Compact implementations of pairings

Anthony Van Herrewege

***Abstract*—The recent discovery of the constructive use of pairings in cryptography has opened up a wealth of new research options into identity-based encryption. In this paper, we will investigate the possible use of pairings in constrained environments. The focus will be on an small, energy efficient ASIC implementation of an accelerator for the Tate pairing over a supersingular curve.**

The results are encouraging for further research. It is possible to obtain an implementation of less than 30k gates consuming only 206 nA. Furthermore, energy efficiency improvements over twenty times compared to other published designs are possible.

***Index Terms*—Identity-based cryptography, elliptic curve cryptography, Tate pairing, hardware accelerator, ASIC.**

I. Introduction

EVER since Shamir's proposal [1] in '84, there's been an interest in identity-based cryptography. Particularly Boneh and Franklin's [2] discovery of the constructive use of pairings for identity-based encryption has helped spur on new research into possible applications and implementations.

Multitudes of protocols have seen the light, however, until recently the lack of efficient hardware accelerators for the computationally expensive pairings was always kind of a show-stopper towards implementing them. Thus most of the published implementations have a focus on speed. Implementations for area- and/or power-constrained devices were either deemed infeasible or just not interesting enough.

In 2007 Oliveira *et al.* introduced their TinyTate [3] implementation to the world. 2008 saw the light of TinyPBC [4] and NanoECC [5] from the same authors. All three papers present implementations of pairings (either the Tate or η_T) on the AT128Mega microchip of a Mica node [6], designed for deeply embedded networks. Thus it was proven that pairings were indeed feasible for use in constrained environments, such as sensor networks.

In this paper, we will investigate the feasibility of a hardware accelerator for the Tate pairing in constrained environments. In Section II necessary parameters will be defined and we will take a look at the pairing arithmetic. Section III consist of a concise overview of the implementation's hardware. Finally, results from synthesis to an ASIC implementation will be presented along with comparisons to existing implementations in Section IV. From these a conclusion will be drawn in Section V.

II. Parameters and arithmetic for the Tate pairing

In this section we will define all the parameters necessary to completely define the Tate pairing calculation. We will also present the algorithm necessary for the calculation and clarify some of the necessary arithmetic.

It should be mentioned here that, recently, variants on the Tate pairing have been published, such as the η_T [7] and Ate [8] pairings. However, seeing as they are very recent discoveries, we felt it more appropriate to focus on the better known Tate pairing.

A. Definition of the Tate pairing

The Tate pairing $e(P,Q)$ is defined as a mapping from two additive groups $\mathbb{G}_1, \mathbb{G}_2$ to a multiplicative group $\mathbb{G}_T$. To be suitable for use in cryptography, it should have the following three properties:

- Well-defined:
$$e(\mathcal{O},Q) = 1 \,\forall\, Q \in \mathbb{G}_2$$
$$e(P,\mathcal{O}) = 1 \,\forall\, P \in \mathbb{G}_1.$$
- Non-degenerate:
$$\forall\, P \in \mathbb{G}_1,\ \exists\, Q \in \mathbb{G}_2 \text{ for which } e(P,Q) \neq 1.$$
- Bilinear: $\forall\, P_1, P_2, P \in \mathbb{G}_1$ and $\forall\, Q_1, Q_2, Q \in \mathbb{G}_2$:
$$e(P_1 + P_2, Q) \equiv e(P_1, Q) \cdot e(P_2, Q)$$
$$e(P, Q_1 + Q_2) \equiv e(P, Q_1) \cdot e(P, Q_2).$$

The point $\mathcal{O}$ is the point at infinity on the elliptic curve E over which the pairing is defined.

