field representation $\eta_T = F = f_0 + f_1\sigma + f_2\rho + f_3\sigma\rho + f_4\rho^2 + r_5\sigma\rho^2$, and the details are illustrated in Algorithm 2.

C. Parameter Selection

Let E be a supersingular elliptic curve defined by $E : y^2 = x^3 - x + b$, with $b \in \{-1, 1\}$. The order of curve is given as $N = 3^m + 1 + \mu b 3^{(m+1)/2}$, with

$$\mu = \begin{cases} +1, & \text{if } m \equiv 1, 11 \pmod{12} \\ -1, & \text{if } m \equiv 5, 7 \pmod{12}. \end{cases}$$

As suggested by [17], one can choose a reduction polynomial as a trinomial of the form $P(x) = x^m - x^n + 1$ so that the computation of cubing can be achieved by several additions. Furthermore, the exponent n should be small, so that the cubing cost can be further reduced. Because of PARI/GP [18], we can obtain optimal trinomials in this paper.

Galbraith *et al.* [8] showed how to compute additions of two elements $a, b \in F_3$ using 12 AND, OR, XOR, and NOT Boolean functions. Harrison *et al.* [19] noted that this operation could be computed using only seven OR and XOR logic operations. This was considered the minimal number of logical operations for this arithmetic operation until Kawahara *et al.* [20] presented an expression that only requires six logical instructions. However, the benefits of this construction are that the multiplication over F_3 is precomputed or it would require pre/postprocessing between addition and multiplication, which is not suitable for high-speed applications. Therefore, we choose the encoding method proposed by Harrison.

Using Harrison's encoding method, an element $a \in F_3$ can be written by two bits like $a = (a_h, a_l)$ for $a_h, a_l \in \{0, 1\}$, $a_h = a/2, a_l = a \mod 2$. Specifically, the symbols $(0, 0), (0, 1), (1, 0)$ mean $0, 1, 2$, respectively. Elements of the field F_{3^m} are represented as polynomial basis. The field F_{3^m} can be regarded as $F_3[x]/f(x)$, where the $f(x)$ is a degree-m irreducible trinomial.

The multiplication $c = a \cdot b$ in F_3 can be written as: $c_l = (a_h \wedge b_h) \vee (a_l \wedge b_l)$ and $c_h = (a_h \wedge b_l) \vee = (a_l \wedge b_h)$. For m-bit long F_3 multiplication, we use serial–parallel scheme to explore the tradeoff between time and area. By [21], we can implement the multiplication over F_{3^m} by processing an operand with D words at each clock cycle. Therefore, in each step, we compute the degree $m + D - 2$ partial product polynomials: $t(x) = \sum_{j=0}^{D-1} a_{D_{i+j}} x^j b(x)$. All the partial products are summed up by a degree $m + D - 1$ accumulator polynomial: $s(x) = t(x) + x^D \cdot (s(x) \mod f(x))$. After $\lceil m/D \rceil$ steps, the output of $a(x)b(x) \mod f(x)$ is equal to the polynomial $s(x)$. Algorithm 3 summarizes this multiplication scheme.

Cubing over F_{3^m} is a simple arithmetic operation over the irreducible trinomial. The operation number of addition/subtraction of cubing is bounded by $m + 2/3 \cdot (2k + n) - 3$, we refer the reader to [17] for the detail of derivation.

Instead of designing a specific operator based on the extended Euclidean algorithm, we suggest to keep the circuit area as small as possible by performing inversion according to [22], which is based on Fermats little theorem. Since this scheme requires only multiplications and cubings over F_{3^m}, we do not have to include dedicated hardware for inversion in our coprocessor.

The security of the pairing is determined by the difficulty of the Discrete Logarithm Problem (DLP) on the input curve and on the output multiplicative group. The embedding degree k acts as a cursor to adjust the size of the multiplicative group $F^*_{q^k}$ with respect to that of F_q. The best known algorithm to attack the DLP on the r-torsion is Pollard's ρ method [23] while the functional field sieve (FFS) [24] is generally used to attack DLP on $\mu_r \subset F^*_{q^k}$. According to [14], the improved FFS can solve the DLP with time complexity

$$\exp(((32/9)^{1/3} - 0.18)(\ln 3^{6m})^{1/3}(\ln \ln 3^{6m})^{2/3}).$$

The time complexity of the improved FFS versus m is plotted in Fig. 1. We can find that the previous work, which builds on $F_{3^{509}}$ [25], is not sufficient to provide the 128-bit security level, as suggested in [26]. Therefore, we list in Table I for a selection of prime extension degrees m that enjoy large r-torsion subgroups. For a given extension degree m, we use PARI/GP [18] to find its corresponding curve, reduction trinomials, and the estimated security level.

III. Proposed η_T Pairing Arithmetic

In Algorithm 1, the composite field representation is used, so that all the operations over $F_{3^{6m}}$ can be replaced by arithmetic over F_{3^m}. In Algorithm 2, instead, we suggest to use torus $T_2(F_{3^{3m}})$ to compress the value of $F^*_{3^{6m}}$, which can further reduced the cost of field operations compared with composite field representation.

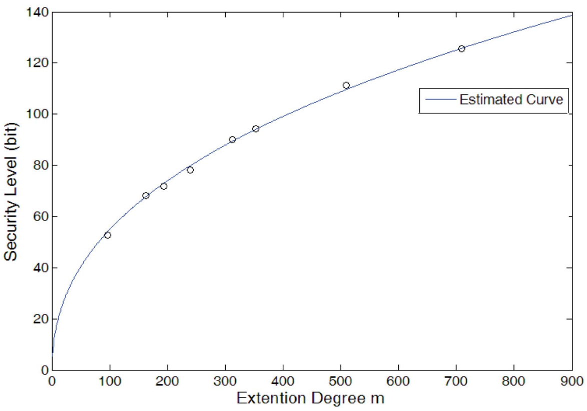


Fig. 1. Time complexity estimation of solving DLP over $F(3^{6m})$.

