Low-Energy Finite Field Arithmetic Primitives for Implementing Security in Wireless Sensor Networks

Kealan McCusker, Noel O'Connor and Dermot Diamond
Adaptive Information Cluster
Dublin City University, Ireland
Email: kealan.mccusker@eeng.dcu.ie

Abstract— **In this paper we propose the use of Identity Based Encryption (IBE) for ensuring a secure wireless sensor network. In this context we have implemented the arithmetic operations required for the most computationally expensive part of IBE, which is the Tate pairing, in 90nm CMOS and obtained area, timing and energy figures for the designs. Initial results indicate that a hardware implementation of IBE would meet the strict energy constraint of a wireless sensor network node.**

I. Introduction

Recent advances in radio and digital electronics have enabled system on chip technologies to be developed that will incorporate sensing, computation and communication. These devices are known as wireless sensor nodes and are the subject of very active research at present. It is envisaged [1] that eventually they will cost considerably less than one dollar, and hence could be leveraged to provide a distributed wireless sensor network containing many thousands of nodes. The characteristics that are attributed to wireless micro-sensors are that they have limited memory, are very inexpensive, and have multi-year life spans from a single power source. The total energy in the power source for the wireless sensor node would be of the order of 1000 joules. Thus, it is imperative that the system architecture of the nodes and the network as a whole should be designed with an aim to minimizing energy dissipation in all aspects of operation. Furthermore, we believe that in order for these networks to be deployed in real applications, it is essential that the issue of security is solved [2], [3].

We believe that a symmetric key cryptosystem is appropriate for communication between the nodes, though a pre-installed system wide symmetric key or pair-wise keys stored on the devices are not suitable for reasons of security and lack of memory, respectively. Therefore an asymmetric or public key system is required to establish the symmetric keys between individual nodes.

With a traditional approach, if node A wants to communicate with node B, it first has to receive B's digital certificate before it can send a message. When the two nodes are in direct radio communication this will mean one transmission from B to A. In the case of B being out of range of A, the digital certificate would have to be relayed to A via intermediate nodes. As the radio is likely to be the main consumer of energy in the node, it is important to minimize the number of transmissions. This can be achieved using IBE [4] in which there is no need for a certificate to bind a node's identity to its public key, as the node's identity can be used as the public key.

The remainder of the paper is organized as follows: the Tate pairing, a key component of IBE, and methods for calculating it, are discussed in Section II in the context of low energy hardware implementation. In light of this discussion, descriptions of the hardware units required for the implementation are outlined in Section III. Finally, conclusions and directions for future work are presented in Section IV.

II. Tate Pairing

The Boneh-Franklin scheme is one example of IBE [5]. This approach requires arithmetic on the supersingular elliptic curve, E, defined over $GF(2^{283})$ (1), and the Tate pairing for its implementation.

$$E(GF(2^{283})) = y^2 + y \equiv x^3 + x + 1 \quad (1)$$

The Tate pairing is the map from two points on the elliptic curve to the multiplicative group $(GF(2^{283\times4}))^*$

$$E(GF(2^{283}))[l] \times E(GF(2^{283\times4}))[l] \rightarrow (GF(2^{283\times4}))^* \quad (2)$$

which has the property of bilinearity. It can be defined as

$$e_l(P,Q) = f_p(\phi(Q))^{2^{2\times283}-1} \quad (3)$$

where $P, Q \in E(GF(2^{283}))[l]$, f_p is a rational function on the curve such that its divisor $(f_p) = l(P) - l(0)$ and ϕ is a distortion map [6]. Such large finite fields are required as the discrete logarithm problem must be hard to solve in the field that the pairing maps to.

It is well known that the Tate pairing is the most computationally expensive operation in an IBE algorithm, therefore in a wireless sensor node, it should be implemented in hardware in order to reduce the amount of energy required to compute the pairing. The pairing can be calculated using Miller's algorithm [7], and improvements have recently been proposed to reduce its cost in terms of its execution time [6], [8], [9], [10]. We are currently implementing the algorithm [9], [10], as it is the most amenable to a low energy hardware implementation , since it is in characteristic two and has a regular structure which maps well to hardware (see Algorithm 1).