Instead of calculating the Tate pairing as proposed by Miller in '86 [9], we will be using an optimized version of Miller's algorithm as proposed by Barreto *et al.* [10]. The Tate pairing is then a mapping:

$$\hat{e}(P,Q) : E(\mathbb{F}_q)[l] \times E(\mathbb{F}_q)[l] \mapsto \mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^l.$$

The notation $E(\mathbb{F}_q)[l]$ meaning the group of points $P \in E(\mathbb{F}_q)$ for which $lP = \mathcal{O}$. The result of the pairing is an element of the equivalence group $\mathbb{F}^*_{q^k}/(\mathbb{F}^*_{q^k})^l$, in which two elements $a \equiv b$ iif $a = bc^l$ with $c \in \mathbb{F}^*_{q^k}$. To eliminate this ambiguity, we will elevate the result of the pairing to the power $\frac{q^k-1}{l}$, the result of which will be an lth root of unity.

B. Parameters

Before we can take a look at the arithmetic behind the Tate pairing computation, some parameters need to be set. First and foremost, the elliptic curve and the field over which it is defined need to be defined. Due to the simplicity of its arithmetic (only XORing), we choose a field $\mathbb{F}_{2^m}$. We are then forced to use the curve [10]:

$$E : y^3 + y = x^3 + x + b,$$

Anthony Van Herrewege obtained his B.S. in computer engineering from K.U. Leuven, Leuven, Belgium in 2007. This paper is part of his thesis towards obtaining a M.Sc. in electrical engineering from the same university. Email: anthonyvh@gmail.com.

with $b \in \{0, 1\}$. We also define [11]:

$$\delta = \begin{cases} b & m \equiv 1, 7 \pmod 8 \\ 1 - b & m \equiv 3, 5 \pmod 8 \end{cases}$$
$$\nu = (-1)^{\delta}$$

The value of b is set to whatever value maximizes the order of the curve:

$$\#E(\mathbb{F}_{2^m}) = 2^m + \nu\sqrt{2^{m+1}} + 1.$$

So, before the value of b can be decided on, m is to be set. We also define $l = \#E$.

Considering that the final implementation should be as small as possible, we settle on $m = 163$, which, according to [12], is equivalent in strength to RSA(652). Although this is not very strong, if necessary, the hardware which will be proposed in Section III can easily be adapted to larger fields. From [13] the reduction polynomial is chosen to be

$$R = z^{163} + z^7 + z^6 + z^3 + 1.$$

Now that these parameters have been set, we can see that b needs to equal one.

The type of supersingular curve that's being used has an embedding degree $k = 4$. The result of the Tate pairing will thus be an element in $\mathbb{F}^*_{2^{4m}}$. We define this field by means of tower extensions [14]:

$$\mathbb{F}_{2^{2m}} \cong \mathbb{F}_{2^m}[x]/\left(x^2 + x + 1\right)$$
$$\mathbb{F}_{2^{4m}} \cong \mathbb{F}_{2^{2m}}[y]/\left(y^2 + (x + 1)y + 1\right)$$

Last, but not least, we need a distortion map ϕ:

$$\phi(Q) : (x_Q, y_Q) \mapsto (x_Q + s^2, y_Q + x_Q s + t^6).$$

The parameters $s, t \in \mathbb{F}_{2^{km}}$ need to be a solution to:

$$\begin{cases} s^4 + s = 0 \\ t^2 + t + s^6 + s^2 = 0. \end{cases}$$

One possible solution is:

$$\begin{cases} s = x + 1 \\ t = xy. \end{cases}$$

C. Arithmetic

Miller's algorithm as modified by Barreto *et al.* is listed in Algorithm 1. The notation $G_{A,B}(S)$ signifies the evaluation of the point S in the equation for the line though the points U and V.