TABLE I
PARAMETER SELECTION

m	security level	b	μ	reduction trinomial
97	53	1	1	$x^{97} - x^{12} + 1$
163	68	-1	-1	$x^{163} - x^{59} + 1$
193	72	-1	1	$x^{193} - x^{64} + 1$
239	78	-1	1	$x^{239} - x^{5} + 1$
313	90	-1	1	$x^{313} - x^{93} + 1$
353	94	-1	-1	$x^{353} - x^{142} + 1$
509	111	1	-1	$x^{509} - x^{151} + 1$
709	126	-1	1	$x^{709} - x^{117} + 1$

A. Miller's Loop

The most critical part of Algorithm 1 is at line 12, which involves multiplication over $F_{3^{6m}}$. Notice that it is a sparse (i.e., some of its terms are trivial) multiplication over $F_{3^{6m}}$, which allows us to optimize the computation. Bertoni *et al.* [27] and Gorla *et al.* [28] take advantage of Karatsuba multiplication and Lagrange interpolation, respectively, to reduce the number of multiplications over F_{3^m} at the expense of several additions. However, to keep the pipeline of our multiplier busy, we have to embed a large multioperand adder and irregular datapath, which would deteriorate the clock frequency. Thus, we do some manipulation on the formula to reduce the number of additions while keeping the number of multiplications small. We first represent F and G using the composite field representation as follows:

$$
\begin{aligned}
F &= f_0 + f_1\sigma + f_2\rho + f_3\sigma\rho + f_4\rho^2 + f_5\sigma\rho^2 \\
G &= g_0 + g_1\sigma + g_2\rho + g_3\sigma\rho + g_4\rho^2 + g_5\sigma\rho^2.
\end{aligned}
$$

According to line 7 in Algorithm 1, $g_3 = 0$, $g_4 = -1$, and $g_5 = 0$ here, and we combine lines 12 and 7 in Algorithm 1 so that $F = FG$ becomes $F = F^3G$. Furthermore, F^3 can be represented as $({f_0}^3 + {f_2}^3 + {f_4}^3) + (-{f_1}^3 - {f_3}^3 - {f_5}^3)\sigma + ({f_2}^3 - {f_4}^3)\rho + (-{f_3}^3 + {f_5}^3)\sigma\rho + {f_4}^3\rho^2 + (-{f_5}^3)\sigma\rho^2$, we thus obtain

$$
\begin{aligned}
{f'}_0 &= ({f_0}^3 + {f_2}^3 + {f_4}^3)(-t^2) + (-{f_1}^3 - {f_3}^3 - {f_5}^3)u \\
&\quad +({f_4}^3)t + (-{f_2}^3 + {f_4}^3) \\
{f'}_1 &= (-{f_1}^3 - {f_3}^3 - {f_5}^3)(-t^2) + ({f_0}^3 + {f_2}^3 + {f_4}^3)u \\
&\quad +{f_3}^3 t + (-{f_1}^3 - {f_5}^3) \\
{f'}_2 &= ({f_2}^3 - {f_4}^3)(-t^2) + ({f_3}^3 - {f_5}^3)u \\
&\quad +(-{f_0}^3 - {f_2}^3 + {f_4}^3)t + ({f_2}^3) \\
{f'}_3 &= (-{f_3}^3 + {f_5}^3)(-t^2) + ({f_3}^3 - {f_5}^3)u \\
&\quad +(-{f_0}^3 - {f_2}^3 + {f_4}^3)t + (-{f_2}^3) \\
{f'}_4 &= ({f_4}^3)(-t^2) + ({f_5}^3)u + (-{f_2}^3 + {f_4}^3)t \\
&\quad +(-{f_0}^3 - {f_2}^3 + {f_4}^3) \\
{f'}_5 &= (-{f_5}^3)(-t^2) + ({f_4}^3)u + ({f_3}^3 - {f_5}^3)t \\
&\quad +({f_1}^3 - {f_3}^3 - {f_5}^3) + ({f_0}^3 + {f_3}^3 - {f_5}^3).
\end{aligned}
$$

Considering hardware sharing, the formula can be rewritten as follows:

$$
\begin{aligned}
{f'}_0 &= ({f_0}^3 + {f_2}^3 + {f_4}^3)(-t^2) + (-{f_1}^3 - {f_3}^3 - {f_5}^3)u \\
&\quad +({f_4}^3)t + (-{f_2}^3 + {f_4}^3) \\
{f'}_1 &= (({f_0}^3 + {f_2}^3 + {f_4}^3) - ({f_1}^3 + {f_3}^3 + {f_5}^3))(-t^2 + u) \\
&\quad -({f_0}^3 + {f_2}^3 + {f_4}^3)(-t^2) + ({f_1}^3 + {f_3}^3 + {f_5}^3)u \\
&\quad +({f_5}^3)t + ({f_3}^3 - {f_5}^3) \\
{f'}_2 &= ({f_2}^3 - {f_4}^3)(-t^2) + ({f_3}^3 - {f_5}^3)u \\
&\quad +(-{f_0}^3 - {f_2}^3 + {f_4}^3)t + ({f_2}^3) \\
{f'}_3 &= (({f_2}^3 - {f_4}^3) - ({f_3}^3 - {f_5}^3))(-t^2 + u) \\
&\quad -({f_2}^3 - {f_4}^3)(-t^2) + ({f_3}^3 + {f_5}^3)u \\
&\quad +({f_1}^3 + {f_3}^3 - {f_5}^3)t + ({f_3}^3) \\
{f'}_4 &= ({f_4}^3)(-t^2) + ({f_5}^3)u + (-{f_2}^3 + {f_4}^3)t \\
&\quad +(-{f_0}^3 - {f_2}^3 + {f_4}^3) \\
{f'}_5 &= (({f_4}^3) - ({f_5}^3))(-t^2 + u) - ({f_4}^3)(-t^2) + ({f_5}^3)u \\
&\quad +({f_3}^3 - {f_5}^3)t + ({f_0}^3 + {f_3}^3 - {f_5}^3).
\end{aligned}
$$

Thus, the cost of sparse multiplication is 15 multiplications, 6 cubings, and 33 additions. Despite of the sparse multiplications, there left two field multiplications, two additions, and four cubings in Miller's loop. Thus, the overall cost is 17 field multiplication, 10 cubings, and 35 additions over F_{3^m} in a single Miller's loop.