0-7803-9584-0/06/$20.00©2006 IEEE.

Algorithm 1 Algorithm for calculating $f_p(\phi(Q))^{2^{2\times 283}-1}$ [9]

$P = (x_p, y_p), Q = (x_q, y_q)$
$C(x) = c_3x^3 + c_2x^2 + c_1x + c_0 = 0$
for $i = 1$ to 283 **do**
  $x_p = x_p^2, y_p = y_p^2$
  $z = x_p + x_q, m_1 = x_p x_q$
  $w = z + m_1 + y_p + y_q + 1$
  $m_2 = c_0 w, m_3 = (c_2 + c_3)(z + 1)$
  $m_4 = (c_1 + c_2 + c_3)w$
  $m_5 = (c_0 + c_2 + c_3)(w + z + 1)$
  $m_6 = c_3(z + 1)$
  $m_7 = (c_1 + c_2)(w + z + 1)$
  $c_0 = m_2 + m_3 + c_3$
  $c_1 = m_2 + m_4 + m_5 + m_6 + c_0 + c_3$
  $c_2 = m_2 + m_4 + m_5 + m_7 + c_1$
  $c_3 = m_4 + m_7 + c_2$
  $x_q = x_q^{2^{283-1}}$
  $y_q = y_q^{2^{283-1}}$
end for
return $C(x) = C(x)^{2^{2\times 283}-1}$

III. Arithmetic Operations

All operations that are used for the various algorithms in this paper take place in binary extension fields; either $GF(2^{283})$ or $GF(2^{283\times 4})$. From algorithm 1 we have identified the arithmetic operations that are required as addition, multiplication, and inversion in both fields. In addition, a circuit is required to perform the squaring operations in $GF(2^{283})$, and exponentiation in $GF(2^{283\times 4})$.

In the subfield $GF(2)$, addition is carried out using modulo two arithmetic, and hence can be performed in hardware using an XOR gate. Addition is equivalent to subtraction in $GF(2)$. Also, multiplication is performed using an AND gate in hardware.

The polynomial basis representation is used for the elements of the two finite fields. The irreducible polynomials which generates the fields $GF(2^{283})$ and $GF(2^{283\times 4})$ are,

$$\begin{aligned} f(x) &= x^{283} + x^{119} + x^{97} + x^{93} + 1 \\ p(x) &= x^4 + x + 1 \end{aligned} \tag{4}$$

and are defined over $GF(2)$ and $GF(2^{283})$, respectively.

A. Addition

Addition in a binary extension field is trivial to implement in hardware. It is an array of XOR gates, one for every two bits of the operands that are to be added. As the circuits are much smaller in area and energy consumption than the circuits for the other operations outlined in this paper, their contribution to the overall energy and timing figures for the Tate pairing are neglected.

B. Multiplication in $GF(2^{283})$

The aim of this work is to reduce the energy consumption of the circuits that are implemented to carry out the various operations. As we are implementing our design in 90 nm CMOS and due to the preeminence of static energy over dynamic energy consumption when using deep sub-micron technologies, it would be best if the operation could be preformed as fast as possible. This would then mean that the circuits could be powered off when not in use, hence static energy is saved at the cost of greatly increased area. A fast bit-parallel multiplier is approximately 300,000 gates in area. This would be prohibitive in terms of manufacturing cost for a wireless sensor node.

Thus, a bit-serial approach to designing the multiplier is warranted. Multiplication is to be performed using the modified shift-register (MSR) multiplier [11] which has superior low energy attributes compared to the standard shift-register due to its early exit mechanism as outlined below.