The formula's for the double and the add step were taken from [14]. Those for the double step (lines 5 and 6) are:

$$\begin{cases} \lambda & = x_V^2 + 1 \\ x_{2V} & = \lambda^2 \\ y_{2V} & = \lambda \cdot (x_{2V} + x_V) + y_V + 1 \\ G_{V,V}(\phi(Q)) & = \lambda \cdot (x_\phi + x_V) + (y_\phi + y_V), \end{cases}$$

Algorithm 1 Optimized Miller's algorithm [10]

Require: $l \in \mathbb{Z}$; $P, Q \in E(\mathbb{F}_{2^m})[l]$
Ensure: $F = \hat{e}(P, Q) \in \mathbb{F}^*_{2^{km}}$
1: $t \leftarrow \lfloor \log_2(l) \rfloor$
2: $F \leftarrow 1$
3: $V \leftarrow P$
4: **for** $i = t - 1$ to 0 **do**
5: $\quad F \leftarrow F^2 \cdot G_{V,V}(\phi(Q))$
6: $\quad V \leftarrow 2 \cdot V$
7: $\quad$ **if** $l_i = 1$ and $i \neq 0$ **then**
8: $\quad\quad F \leftarrow F \cdot G_{V,P}(\phi(Q))$
9: $\quad\quad V \leftarrow V + P$
10: $\quad$ **end if**
11: **end for**
12: $F \leftarrow F^{\frac{2^{km}-1}{l}}$
13: **return** F

and those for the add step (lines 8 and 9):

$$\begin{cases} \lambda & = \frac{y_V + y_P}{x_V + x_P} \\ x_{V+P} & = \lambda^2 + x_V + x_P \\ y_{V+P} & = \lambda \cdot (x_{V+P} + x_P) + y_P + 1 \\ G_{V,P}(\phi(Q)) & = \lambda \cdot (x_\phi + x_P) + (y_\phi + y_P). \end{cases}$$

The add step only needs to be executed once, in this case, since

$$l = \#E = 2^{163} + 2^{82} + 1.$$

The division in the add step is calculated with an inversion, which is calculated with Fermat's little theorem. One inversion takes 9 multiplications and 162 squarings.

The final exponentiation F^M is split up as in [11]:

$$\begin{aligned} M &= \frac{2^{4m} - 1}{l} \\ &= \frac{(2^{2m} + 1)(2^{2m} - 1)}{l} \\ &= (2^{2m} - 1)(2^m - 2^{\frac{m+1}{2}} + 1) \\ &= (2^{2m} - 1)(2^m + 1) + (1 - 2^{2m})2^{\frac{m+1}{2}} \end{aligned}$$

III. Hardware implementation

In this section we propose a hardware architecture can execute Miller's algorithm as it was shown in the previous section. The circuit should be both compact and energy efficient, so the focus will not be on speed. In the next section this design will be synthesized as an ASIC implementation.

A. Restrictions

Since the design will be synthesized using a 0.13 μm low leakage library by Faraday Corporation [15], we first take a look at which cells take up the largest area with that technology. A listing is given in Table I. It's clear that the usage of both flip-flops (registers) and multiplexors should be kept to a minimum.

Table I
AREA OF CELLS IN AN ASIC CIRCUIT (0.13 μm LOW LEAKAGE TECHNOLOGY BY FARADAY CORPORATION [15])

Cell	Area $\left[\frac{\text{gate}}{\text{bit}}\right]$
D flip-flop (reset)	6
D flip-flop (no reset)	5,5
D latch	4,25
3 input MUX	4
2 input XOR	3,75
2 input MUX	2,25
2 input NAND	1
NOT	0,75

B. Arithmetic core

The core of the implementation is the MALU [16], [17], which can calculate the sum of two elements $a, b \in \mathbb{F}_{2^m}$ as well as a modulo operation. Its design is shown in Fig. 1. For the given parameters we need 167 XOR gates to construct this circuit.