B. Final Exponentiation

In Algorithm 2, the operations involve power of $3^{3m} - 1$, $3^m + 1$, and $3^{(m+1)/2}$ in $F^*_{3^{6m}}$. We aim at reducing the field operation numbers, so that hardware complexity for final exponentiation can be alleviated. The following section depicts the idea of our implementation, and we take $m = 97$ to evaluate the detail cost of each operation to show the improvement over relative works.

1) Power of $3^{3m} - 1$ *in* $F^*_{3^{6m}}$*:* Granger *et al.* [30] introduced the torus $T_2(F_{q^3})$ for compressing value of F_{q^6}, and we used this idea to compute the power of $3^{3m} - 1$ in $F_{3^{6m}}$. Note that $F^*_{3^{6m}}$ is a second extension field of $F^*_{3^{3m}}$, thus every element in $F^*_{3^{6m}}$ can be represented as $A = A_0 + A_1\sigma$ with $A_0, A_1 \in F^*_{3^{3m}}$. Or more formally, torus $T_2(F^{3m}_3) = \{A_0 + A_1\sigma \in F^*_{3^{6m}} : (A_0)^2 + (A_1)^2 = 1\}$.

Thus, $A^{3^{3m}} = (A_0 + A_1\sigma)^{3^{3m}} = A_0^{3^{3m}} + A_1^{3^{3m}}\sigma^{3^{3m}} = A_0 - A_1\sigma$, and $A^{3^{3m}-1} = A^{3^{3m}}/A = (A_0 - A_1\sigma)/(A_0 + A_1\sigma) = (A_0 - A_1\sigma)^2/(A_0^2 + A_1^2) = (A_1^2 - A_0^2 - 2A_0A_1\sigma)/(A_0^2 + A_1^2)$. Because $(A^{3^{3m}-1})^2 = ((A_1^2 - A_0^2 - 2A_0A_1\sigma)/(A_0^2 + A_1^2))^2 = 1$, the value $A^{3^{3m}-1} \in T_2(F_{3^{3m}})$.

To compute the multiplication and squaring in $F^*_{3^{3m}}$, direct implementation will cost ten and six field multiplications, respectively. We can use Karatsuba scheme [31] to reduce the cost to six and five field multiplications, respectively. On the other hand, to calculate the inversion in $F^*_{3^{3m}}$, let $B = b_0 + b_1\rho + b_2\rho^2$ be the multiplicative inverse of $A = a_0 + a_1\rho + a_2\rho^2$. Since $AB = 1$, we obtain the following matrix:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = w^{-1} \begin{bmatrix} a_0^2 - a_1^2 + a_2^2 - a_2a_0 - a_2a_1 \\ a_2^2 - a_0a_1 \\ a_1^2 - a_2^2 - a_0a_1 \end{bmatrix}$$

where $w = (a_0 + a_1 + a_2)^3 - a_0(a_0a_2 - a_1^2) + a_2^2(a_0 - a_1)$, which involves seven field multiplications and one field inversion. If we construct the squaring from multiplication, power of $3^{3m} - 1$ in $F^*_{3^{6m}}$ requires 37 field multiplications, 71 additions, and one field inversions in F_{3^m}.

2) Power of $3^m + 1$ *in* $F^*_{3^{6m}}$*:* For powering $3^m + 1$ in $F_{3^{6m}}$, we still use the torus concept. We adopt the result in [28], which requires nine field multiplications and 18 field additions.

3) Power of $3^{(m+1)/2}$ *in* $F^*_{3^{6m}}$*:* In this section, we take advantage of Frobenius, so that the addition number can be significantly reduced. Let $F = f_0 + f_1\sigma + f_2\rho + f_3\sigma\rho + f_4\rho^2 + f_5\sigma\rho^2 \in F^*_{3^{6m}}$. Noting that $\sigma^{3^i} = (-1)^i\sigma$, $\rho^{3^i} = \rho + ib$, and $(\rho^2)^{3^i} = \rho^2 - ib\rho + i^2$. We obtain the following formula, depending on m:

$$\begin{aligned} F^{3^{(m+1)/2}} = {} & f_0^{3^{(m+1)/2}} + f_1^{3^{(m+1)/2}}\sigma^{3^{(m+1)/2}} + f_2^{3^{(m+1)/2}}\rho^{3^{(m+1)/2}} \\ & + f_3^{3^{(m+1)/2}}\sigma^{3^{(m+1)/2}}\rho^{3^{(m+1)/2}} + f_4^{3^{(m+1)/2}}(\rho^2)^{3^{(m+1)/2}} \\ & + f_5^{3^{(m+1)/2}}\sigma^{3^{(m+1)/2}}\rho^{2^{3^{(m+1)/2}}}. \end{aligned}$$

Take $m = 97$

$$\begin{aligned} F^{3^{49}} = {} & (f_0^{3^{49}} + f_2^{3^{49}} + f_4^{3^{49}}) - (f_1^{3^{49}} + f_3^{3^{49}} + f_5^{3^{49}})\sigma \\ & + (f_2^{3^{49}} - f_4^{3^{49}})\rho - (f_3^{3^{49}} + f_5^{3^{49}})\sigma\rho \\ & + f_4^{3^{49}}(\rho^2) - f_5^{3^{49}}(\sigma\rho^2). \end{aligned}$$

Thus, to complete the final exponentiation power, we requires 79 field multiplications, 390 field cubings, and 180 field additions for $m = 97$. The comparison of these computation costs with other state-of-the-art works are listed in Table II. As shown in Table II, since Ronan *et al.* [12] construct element in $F_{3^{6m}}$ using composite field representation instead of torus representation, their computation cost is markedly larger than ours. On the other hand, though Beuchat *et al.* [29] also employ torus representation, they do not take advantage of Frobenius and some mathematical manipulation, thus their computation cost is still higher than ours.

TABLE II

COMPARISON AMONG PREVIOUS WORKS IN FINAL EXPONENTIATION FOR $m = 97$

	Proposed	Beuchat et al. [29].	Ronan et al. [12]
Addition(F_{3^m})	180	477	1321
Multiplication(F_{3^m})	79	87	231
Cubing(F_{3^m})	390	390	304
Execution Cycles	804	2527	4993

IV. HARDWARE ARCHITECTURE FOR η_T PAIRING ACCELERATOR

To justify the need of building hardware accelerator in system, we first implement the η_T pairing algorithm in software in the context of pairing protocols [4] using MIRACL SDK [32], where the results are shown in Table III.