The MSR multiplier is based on the following observation for $A(x), B(x), C(x) \in GF(2^{283})$.

$$\begin{aligned} C(x) &= A(x)B(x) \\ &= (b_{282}x^{282}A(x) \pmod{f(x)}) + \cdots \\ &+ (b_1 xA(x) \pmod{f(x)}) + (b_0 A(x) \pmod{f(x)}) \end{aligned} \tag{5}$$

$C(x)$ can be calculated using a shift and add algorithm where the first partial product is $b_0A(x)$. $B(x)$ is then shifted right one bit while at the same time $A(x)$ is multiplied by x and reduced mod $f(x)$. It is added to the previous product if b_1 is equal to 1. The algorithm will terminate when the value of the right shift register is equal to zero (see Algorithm 2). This is the early exit mechanism referred to above, as it could finish after one clock cycle or 283 clock cycles. The datapath circuitry is shown in Fig. 1. We use a right shift and linear feedback barrel shift registers to improve the performance of the circuit. Two, three, four or five consecutive zero bits are searched for, and the registers shifted accordingly. Five bits is considered the optimum choice, as for each bit searched there is a cost of approximately 400 gates, and the probability of five zeros is $\frac{1}{32}$ which is quite high. When the system is complete we will implement clock tree gating at a higher level to reduce the energy being dissipated on the clock nets.

Algorithm 2 Multiplication in $GF(2^{283})$

Require: $C(x) = A(x)B(x) \pmod{f(x)}$
  $C(x) = 0$
  while $B(x) \neq 0$ **do**
    $C(x) = C(x) + b_0A(x)$
    right shift $B(x)$
    $A(x) = xA(x) \pmod{f(x)}$
  end while

C. Multiplication in $GF(2^{283\times 4})$

Multiplication in $GF(2^{283\times 4})$ is performed using a parallel design which contrasts with the serial approach taken for $GF(2^{283})$. We have done this as the node should be as inexpensive as possible, hence resource reuse should take place at the higher level. As the multiplication in $GF(2^{283})$ is

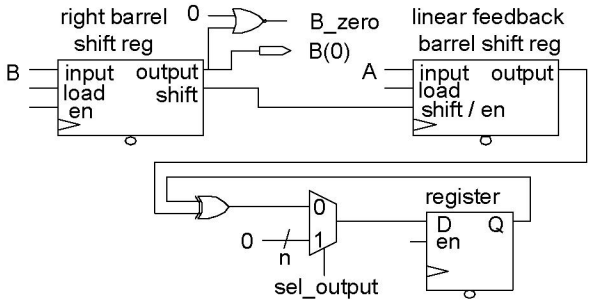

Fig. 1. Datapath circuit for the multiplier in $GF(2^{283})$

required for the calculation of the Tate pairing, it can be used to perform the multiplication in $GF(2^{283\times4})$. This sharing of resources will lead to a decrease in the monetary cost of the system.

If normal polynomial multiplication for the two elements is employed, and the terms reduced mod $p(x)$, then this would result in sixteen multiplications in $GF(2^{283})$. However, using Karatsuba's algorithm[12] this figure can be reduced. The two operands $A(x), B(x) \in GF(2^{283\times4})$ are written as,

$$\begin{aligned} A(x) &= x^2(a_3x + a_2) + (a_1x + a_0) = x^2A_h + A_l \\ B(x) &= x^2(b_3x + b_2) + (b_1x + b_0) = x^2B_h + B_l \end{aligned} \tag{6}$$

and the partial products are,

$$\begin{aligned} D_1 &= A_lB_l \\ D_2 &= (A_l + A_h)(B_l + B_h) \\ D_3 &= A_hB_h \end{aligned} \tag{7}$$

and hence,

$$\begin{aligned} C(x) = A(x)B(x) &= (x^2A_h + A_l)(x^2B_h + B_l) \\ &= x^4A_hB_h + x^2A_hB_l + x^2A_lB_h + A_lB_l \\ &= x^4D_3 + x^2(D_2 - D_1 - D_3) + D_1 \end{aligned} \tag{8}$$

Karatsuba's algorithm can be applied recursively to D_1, D_2 and D_3 and the values for D_1, D_2 and D_3 substituted into (8). The resultant equation is reduced mod $p(x)$ and this leads to the following coefficients for $C(x)$;