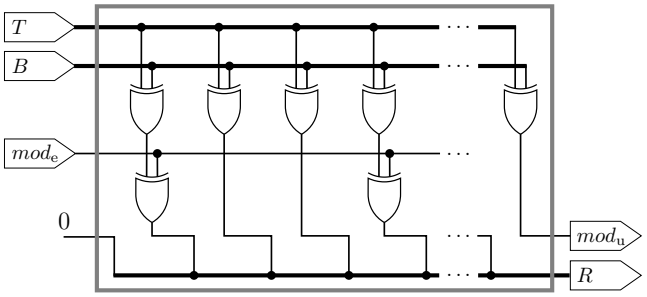

Figure 1. Arithmetic core for addition and modular reduction in $\mathbb{F}_{2^m}$

Building on this, we construct a wrapper that allows multiplication in $\mathbb{F}_{2^m}$ as well. The result of a multiplication is calculated using the 'shift-and-add' technique. To be able to do this, we need one extra register T to store temporary results. The circuit and its control logic is shown in respectively Fig. 2 and Fig. 3. Note that when executing a multiplication, the input A needs to be shifted to the left by one bit every clock cycle. This task is left up to whatever circuit implements this one.

Instead of one MALU, multiple ones can be daisy-chained together to speed up the calculation of a multiplication. When doing this, one should pick a number of MALUs d for which $m \bmod d$ equals one. That way, the result of both an addition and a multiplication will be present at the same MALU's output and no extra multiplexers need to be added to the circuit.

C. Controller for Miller's algorithm

Finally, we create a controller that contains memory and the aforementioned circuit. The general design is shown in Fig. 4. The *next* input is used to signal that there's either a new coordinate available at the input or that the next part of the result should be put on the output.

Before we can correctly asses the merits of various memory designs, we need to know how many registers we need. After writing out every calculation in Miller's algorithm, the

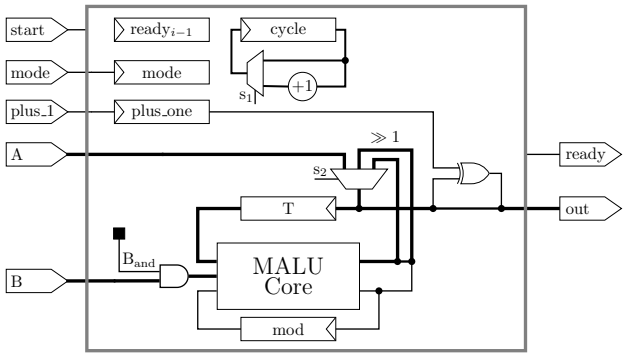

Figure 2. $\mathbb{F}_{2^m}$ arithmetic wrapper circuit

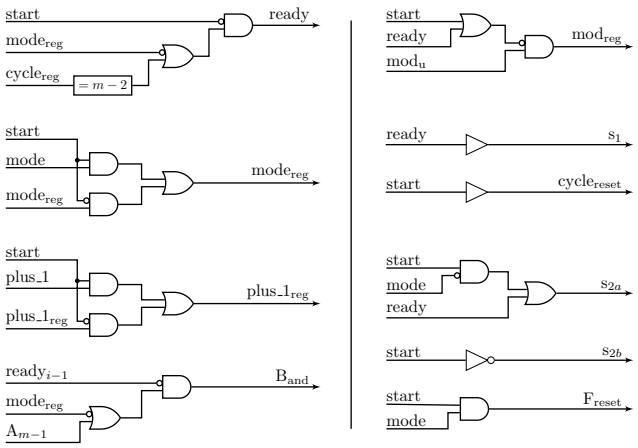

Figure 3. $\mathbb{F}_{2^m}$ arithmetic wrapper circuit - Control logic

minimum number of registers was found to be fifteen registers. This excludes the one register in the $\mathbb{F}_{2^m}$ wrapper.

The basis for the memory is a compact design by Lee and Verbauwhede [18]. They propose a unidirectional circular shift register file in which the first two registers are connected to the arithmetic core. Furthermore, it's possible to swap the contents of the first two registers. While this design is certainly about as small as it gets, using it in a register file with fifteen registers is not really feasible for a low power design. We will try to prove this in the next few paragraphs by calculating the average number of write operations $\overline{w}$ that have to be executed before every arithmetic operation. This number is directly proportional to the energy consumed.