TABLE III

PROFILING OF A PAIRING PROTOCOL [4]

Operation	Time(ms)
ECSM	1.87
Bilinear Pairing	9.10
Hash from bitstream to G1	0.11
Hash from G2 to bitstream	0.08
Randomly generate an element in G1	0.31

Platform: AMD Athlon II X4 640 Processor

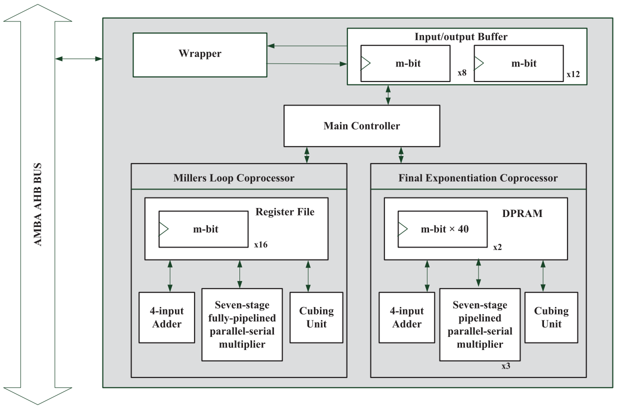


Fig. 2. Block diagram of our proposed pairing accelerator.

The result shows that when building the pairing protocol, the bilinear pairing is the most time critical part. Furthermore, the computation time of elliptic curve scalar multiplication (ECSM) takes much time as well, thus it may be beneficial to compute both bilinear pairing and ECSM using hardware accelerator. Our pairing accelerator wrapped with AMBA AHB bus can work with the ECSM accelerator in [33]–[35] to alleviate the bottleneck of pairing protocols.

Fig. 2 shows our overall proposed pairing accelerator with a standard AMBA AHB interface. For supporting arbitrary field length, the pairing arithmetic unit is designed to have programmable datapath. The η_T pairing can be calculated using pairing arithmetic unit controlled by the main controller.

The Miller's algorithm is an iterative algorithm, while final exponentiation involves irregular computation. To achieve high-speed requirement, we design two independent dedicate hardware modules for Miller's algorithm and final exponentiation, respectively. By choosing appropriate multiplier architecture and selecting adequate number of multipliers, we pipeline these two steps to enhance throughput.

TABLE IV

HARDWARE SCHEDULING OF MILLER'S LOOP

cycle	MUL IN1	MUL IN2	MUL OUT	CUBE	OUT
9	$-t^2$	$f_2 - f_4$	f_4t		X_Q
10	$-t^2$	$f_0 + f_2 + f_4$	f_2t		X_Q
11	u	f_5	f_0t		Y_Q
12	u	$f_3 - f_5$	f_5t		Y_Q
13	u	$f_1 + f_3 + f_5$	f_3t		
14	$-t^2 + u$	$f_4 - f_5$	f_1t		
15	$-t^2 + u$	$f_2 - f_4 - f_3 + f_5$	f_4t^2		
16	$-t^2 + u$	$f_0 - f_1 + f_2 - f_3 + f_4 - f_5$	$(-f_2 + f_4)t^2$		
0	t	t	$(f_0 + f_2 + f_4)(-t^2)$t		
1	Y_P	Y_Q	f_5u	f_4'	f_4'
2	t	f_4'	$(f_3 - f_5)u$	f_2'	f_2'
3	t	f_2'	$(f_1 + f_3 + f_5)u$	f_0'	f_0'
4	t	f_0'	$(f_4 - f_5)(-t^2 + u)$	f_5'	f_5'
5	t	f_5'	$(f_2 - f_4 - f_3 + f_5)(-t^2 + u)$	f_3'	f_3'
6	t	f_3'	$(f_0 - f_1 + f_2 - f_3 + f_4 - f_5)$ $(-t^2 + u)$	f_1'	f_1'
7	t	f_1'	$t \cdot t$		
8	$-t^2$	f_4'	u		
9	$-t^2$	$f_2' - f_4'$	$f_4't$		X_Q
10	$-t^2$	$f_0' + f_2' + f_4'$	$f_2't$		X_Q
11	u	f_5'	f_0't		Y_Q
12	u	$f_3' - f_5'$	$f_5't$		Y_Q
13	u	$f_1' + f_3' + f_5'$	$f_3't$		
14	$-t^2 + u$	$f_4' - f_5'$	$f_1't$		
15	$-t^2 + u$	$f_2' - f_4' - f_3' + f_5'$	$f_4't^2$		
16	$-t^2 + u$	$f_0' - f_1' + f_2' - f_3' + f_4' - f_5'$	$(-f_2' + f_4')t^2$		
⋮	⋮	⋮	⋮	⋮	⋮

A. Miller's Loop Implementation

Multiplication is the pertinent operation of a bilinear pairing computation, as stated in Section III. Variants of parallel–serial and Karatsuba multiplier are by far the most efficient two used in the context. To study a wide range of implementation strategies, we write a Verilog code generator for parallel–serial multiplier and modify the Verilog code generator in [36], which can produce variants of Karatsuba multipliers. The parallel–serial multiplier seems to be more attractive because it has smaller area overhead. However, we should carefully deal with the scheduling and manage to keep the pipeline busy. To enhance throughput while still maintaining a competitive critical path delay, we design a seven-stage parallel–serial multiplier. The coprocessor shown in Fig. 2 embeds this seven-stage fully pipelined parallel–serial multiplier, which is similar to our seven-stage pipelined parallel–serial multiplier stated in the following section except the hardware is duplicated and control is altered as well. There is a four-input adder to support the field addition, subtraction, and accumulation in Algorithm 1. A standalone cubing unit that can complete a cubing operation in one cycle is also included.

With these arithmetic units, the 17 multiplications in each iteration of Miller's loop is scheduled at higher priority, and the field additions and field cubings are parallel computed with multiplication, as shown in Table IV. In Table III, MUL IN1, MUL IN2, and MUL OUT denote the two inputs and output of the multiplier, respectively, and CUBE indicates the cycle when cubing operation is conducted. The OUT is the output of the Miller's loop coprocessor. We aim to keep the pipeline busy, so that the data dependency is avoided and the register usage is minimized. From the scheduling result, each single Miller's loop can be completed in 17 clock cycles with the proposed coprocessor, thus Algorithm 1 can be completed in $17 \cdot (m + 1)/2$ cycles including the initial step.