$$\begin{aligned} c_0 &= a_2b_2 + (a_1 + a_3)(b_1 + b_3) + a_0b_0 + a_3b_3 + a_1b_1 \\ c_1 &= (a_2 + a_3)(b_2 + b_3) + (a_1 + a_3)(b_1 + b_3) \\ &\quad + (a_0 + a_1)(b_0 + b_1) + a_0b_0 \\ c_2 &= (a_2 + a_3)(b_2 + b_3) + a_1b_1 + (a_0 + a_2)(b_0 + b_2) + a_0b_0 \\ c_3 &= a_0b_0 + (a_0 + a_1)(b_0 + b_1) + a_1b_1 + (a_0 + a_2)(b_0 + b_2) \\ &\quad + (a_0 + a_2 + a_1 + a_3)(b_0 + b_2 + b_1 + b_3) \\ &\quad + (a_1 + a_3)(b_1 + b_3) + a_2b_2 + (a_2 + a_3)(b_2 + b_3) \end{aligned} \tag{9}$$

From (9) it can be seen that nine multiplications and twenty two additions are required in $GF(2^{283})$.

The datapath circuitry is shown in Fig. 2. The datapath width is 283 bits wide. As well as using the MSR multiplier which improves upon energy consumption, reductions can also be achieved by holding wires at a constant value when not in use, and hence reducing dynamic energy dissipation. This is accomplished through the signals enadd10 and enadd12 (not shown), which gate the inputs and the combinational logic, respectively. Clock tree gating is used at a higher level to reduce the energy being dissipated on the clock nets. The MSR multipliers clocks are gated with their "done" signals. This technique take advantage of the early exit of the MSR multipliers due to their structure.

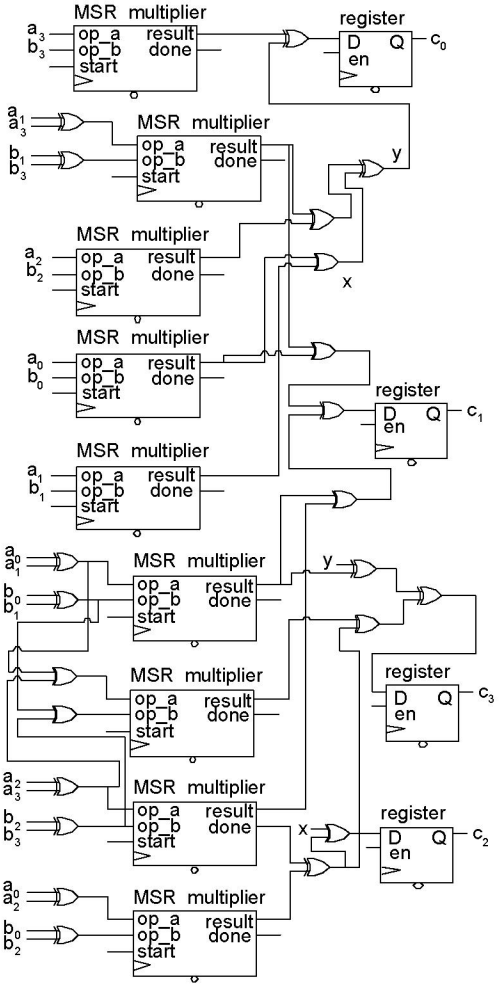

Fig. 2. Datapath circuit for the multiplier in $GF(2^{283\times4})$

D. Squaring

The bit-serial multiplier described in Section III-B could be used for squaring, but as squaring is used 283 times in each loop and in the inversion circuitry, this is not the optimum choice. Instead, we have implemented a bit-parallel squaring circuit which does not require a significant amount of area (see Table I). In our architecture there are two squaring circuits that operate in parallel.

E. Exponentiation

The only exponentiation that is required is $\beta = \alpha^{2^{283}}$ where $\alpha, \beta \in GF(2^{283\times 4})$. This is also known as the Frobenius map. The exponentiation is as follows;

$$\begin{aligned}\sum_{i=0}^{3} b_i x^i &= \left(\sum_{i=0}^{3} a_i x^i\right)^{2^{283}} \\ &= \sum_{i=0}^{3} a_i^{2^{283}} x^{i2^{283}} \\ &= \sum_{i=0}^{3} a_i x^{i2^{283}} \pmod{p(x)}\end{aligned} \tag{10}$$

For a proof see [13]. Due to the fact that $x^{2^4} = x \pmod{p(x)}$, it can be seen that

$$\sum_{i=0}^{3} b_i x^i = (a_0 + a_1) + (a_2 + a_3)x + a_1 x^2 + a_3 x^3 \tag{11}$$

The Frobenius map can therefore be implemented in hardware with two additions in $GF(2^{283})$ and reordering of the coefficients.