The numbers that follow are coarse estimates and should be interpreted as such. Assume the size of the register file is n. First, calculate the average distance $\overline{r}$ between two registers by dividing the sum of possible distances s by the number of possible combinations c:

$$s = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} (j - i - 1) \qquad c = \sum_{i=0}^{n-1} i$$
$$= \frac{n \cdot (n-1) \cdot (n-2)}{6} \qquad = n \cdot \frac{n-1}{2},$$

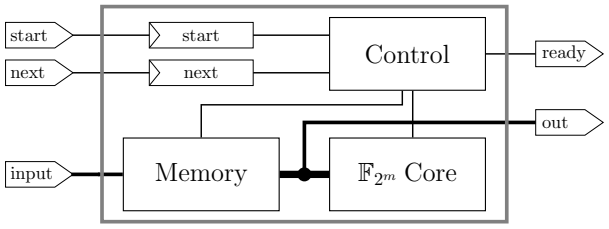

Figure 4. Controller for Miller's algorithm

thus:

$$\overline{r} = \frac{n-2}{3}.$$

Now the average number of write operations $\overline{w}$, which have to be executed before every arithmetic operation, can be calculated. First, calculate the average number of cycles it takes to move the content of two registers to register one and two. This is equal to $\overline{r}$ times n. Multiplying this by the number of writes that have to be executed each cycle, gives us $\overline{w}$. Since register contents can only be shifted in one direction, every cycle n shift operations will have to be executed, demanding n write operations. The result is

$$\overline{w} = O(n^3).$$

Now we will calculate $\overline{w}$ for a register file in which shifts in both directions are possible. Again, we first calculate:

$$\begin{aligned}\overline{r} &= \frac{1}{n} \cdot \sum_{i=1}^{n} \mathsf{min}(j-1, n-j+1) \\ &= \frac{n-1}{4}.\end{aligned}$$

In this case, the content of non-adjacent registers can be swapped independently. Thus, the average number of clock cycles will be equal to the average distance to the start of the register file: $\frac{n}{2}$. Every swap operation requires two write operations and there's two values to be moved to the front of the register file. With all this in mind, we find

$$\overline{w} = O(n).$$

Even though the resulting register file will be larger due to the addition of muxes, we favor it due to its lower energy consumption. A diagram of the design is shown in Fig. 5.

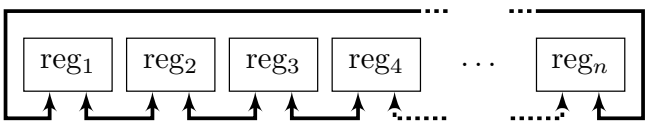

Figure 5. Register file design

Notice that since the first register is connected to the arithmetic core's input A, it needs to be able to store its own value shifted to the left. Thus register one will require a larger mux.

Using this register file design, the FSM to control the circuit consists of 553 states. Most of these are due to register contents having to be swapped around.

D. Optimizations

Since the arithmetic core is already very small, we will focus our optimization efforts on the register file.

First of all, the reset inputs of as many registers as possible are removed. As can be seen in Table I, this will save 8.5% area compared to a register with resets.

Clock gating is implemented for every register using the circuit shown in Fig. 6. Compared to a clock gating circuit which ANDs the clock and enable signal together, this one has the benefit of keeping the clock input high while idle. As shown in [19] this reduces power consumption. It can be argued though that, for this register file design, power saving improvements will probably be negligible.