B. Final Exponentiation Implementation

Note that the critical operations in Algorithm 2 are multiplications and long sequences of cubings, which are different from the design of Miller's coprocessor. As a result, we propose a new module for arithmetics in final exponentiation. The computation cost of final exponentiation is smaller than Miller's loop according to Section III, thus we can deploy a slightly more serial architecture to reduce hardware cost.

Inputs and outputs, as well as intermediate results, are stored in a dual-port random access memory since the hardware scheduling involves simultaneously read to the register file. The coprocessor embeds three seven-stage pipelined parallel–serial multipliers, as shown in Fig. 3, to match the computation

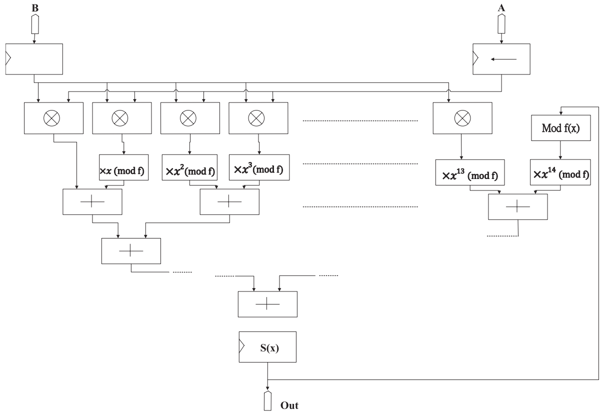


Fig. 3. Seven-stage pipelined parallel–serial multiplier.

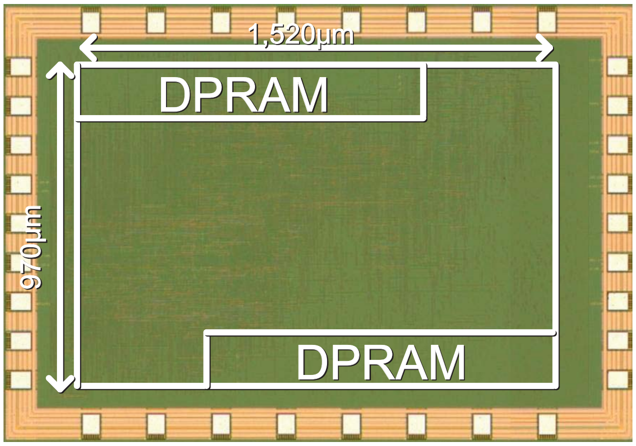


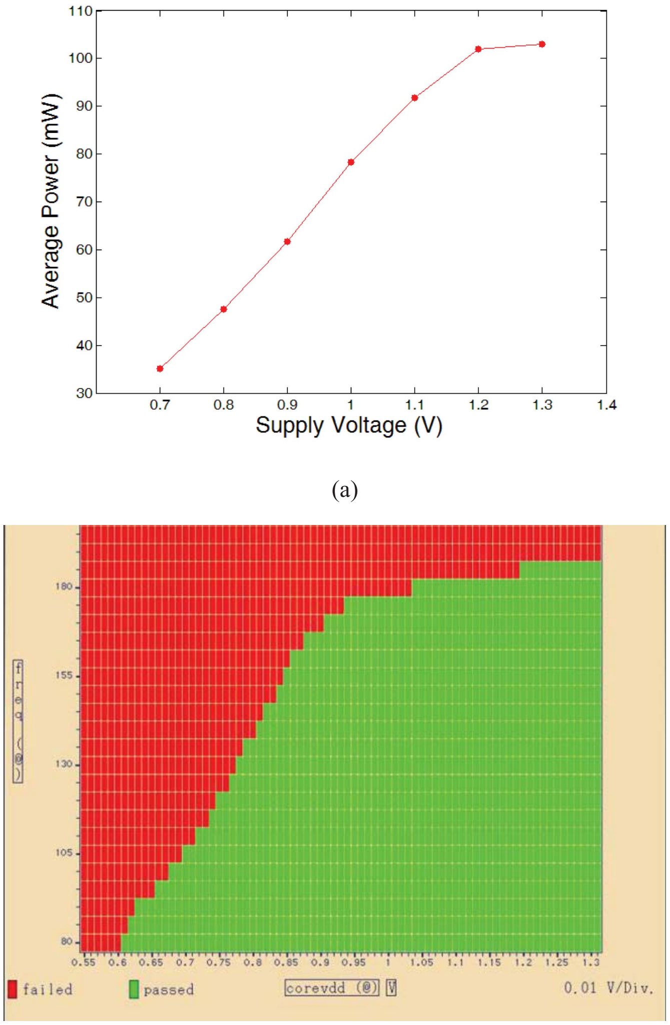
Fig. 4. Chip micrograph of our proposed pairing accelerator.

time of Miller's loop. Note that in Algorithm 3 the operation $\times x^D$ involves only wiring, and a single modulo $f(x)$ reduction is needed in each iteration. As previously mentioned, since trinomial is chosen as the irreducible polynomial, reduction can be done by performing several additions. We also embed a four-input adder and a cubing unit to deal with addition, subtraction, and cubing operations. This architecture allows us to efficiently implement the final exponentiation algorithm, and the cycle count compared with previous design is listed in Table II.

Finally, Miller's loop and the final exponentiation are computed in a pipelined manner in our design, and we can learn that the computation time of these two steps are nearly equal. Therefore, the throughput of this pairing accelerator almost doubles the throughput of nonpipelined pairing accelerator.

V. Implementation Results

With our proposed pairing arithmetic, hardware scheduling, pipelining multiplier architecture, and parallelism scheme, a test chip of the η_T pairing accelerator is fabricated in 90-nm 1P9M CMOS process. Due to the limitation of chip size and for the comparison purpose, the field $F(3^{97})$ is selected. The chip micrograph is shown in Fig. 4. In this section, we describe the measurement results, summarize the key characteristics, and compare our application specified integrated circuit (ASIC) implementation results to other state-of-the-art designs.