F. Inversion in $GF(2^{283})$

The technique for inversion for $\beta \in GF(2^{283})$ is based on Fermat's little theorem (12). We have used this method as it allows reuse of the squaring and multiplication circuits.

$$\beta^{2^{283}-1} \equiv 1 \pmod{f(x)}. \tag{12}$$

This means that $\beta^{2^{283}-2}\beta \equiv 1 \pmod{p(x)}$ and therefore $\beta^{2^{283}-2}$ is the inverse of β. The inverse of β can be calculated with the square and multiply technique using the following observations;

$$\begin{aligned}\beta^{-1} = \beta^{2^{283}-2} &= \beta^{2^1}\beta^{2^2}\beta^{2^3}\cdots\beta^{2^{282}} \\ &= (\cdots(((\beta)^2\beta)^2\beta)^2\cdots\beta)^2\end{aligned} \tag{13}$$

This algorithm requires 282 squarings and 281 multiplications in $GF(2^{283})$. From Section III-B and III-D, it can be seen that multiplication in $GF(2^{283})$ is a very expensive operation in terms of time and energy. It would be beneficial to reduce the number of multiplications. This can achieved using the techniques of Itoh and Tsujii [14] where the following recursive formula is used to reduce the number of multiplications. When n is odd

$$\beta^{2^{n-1}-1} = \left(\beta^{2^{\frac{n-1}{2}}-1}\right)^{2^{\frac{n-1}{2}}} \beta^{2^{\frac{n-1}{2}}-1} \tag{14}$$

and when n is even

$$\beta^{2^{n-1}-1} = \left(\beta^{2^{n-2}}\right)^2 \beta \tag{15}$$

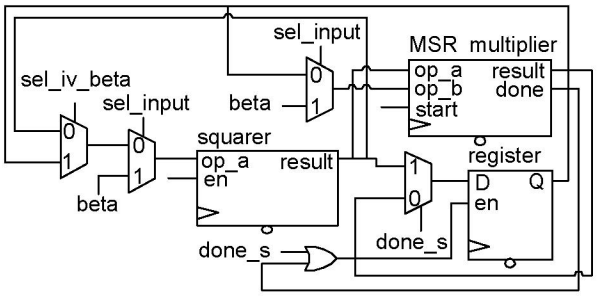

Fig. 3. Datapath circuit for the inverter in $GF(2^{283})$

β^{-1} can now be decomposed using (14) and (15)

$$\begin{aligned}\beta^{2^{283}-2} &= \left(\beta^{2^{283-1}-1}\right)^2 \\ \beta^{2^{283-1}-1} &= \left(\beta^{2^{141}-1}\right)\left(\beta^{2^{141}-1}\right)^{2^{141}} \\ \beta^{2^{142-1}-1} &= \beta\left(\beta^{2^{140}-1}\right)^2 \\ \beta^{2^{141-1}-1} &= \left(\beta^{2^{70}-1}\right)\left(\beta^{2^{70}-1}\right)^{2^{70}} \\ \beta^{2^{71-1}-1} &= \left(\beta^{2^{35}-1}\right)\left(\beta^{2^{35}-1}\right)^{2^{35}} \\ \beta^{2^{36-1}-1} &= \beta\left(\beta^{2^{34}-1}\right)^2 \\ \beta^{2^{35-1}-1} &= \left(\beta^{2^{17}-1}\right)\left(\beta^{2^{17}-1}\right)^{2^{17}} \\ \beta^{2^{18-1}-1} &= \beta\left(\beta^{2^{16}-1}\right)^2 \\ \beta^{2^{17-1}-1} &= \left(\beta^{2^{8}-1}\right)\left(\beta^{2^{8}-1}\right)^{2^{8}} \\ \beta^{2^{9-1}-1} &= \left(\beta^{2^{4}-1}\right)\left(\beta^{2^{4}-1}\right)^{2^{4}} \\ \beta^{2^{5-1}-1} &= \left(\beta^{2^{2}-1}\right)\left(\beta^{2^{2}-1}\right)^{2^{2}} \\ \beta^{2^{3-1}-1} &= \beta\beta^2\end{aligned} \tag{16}$$

Therefore only 11 multiplications and 283 squarings are required to obtain the inverse of β.The datapath circuitry is shown in Fig. 3.