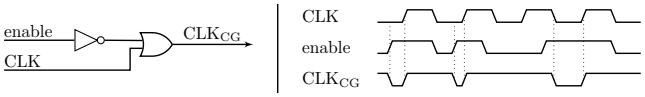

Figure 6. Clock gating circuit

IV. Results and comparison

In this section the synthesis results of the circuit will be presented. We will also compare these results to implementations found in the literature. First, however, formula's to determine the circuit's calculation time will be given.

Note that the power consumption estimates are to be taken with a grain of salt, since it's very hard for the synthesis tool to come up with accurate numbers.

A. Calculation time

The amount of cycles it takes the circuit to calculate one pairing depends on the size of the field $\mathbb{F}_{2^m}$ and the number of MALUs d used in the arithmetic core. A formula for the number of cycles is

$$c = 21681 + 4322 + 2998 \cdot \left\lceil \frac{m}{d} \right\rceil,$$

with the last two constants being the number of additions and multiplications respectively. The benefits of adding extra MALUs to the circuit quickly diminish due to the large constant factors present in the formula.

B. Synthesis results

The synthesis to an ASIC implementation was performed with Synopsys Design Vision with the maximum area constrained to zero.

Table II contains a breakdown of the component area for the circuit with one MALU. What's striking is that the controller (registers and FSM) contains 92% of the circuit's gates. The large size of the FSM can be explained by the design of the memory. Lots of states are necessary to move every value to its correct position in the register file. It should also be noted that the area of one MALU is almost negligible compared to the rest of the circuit. It's probably a good idea to add some extra MALUs to a real-life implementation to speed up the

Table II
COMPONENT SIZE BREAKDOWN FOR AN IMPLEMENTATION WITH ONE MALU

Component	Area [gates]	
MALU	458	1.7%
$\mathbb{F}_{2^m}$ core		
Logic	783	2.8%
Registers	962	3.5%
Controller		
Logic	13 044	47%
Registers	12 487	45%
Total	27 734	100%

calculations. We will investigate this possibility in the next few paragraphs.

A few implementations with multiple MALUs and a clock frequency of 10 kHz were synthesized, the results of which are listed in Table III. As can be seen, extra MALUs don't add a lot of area. For example, the implementation with six MALUs is only 12% bigger than the one with one MALU. Power consumption also rises pretty slow, the implementation with six MALUs requires only 17% more power. The lower power consumption for the implementations with two MALUs is probably due to a quirk in the synthesis tool.

Table III
SYNTHESIS RESULTS FOR IMPLEMENTATIONS WITH d MALUS

d	Area [gates]		Power @ 10 kHz $[nW]$				Time savings
			Dynamic		Leakage		
1	27 734		96		110		
2	28 423	102%	90	94%	113	103%	47.2%
3	29 071	105%	103	107%	118	107%	62.9%
4	30 278	109%	108	113%	122	111%	71.1%
6	30 956	112%	112	117%	127	115%	78.6%
8	32 782	118%	122	127%	136	124%	82.7%
16	37 798	136%	162	169%	163	148%	88.5%
32	47 833	172%	212	221%	213	194%	91.5%

At a frequency of 10 kHz, it takes an implementation with one MALU 51.5 seconds to complete one pairing calculation. Since this is probably unacceptable in a real-life application, Table IV lists the synthesis results for two implementations that both take 50 ms to finish one calculation. It is obvious that the implementation with two MALUs comes out a lot better, unless a small area is an extremely important factor. Due to the fact that its clock frequency is much lower, power consumption is cut in half compared to the implementation with one MALU.

Table IV
SYNTHESIS RESULTS FOR TWO IMPLEMENTATIONS WITH A CALCULATION TIME OF 50 MS

	1 MALU	2 MALUs	
f [MHz]	10.3	5.44	53%
Area [gates]	27 430	28 155	103%
Power $[\mu W]$			
Dynamic	98.2	48.5	49%
Leakage	$107 \cdot 10^{-3}$	$111 \cdot 10^{-3}$	104%

C. Comparisons

Unfortunately, at the time of this writing, only three ASIC implementations had been published. All of those three focus on speed and it is thus hard to compare them to this design. Since neither Kammler *et al.* [20], nor Kömürcü and Savas [21] list full specifications, it is impossible to compare against their implementations. The implementation by Beuchat et al. [22] does come with full specifications however, and it is thus with this implementation that comparisons will be made. An overview is given in Table V. It should be noted that the implementation in [20] is smaller with its 97k gates than the one in [22].