(a)

(b)

Fig. 5. Measurement results. (a) Measured power consumption under different supply voltages. (b) Shmoo plot.

A. Measurement Results

The measurement result of the test chip under different supply voltages is shown in Fig. 5(a). The results shows that the accelerator draws 103 mW under 1.3 V supply voltage while running at 185 MHz. Since the clock cycle is $17 \times (97+1)/2$ for the Miller's loop, the throughput is thus about 222K bilinear pairings per second. When supply voltage is scaled down to 0.7 V, the power is reduced to 35.1 mW with a better energy efficiency. The Shmoo plot is shown in Fig. 5(b). Table V summarizes key features of the proposed pairing accelerator over $F(3^{97})$ test chip under 1.0 V supply voltage.

B. Comparison With Other η_T Pairing Accelerator

In considering the scaling effect of fabrication technology and supply voltage, the normalization factor of area–time (AT)

TABLE V

CHIP SUMMARY

Process	UMC 90nm 1P9M
Curve	$E/F_{3^{97}}$
Gate Counts	336K
Memory Area	$(0.102mm^2)\times$ 2
Chip Area	$1.47mm^2$
Max. Clock Frequency	175MHz
Time	$4.76\mu s$
Average Power	$78.36mW$, 1.0V

TABLE VI

COMPARISON WITH OTHER η_T PAIRING ACCELERATOR

	This work	Beuchat et al. [13]
Curve	$E/F_{3^{97}}$	$E/F_{3^{97}}$
Technology	90nm	180nm
Gate Counts	336K	193.7K
Core Size	$1.52mm \times 0.97\ mm$	$3.85mm \times 3.85\ mm$
Memory Area	$(0.102mm^2)\times$ 2	FPGA embedded memory
Frequency(MHz)	175	200
Operation Time(μs)	4.76	45.9
AT	1	5.56 (2.78[a])
Energy(μJ/Pairing)	0.37	31.37 (2.51[b])

AT product = gate count × time

Energy = average power × time

[a] Normalization factor is 0.50 $(90nm/180nm)$

[b] Normalization factor is 0.08 $(90nm/180nm)^2 \times (1.0V/1.8V)^2$

product and energy can be referred to [40] and [41]. The normalization factor for AT product is proportional to the ratio of minimum gate length for transistor; the normalization factor for energy is proportional to the square ratio of minimum gate length for transistor multiplied by square ratio of supply voltage.

To the best of our knowledge, Beuchat *et al.* [13] design the first and the only ASIC implementation of η_T pairing, and their ASIC implementations embeds nine multipliers, an addition unit, and a cubing unit. However, the computation cost of their methodology is higher than ours. Moreover, they do not employ the pipeline techniques, and the computation time of the two steps is far from balance. According to Table VI, our design achieves 178% better in terms of technology AT product.

C. Comparison With Other Ate Pairing Accelerator

The comparison among different pairing accelerator is especially difficult since the implementation style and design parameters are distinct from each other.

According to NIST recommendation, 128-bit symmetric security is essential beyond 2030 [26]. Therefore, various implementations of pairing accelerator targeting at 128-bit security level are presented in recent years. A comparison with three state-of-the-art pairing accelerator designs is given in Table VII. Note that to make a fair comparison, we use the widely accepted attack model proposed in [42] and choose the curve parameters derived in Section II. Such that η_T pairing over supersingular curve with curve parameter $E/F_{3^{709}}$ and pairing over ordinary curve with curve parameter E/F_{256} are in roughly the same security level.

TABLE VII

COMPARISON WITH OTHER PAIRING ACCELERATOR

	This work	Fan et al. [37]	Kammler [38]	Li et al. [39]
Curve	$E/F_{3^{709}}$	E/F_{256}	E/F_{256}	E/F_{256}
Security (bit)	126	128	128	126
Technology	90nm	130nm	130nm	65nm
Area	5950K	183K	97K	319K
Frequency(MHz)	166	204	338	800
Operation Time(μs)	36.2	2910	15800	640
AT	1	2.47(1.71[a])	7.12(4.93[a])	0.95(1.31[b])
Energy(μJ/Pairing)	140.09	-	-	170.56 (226.8[c])

AT product = gate count × time

Energy = average power × time

[a] Normalization factor is 0.69 $(90nm/130nm)$

[b] Normalization factor is 1.38 $(90nm/65nm)$

[c] Normalization factor is 1.33 $(90nm/65nm)^2 \times (1.0V/1.2V)^2$

The first ASIC implementations of pairings with 128 bits of security were presented in [37] and [38], and [39] describes a more efficient accelerator by modifying the multiplication scheme. These three implementations use BN-curves so as to exploit their optimal embedding degree $k = 12$ while targeting 128 bits of security, but the arithmetic on these curves are over prime field, which should deal with carry propagation carefully. In addition, pairings over BN-curves are asymmetric pairing, which is not well suited for many existing protocols. We take advantage of the efficient arithmetic of ternary field, and further minimize the complexity by mathematical formulation and exploration of hardware-efficient multiplier. From the comparison table, we can observe that our design outperforms other designs in terms of operation time. Furthermore, the gate counts of our design are larger than the others due to the fully pipelining and parallelism techniques, but our design still achieves better AT product compared with these three designs.

VI. CONCLUSION

We have presented an efficient pairing accelerator supporting η_T pairing computation over characteristic three that targets high throughput and hardware efficiency. The test chip with 1.47-mm^2 core area is fabricated in 90-cm CMOS 1P9M technology. It achieves one bilinear pairing computation over $F(3^{97})$ in 4.76 μs at 175 MHz. Moreover, the 126-bit security level version of our design can perform a bilinear pairing computation over $F(3^{709})$ in 36.2 μs at 166 MHz. The performance comparison shows that our architecture proposal outperforms other related pairing designs in both operation time and hardware efficiency. These benefits demonstrate that our proposal solution is well suitable for high-speed pairing-based protocols in cloud services and applications.

ACKNOWLEDGMENT

The authors would like to thank Dr. C.-C. Chung for his technical support in chip implementation.

References

[1] I. Blake, G. Seroussi, N. Smart, and J. W. S. Cassels, *Advances in Elliptic Curve Cryptography* (London Mathematical Society Lecture Note Series). New York, NY, USA: Cambridge Univ. Press, 2005.