G. Inversion in $GF(2^{283\times 4})$

Fermat's little theorem (12), [15] can also be used to get the inversion of an element $\alpha \in GF(2^{283\times 4})$(see Algorithm 3). Again, this approach has been used to reduce area, whilst achieving maximum performance in terms of energy and time per operation. If the general case $\alpha \in GF(2^{nm})$ is considered then the inverse is

$$\begin{aligned}\alpha^{-1} = \alpha^{2^{nm}-2} &= \alpha^{\frac{2^{nm}-1}{2^n-1}(2^n-1)-1} \\ &= \alpha^{r(2^n-2)+r-1} = (\alpha^r)^{-1}\alpha^{r-1}\end{aligned} \tag{17}$$

where $r = \frac{2^{nm}-2}{2^n-1}$. The technique is based on the fact that

$$\alpha^r \in GF(2^n), \qquad \forall \alpha \in GF(2^{nm}) \tag{18}$$

Algorithm 3 Inversion in $GF((2^n)^m)$

Require: $\beta = \alpha^{-1}$
α^{r-1}
$\alpha^r = \alpha^{r-1}\alpha$
$(\alpha^r)^{-1}$
$(\alpha^r)^{-1}\alpha^{r-1}$

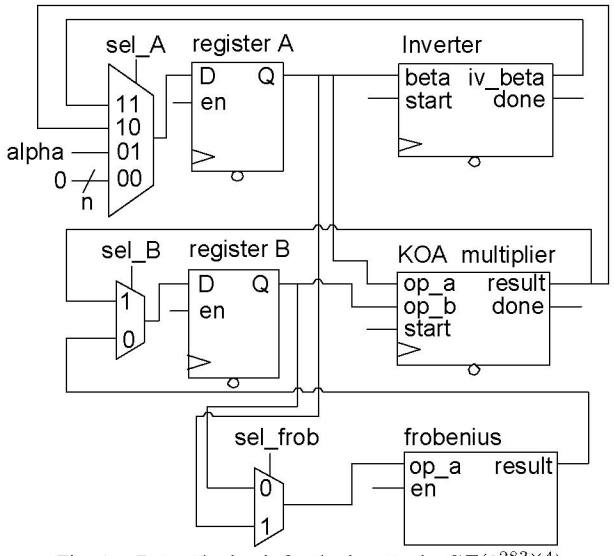

Fig. 4. Datapath circuit for the inverter in $GF(2^{283\times4})$

When working in the field $GF(2^{283\times4})$ the first stage of the inversion in algorithm 3 is obtained by the following equations.

$$r - 1 = 2^{283} + (2^{283})^2 + (2^{283})^3 \tag{19}$$

If we let

$$\beta = \alpha^{r-1} = \alpha^{2^{283}+(2^{283})^2+(2^{283})^3} \tag{20}$$

where $\alpha, \beta \in GF(2^{283\times4})$ this can be rewritten as

$$\beta = \left(\left(\alpha^{2^{283}}\alpha\right)^{2^{283}}\alpha\right)^{2^{283}} \tag{21}$$

Three 2^{283} exponentiations (see Section III-E) and two multiplications in $GF(2^{283\times4})$ (see Section III-C) are required to perform this operations. The next stage is multiplication in $GF(2^{283\times4})$. As $\alpha^r \in GF(2^{283})$, $(\alpha^r)^{-1}$ is an inversion in $GF(2^{283})$, and finally the last step is multiplication in $GF(2^{283\times4})$. The datapath circuitry is shown in Fig. 4.