Table V
COMPARISON OF OUR IMPLEMENTATION WITH OTHER PUBLISHED ASIC IMPLEMENTATIONS

	This work		Beuchat *et al.* [22]
	1 MALU	2 MALUs	
Field	$\mathbb{F}_{2^{163}}$	$\mathbb{F}_{2^{163}}$	$\mathbb{F}_{3^{97}}$
Pairing	Tate	Tate	η_T
Security [bit]†	652	652	922
Technology	0.13 μm	0.13 μm	0.18 μm
Area [gates]	27 430	28 155	193 765
f [MHz]	10.3	5.44	200
Calc. time $[\mu s]$	$50 \cdot 10^3$	$50 \cdot 10^3$	46.7
Power $[mW]$	$98.3 \cdot 10^{-3}$	$48.6 \cdot 10^{-3}$	672
Efficiency $\left[\frac{nJ}{\text{bit}}\right]$‡	7.54	3.73	34.0

† Taken from [11] which is by the same authors as [22].
‡ The lower, the more energy efficient the implementation is. See text.

As is immediately obvious, the implementations in this work are much smaller than the one by Beuchat *et al.* To assess the efficiency of the implementations for application in constrained environments, we propose to calculate the energy efficiency per bit security, which is given by

$$\text{EE} = \frac{\text{power} \times \text{calc. time}}{\text{security}}.$$

Unlike an often used efficiency measure for which power is divided by throughput, this measure takes into account the security of the output.

It's obvious that for use in constrained environments our design is head and shoulders above the design by Beuchat *et al.* With some more MALUs in the implementation, the efficiency will easily be up to twenty times better than theirs. In their defense however, they focused heavily on speed and thus such results are to be expected. It is certainly not the aim of the author to make light of their work.

V. CONCLUSION

In this paper, we presented a hardware implementation for the calculation of the Tate pairing in $\mathbb{F}_{2^m}$. The design focused heavily on a small area and a low energy consumption. An arithmetic core, which can be sped up without any changes to the controller's FSM, was shown. The memory register was optimized for energy efficiency. Due to the flexibility of the arithmetic core, the size and power consumption of the implementation can be fine-tuned in function of the application.

The synthesis results are promising, it is possible to obtain an area lower than 30k gates. Furthermore, the energy efficiency of the circuit is easily ten to twenty times better than existing designs. With a dynamic power consumption as low as 96 nA, this design is a prime candidate for application in constrained environments.

Future work should focus on reducing the FSM's footprint and improve the design of the memory block. Optimal placements of the variables in the register file might cut down on both power consumption and calculation time.

Furthermore, the effect of larger field sizes should be investigated. Finally, the implementation of new pairings such as the η_T and Ate pairing might prove interesting. The calculation time required will be much lower and since they're both based on the Tate pairing, no changes to the underlying should be necessary.

REFERENCES

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proceedings of CRYPTO 84 on Advances in cryptology*. New York, NY, USA: Springer-Verlag New York, Inc., 1985, pp. 47–53.

[2] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing," *SIAM J. Comput.*, vol. 32, no. 3, pp. 586–615, 2003.

[3] L. B. Oliveira, D. F. Aranha, E. Morais, F. Daguano, J. López, and R. Dahab, "TinyTate: Computing the Tate Pairing in Resource-Constrained Sensor Nodes," in *Sixth IEEE International Symposium on Network Computing and Applications*. IEEE Computer Society, 2007, pp. 318–323.