[2] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *Proc. 21st Annu. Int. Cryptol. Conf. Adv. Cryptol.*, Aug. 2001, pp. 213–229.

[3] A. Sahai and B. Waters, "Fuzzy identity based encryption," in *IACR*, New York, NY, USA: Springer-Verlag, 2004.

[4] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. EUROCRYPT*, 2004, pp. 506–522.

[5] (2014). *CertiVox* [Online]. Available: https://certivox.com/

[6] *IEEE P1363.3 Draft Standard for Identity-Based Public-Key Cryptography Using Pairing*, IEEE Standard P1363.3, Apr. 2006.

[7] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient algorithms for pairing-based cryptosystems," in *Proc. 22nd Annu. Int. Cryptol. Conf. Adv. Cryptol.*, 2002, pp. 354–368.

[8] S. D. Galbraith, K. Harrison, and D. Soldera, "Implementing the Tate pairing," in *Proc. 5th Int. Symp. Algorithmic Number Theory*, 2002, pp. 324–337.

[9] P. S. L. M. Barreto, S. D. Galbraith, C. Ó'hÉigeartaigh, and M. Scott, "Efficient pairing computation on supersingular Abelian varieties," *Des., Codes Cryptogr.*, vol. 42, no. 3, pp. 239–271, 2007.

[10] F. Hess, N. Smart, and F. Vercauteren, "The Eta pairing revisited," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4595–4602, Oct. 2006.

[11] J.-L. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto, M. Shirase, and T. Takagi, "Algorithms and arithmetic operators for computing the ηT pairing in characteristic three," *IEEE Trans. Comput.*, vol. 57, no. 11, pp. 1454–1468, Nov. 2008.

[12] R. Ronan, C. O'Eigeartaigh, C. Murphy, T. Kerins, and P. S. L. M. Barreto, "Hardware implementation of the ηT pairing in characteristic 3," *IACR Cryptol. ePrint Archive*, 2006, p. 371.

[13] J.-L. Beuchat, H. Doi, K. Fujita, A. Inomata, P. Ith, and A. Kanaoka, "FPGA and ASIC implementations of the eta_t pairing in characteristic three," *Comput. Electr. Eng.*, vol. 36, no. 1, pp. 73–87, 2010.

[14] N. Shinohara, T. Shimoyama, T. Hayashi, and T. Takagi, "Key length estimation of pairing-based cryptosystems using η_t pairing," in *Proc. 8th Int. Conf. Inf. Security Pract. Exper.*, 2012, pp. 228–244.

[15] V. S. Miller, "Short programs for functions on curves," Ph.D. dissertation, Exploratory Comput. Sci., IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA, 1986.

[16] M. Scott, *Implementing Cryptographic Pairings* (Lecture Notes in Computer Science). Berlin, Germany: Springer-Verlag, 2007, pp. 177–196.

[17] O. Ahmadi and F. Rodriguez-Henriquez, "Low complexity cubing and cube root computation over F_{3^m} in polynomial basis," *IEEE Trans. Comput.*, vol. 59, no. 10, pp. 1297–1308, Oct. 2010.

[18] (2012). *PARI/GP, version 2.5.2*, The PARI Group, Bordeaux, France [Online]. Available: http://pari.math.u-bordeaux.fr/

[19] K. Harrison, D. Page, and N. P. Smart, "Software implementation of finite fields of characteristic three, for use in pairingbased cryptosystems," *LMS J. Comput. Math.*, vol. 5, pp. 181–193, Nov. 2002.

[20] Y. Kawahara, K. Aoki, and T. Takagi, "Faster implementation of ηT pairing over GF(3^{m}) using minimum number of logical instructions for GF(3)-addition," in *Pairing-Based Cryptography* (Lecture Notes in Computer Science). Berlin, Germany: Springer-Verlag, 2008, pp. 282–296.

[21] L. Song and K. K. Parhi, "Low-energy digit-serial/parallel finite field multipliers," *J. VLSI Signal Process. Syst. Signal, Image Video Technol.*, vol. 19, no. 2, pp. 149–166, Jul. 1998.

[22] T. Itoh and S. Tsujii, "A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases," *Inf. Comput.*, vol. 78, no. 3, pp. 171–177, Sep. 1988.

[23] J. M. Pollard, "English Monte Carlo methods for index computation (mod p)," *English Math. Comput.*, vol. 32, no. 143, pp. 918–924, Jul. 1978.

[24] L. M. Adleman and M.-D. A. Huang, "Function field sieve method for discrete logarithms over finite fields," *Inf. Comput.*, vol. 151, nos. 1–2, pp. 5–16, 1999.

[25] J.-L. Beuchat, E. López-Trejo, L. Martínez-Ramos, S. Mitsunari, and F. Rodríguez-Henríquez, "Multi-core implementation of the Tate pairing over supersingular elliptic curves," in *Proc. 8th Int. Conf. Cryptol. Netw. Security*, 2009, pp. 413–432.

[26] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "Recommendation for key management—Part 1: General (revised)," NIST, Boulder, CO, USA, Tech. Rep. SP800-57, Mar. 2007.

[27] G. Bertoni, L. Breveglieri, P. Fragneto, and G. Pelosi, "Parallel hardware architectures for the cryptographic Tate pairing," in *Proc. 3rd Int. Conf. Inf. Technol., New Generat.*, Apr. 2006, pp. 186–191.

[28] E. Gorla, C. Puttmann, and J. Shokrollahi, "Explicit formulas for efficient multiplication in $F_{3^{6m}}$," in *Proc. 14th Int. Conf. Sel. Areas Cryptography*, 2007, pp. 173–183.

[29] J.-L. Beuchat, N. Brisebarre, M. Shirase, T. Takagi, and E. Okamoto, "A coprocessor for the final exponentiation of the η_t pairing in characteristic three," in *Proc. 1st Int. Workshop Arithmetic Finite Fields*, 2007, pp. 25–39.

[30] R. Granger, D. Page, and M. Stam, "On small characteristic algebraic tori in pairing-based cryptography," *LMS J. Comput. Math.*, vol. 9, pp. 64–85, Mar. 2004.