IV. Conclusion

The primitives were implemented in VHDL and synthesised targeting a 90nm library. Power measurements were taken using the PrimePower tool from Synopsys. All circuits in our design operate at 250MHz. From an analysis of the algorithm we are implementing (see Algorithm 1), and using the results given in Table I, it is estimated the Tate pairing would require 18.36 μJ. Given that the Tate pairing is the most computationally expensive process in IBE, we believe that this figure is appropriate in the context of meeting the extremely restrictive energy constraint of a wireless sensor node. We estimate the time taken to compute the pairing to be 0.84 ms, this compares favourably with the latency estimate of 1.48 ms presented in [16]. In our future work we will integrate these arithmetic primitives into a low energy, low cost implementation of the Tate pairing.

TABLE I
Results for $GF(2^{283})$ and $GF(2^{283\times4})$ arithmetic primitives

operation	area (gates)	power (mW)	time/op. (ns)	energy/op. (nJ)
mult. $GF(2^{283})$	14196	9.78	897	8.77
mult. $GF(2^{283\times4})$	162550	88.70	940	83.41
squaring $GF(2^{283})$	4167	2.46	4	0.01
inverse $GF(2^{283})$	23945	12.15	10769	130.85
inverse $GF(2^{283\times4})$	220625	6.61	14666	970.20

Acknowledgment

This material is based on work supported by Science Foundation Ireland under Grant Nos. 03/IN.3/I361 and the Informatics Commercialisation initiative of Enterprise Ireland.

References

[1] J. M. Rabaey, M. Ammer, J. L. da Silva Jr., D. Patel, and S. Roundy, "PicoRadio supports ad hoc ultra-low power wireless networking," *Computer Magazine*, pp. 42–48, July 2000.

[2] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensors networks," *Communications of the ACM*, vol. 47, no. 6, pp. 53–57, June 2004.

[3] H. Chan and A. Perrig, "Security and privacy in sensors networks," *IEEE Computer*, vol. 36, no. 10, pp. 103–105, Oct. 2003.

[4] A. Shamir, "Identity-based cryptosystems and signature schemes," in *Proc. Crypto '84*, Santa Barbara, California, USA, Aug. 1984, pp. 47–54.

[5] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," *SIAM J. of Computing*, vol. 32, no. 3, pp. 586–614, 2003.

[6] P. S. L. M. Barreto, B. Lynn, and M. Scott, "Effi cient implementation of pairing-based cryptosystems," *J. Cryptol.*, vol. 17, no. 4, pp. 321–334, 2004.

[7] V. S. Miller, "Short programs for functions on curves," 1986, unpublished. [Online]. Available: http://crypto.stanford.edu/miller/miller.pdf

[8] S. D. Galbraith, K. Harrison, and D. Soldera, "Implementing the Tate pairing," in *ANTS-V: Proceedings of the 5th International Symposium on Algorithmic Number Theory*. London, UK: Springer-Verlag, 2002, pp. 324–337.

[9] S. Kwon, "Effi cient Tate pairing computation for elliptic curves over binary fi elds." in *ACISP*, 2005, pp. 134–145.

[10] P. S. L. M. Barreto, S. Galbraith, C. O. hEigeartaigh, and M. Scott, "Effi - cient pairing computation on supersingular abelian varieties,"Cryptology ePrint Archive, Report 2004/375, 2004, http://eprint.iacr.org/.

[11] E. D. Mastrovito, "VLSI architectures for computation in Galois fi elds," Ph.D. dissertation, Linköping University, Linköping, Sweden, 1989.

[12] C. Paar, P. Fleischmann, and P. Roelse, "Effi cient multiplier architectures for Galois fi elds $GF(2^{4n})$," *IEEE Trans. Comput.*, vol. 47, no. 2, pp. 162–170, 1998.

[13] R. J. McEliece, *Finite fields for computer scientists and engineers*. Kluwer Academic Publishers, 1987.

[14] 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, 1988.

[15] J. Guajardo and C. Paar, "Itoh-Tsujii inversion in standard basis and its application in cryptography and codes," *Des. Codes Cryptography*, vol. 25, no. 2, pp. 207–216, 2002.

[16] M. Keller, T. Kerins, and W. P. Marnane, "FPGA implementation of a $GF(2^{4M})$ multiplier for use in pairing based cryptosystems," in *Field Programmable Logic and Applications*, 2005, pp. 594–597.