[4] L. B. Oliveira, M. Scott, J. López, and R. Dahab, "TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks," in *5th International Conference on Networked Sensing Systems*, 2008, pp. 173–180.

[5] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, and R. Dahab, "NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks." *European conference on Wireless Sensor Networks (EWSN08)*, 2008.

[6] J. L. Hill and D. E. Culler, "Mica: a wireless platform for deeply embedded networks," *Micro, IEEE*, vol. 22, no. 6, pp. 12–24, 2002.

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

[8] F. Hess, N. P. Smart, and F. Vercauteren, "The Eta Pairing Revisited," *IEEE Transactions on Information Theory*, vol. 52, pp. 4595–4602, 2006.

[9] V. S. Miller, "Short Programs for Functions on Curves," 1986.

[10] P. S. L. M. Barreto, H. Y. Kim, B. Lynn, and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems," in *CRYPTO 02: Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*. Springer-Verlag, 2002, pp. 354–368.

[11] J.-L. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto, and F. Rodrguez-Henrquez, "A Comparison Between Hardware Accelerators for the Modified Tate Pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$," in *Lecture Notes in Computer Science*, vol. 5209. Springer, 2008, pp. 297–315.

[12] A. K. Lenstra and E. R. Verheul, "Selecting Cryptographic Key Sizes," *Journal of Cryptology*, vol. 14, pp. 255–293, 2001.

[13] Certicom Corporation, *SEC 2: Recommended Elliptic Curve Domain Parameters*, september 2000. [Online]. Available: http://www.secg.org

[14] G. Bertoni, L. Breveglieri, P. Fragneto, G. Pelosi, and L. Sportiello, "Software implementation of Tate pairing over GF(2^m)," in *DATE 06: Proceedings of the conference on Design, automation and test in Europe*. European Design and Automation Association, 2006, pp. 7–11.

[15] Faraday Technology Corporation, $0.13\mu m$ *Platinum Standard Cell Databook*, 2004. [Online]. Available: http://www.faraday-tech.com

[16] K. Sakiyama, "Secure Design Methodology and Implementation for Embedded Public-key Cryptosystems," Ph.D. dissertation, KU Leuven, december 2007.

[17] L. Batina, "Arithmetic And Architectures For Secure Hardware Implementations Of Public-Key Cryptography," Ph.D. dissertation, KU Leuven, 2005.

[18] Y. K. Lee and I. Verbauwhede, "A Compact Architecture for Montgomery Elliptic Curve Scalar Multiplication Processor." in *WISA*, ser. Lecture Notes in Computer Science, vol. 4867. Springer, 2007, pp. 115–127.

[19] M. Müller, A. Wortmann, S. Simon, M. Kugel, and T. Schoenauer, "The impact of clock gating schemes on the power dissipation of synthesizable register files," in *ISCAS (2)*, 2004, pp. 609–612.

[20] D. Kammler, D. Zhang, P. Schwabe, H. Scharwaechter, M. Langenberg, D. Auras, G. Ascheid, R. Leupers, R. Mathar, and H. Meyr, "Designing an ASIP for Cryptographic Pairings over Barreto-Naehrig Curves," in *Cryptology ePrint Archive, Report 2009/056*, 2009.

[21] G. Kömürcü and E. Savas, "An Efficient Hardware Implementation of the Tate Pairing in Characteristic Three." in *ICONS*. IEEE Computer Society, 2008, pp. 23–28.

[22] J.-L. Beuchat, H. Doi, K. Fujita, A. Inomata, A. Kanaoka, M. Katouno, M. Mambo, E. Okamoto, T. Okamoto, T. Shiga, M. Shirase, R. Soga, T. Takagi, A. Vithanage, and H. Yamamoto, "FPGA and ASIC Implementations of the η_T Pairing in Characteristic Three," in *Cryptology ePrint Archive, Report 2008/280*, 2008.