[31] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on automata," *Soviet Phys. Doklady*, vol. 7, pp. 595–596, Jan. 1963.

[32] (2013). *MIRACL SDK, Version 5.5 CertiVox* [Online]. Available: http://www.certivox.com/miracl/

[33] S.-C. Chung, J.-W. Lee, H.-C. Chang, and C.-Y. Lee, "A high-performance elliptic curve cryptographic processor over GF(p) with SPA resistance," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2012, pp. 1456–1459.

[34] J.-W. Lee, S.-C. Chung, H.-C. Chang, and C.-Y. Lee, "Processor with side-channel attack resistance," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2013, pp. 50–51.

[35] J.-Y. Lai and C.-T. Huang, "Elixir: High-throughput cost-effective dual-field processors and the design framework for elliptic curve cryptography," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 11, pp. 1567–1580, Nov. 2008.

[36] C. Rebeiro and D. Mukhopadhyay, "High speed compact elliptic curve cryptoprocessor for fpga platforms," in *Proc. 9th Int. Conf. Cryptol. India, Progr. Cryptol.*, 2008, pp. 376–388.

[37] J. Fan, F. Vercauteren, and I. Verbauwhede, "Faster arithmetic for cryptographic pairings on barreto-naehrig curves," in *Proc. 11th Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2009, pp. 240–253.

[38] D. Kammler, D. Zhang, P. Schwabe, H. Scharwaechter, M. Langenberg, D. Auras, *et al.*, "Designing an ASIP for cryptographic pairings over barreto-naehrig curves," in *Proc. 11th Int. Workshop Cryptograph. Hardw. Embedded Syst.*, 2009, pp. 254–271.

[39] Y. Li, J. Han, S. Wang, D. Fang, and X. Zeng, "An 800Mhz cryptographic pairing processor in 65nm CMOS," in *Proc. IEEE A-SSCC*, Nov. 2012, pp. 217–220.

[40] H.-Y. Hsu, A.-Y. Wu, and J.-C. Yeo, "Area-efficient VLSI design of Reed–Solomon decoder for 10GBase-LX4 optical communication systems," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 53, no. 11, pp. 1245–1249, Nov. 2006.

[41] C.-C. Wong and H.-C. Chang, "High-efficiency processing schedule for parallel turbo decoders using QPP interleaver," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 6, pp. 1412–1420, Jun. 2011.

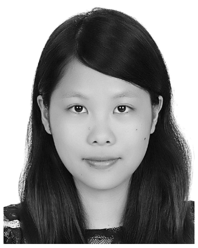
[42] A. K. Lenstra, "Unbelievable security. Matching AES security using public key systems," in *ASIACRYPT*, C. Boyd, Ed. New York, NY, USA: Springer-Verlag, 2001, pp. 67–86.

Szu-Chi Chung (S'14) received the B.S. degree in electrical engineering and computer science from National Chiao-Tung University, Hsinchu, Taiwan, in 2011, where he is currently pursuing the Ph.D. degree in electronics engineering.

His current research interests include VLSI implementation of security systems.

Jing-Yu Wu received the B.S. and M.S. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 2009 and 2011, respectively.

She is currently with MediaTek Inc., as a Digital Hardware Engineer. Her current research interests include algorithms and VLSI implementation of cryptographic systems.

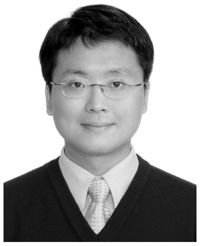


Hsing-Ping Fu Hsing- Ping Fu received the B.S. and M.S. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 2010 and 2012, respectively.

Her current research interests include advanced encryption standard hardware and attacks.

Jen-Wei Lee (S'14) received the B.S. and Ph.D. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 2007 and 2013, respectively.

He was with MediaTek, Inc., ASIC and Communication Systems Group, Hsinchu, where he is currently a Senior Engineer. His current research interests include digital video broadcasting system-on-ship design for high-performance receiver.



Hsie-Chia Chang (M'14) received the B.S., M.S., and Ph.D. degrees in electronics engineering from National Chiao-Tung University, Hsinchu, Taiwan, in 1995, 1997, and 2002, respectively.

He was with OSP/DE1, MediaTek, Inc., Hsinchu, from 2002 to 2003, where he was involved in decoding architectures for combo single chip. In 2003, he joined the faculty of the Electronics Engineering Department, National Chiao-Tung University, where he has been a Professor since 2010. His current research interests include algorithms and VLSI architectures in signal processing, in particular for error control codes and cryptosystems.

Dr. Chang has served as the Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS since 2012. He served as a Technique Program Committee Member for the IEEE Asian Solid-State Circuits Conference from 2011 to 2013. He received the Outstanding Youth Electrical Engineer Award from the Chinese Institute of Electrical Engineering in 2010 and the Outstanding Youth Researcher Award from Taiwan IC Design Society in 2011.

Chen-Yi Lee (M'14) received the B.S. degree in electrical engineering from National Chiao-Tung University, Hsinchu, Taiwan, and the M.S. and Ph.D. degrees in electrical engineering from Katholieke University Leuven, Leuven, Belgium, in 1982, 1986, and 1990, respectively.

He was with the Interuniversity Microelectronics Centre, Leuven, or VLSI Systems and Design Methodology (VSDM) division of the Interuniversity Micro Electronics Center (IMEC), from 1986 to 1990, where he was involved in architecture synthesis for digital signal processing (DSP). In 1991, he joined the faculty of the Electronics Engineering Department, National Chiao-Tung University, where he is currently a Professor. He is active in various aspects of short-range wireless communications, system-on-chip design technology, very low-power designs, and multimedia signal processing. He has published more than 200 journal/conference papers and holds more than 25 R.O.C./U.S. patents. His current research interests include VLSI algorithms and architectures for high-throughput and energy-efficient DSP applications.

Dr. Lee served as the Director with Chip Implementation Center, Hsinchu, from 2000 to 2003. He was the former IEEE CAS Taipei Chapter Chair from 2000 to 2001, the SIP Task Leader of National System-on-Chip Research Program from 2003 to 2005, and the Microelectronics Program Coordinator of Engineering Division under the National Science Council of Taiwan from 2003 to 2005.