On the Development of a Fast Elliptic Curve Cryptosystem[1]

G.B. Agnew, R.C. Mullin, S.A. Vanstone
University of Waterloo

Abstract

In this paper, we look at the development of a high speed elliptic curve cryptosystem based on a 40 Mhz. Motorola M68030 processor and a high speed optimal normal basis coprocessor for the ground field $GF(2^{155})$. The advantage of this system is the relatively small block size required for high security applications such as key management and digital signatures. In addition, the design is very compact and efficient and can be easily fit onto a standard Smart Card wafer (the coprocessor core requires less than 1 sq.mil. or < 4% of the area available on a Smart Card).

Introduction

Since their introduction by Diffie and Hellman in 1976 [1], researchers have been searching for practical implementations of public key cryptographic systems. The two most popular systems are based on the complexity of factoring the product of two large primes (RSA) or the difficulty of taking logarithms over a large field. To attain acceptable levels of security, both of these systems must use very large block sizes (the current trend is to use blocks of over 1000 bits for long term security). The complexity of performing calculations over these large block sizes generally makes the performance of software implementations unacceptable. Thus, hardware implementation in VLSI have been fabricated. At present, both public key systems have been realized with commercially available VLSI devices [2], [3], [4].

Even with the availability of high speed implementations of these systems, two issues still remain. The requirements that large block sizes be used leads to large storage requirements especially in such applications as financial transactions where digital signatures may be used for long term verification. In addition, the high levels of complexity of such devices makes transfer to such applications as smart cards difficult.

In 1985, Koblitz [5] and independently Miller [6], proposed the use of elliptic curves as the basis of a public key cryptographic system. It was believed at that time that no subexponential algorithm existed for solving the elliptic logarithm problem (the only known attack was Shank's Giant-Step-Baby-Step). More recently, Menezes, Vanstone and Okamoto [7], have discovered a new method of computing logarithms on the small, very special class of super-singular curves. Use of non-super singular curves does not have any such drawbacks.

---

[1] For a more complete treatment of this topic the reader is refered to reference [11].

R.A. Rueppel (Ed.): Advances in Cryptology - EUROCRYPT '92, LNCS 658, pp. 482-487, 1993.

Non-Super Singular Curves Using Affine Co-ordinates

The use of elliptic curves in public key cryptographic systems is analogous to the use of discrete exponentiation substituting addition of points on the curve for multiplication in the field. The elliptic logarithm problem can thus be expressed as:

given a known starting point on the curve $P = (x_1, y_1)$ and a second point

$$Q = k\,P$$

find the integer k.

This is thought to be a hard problem even for relatively small curves.

To implement a system in hardware, we will only consider curves of characteristic 2. The curves we are interested in are of the form:

$$y^2 + xy = x^3 + ax + b$$

Operations on the curve are defined in the following way. For points $P = (x_1, y_1)$ and $Q = (x_2, y_2)$,

$$-P = (x_1, y_1 + x_1)$$

and, for $Q \neq -P$

$$x_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)^2 + \left(\dfrac{x_1 + y_2}{x_1 + x_2}\right) + x_1 + a & P \neq Q \\ \dfrac{b}{x_1^2} + x_1^2 & P = Q \end{cases}$$

$$y_3 = \begin{cases} \left(\dfrac{y_1 + y_2}{x_1 + x_2}\right)(x_1 + x_3) + x_3 + y_1 & P \neq Q \\ x_1^2 + \left(x_1 + \dfrac{y_1}{x_1}\right)x_3 + x_3 & P = Q \end{cases}$$

As mentioned above, using the elliptic curve as a cryptosystem involves choosing a secret integer k and forming $Q = k\,P$. In practice, this process would be done by using the binary expansion of k and forming the successive "squares" of the known point P, that is,

$$\underline{k} = k_0 + k_1 2^1 + k_2 2^2 + \ldots + k_{n-1} 2^{n-1}$$

or

$$Q = (\ldots(((P\,k_0) + 2P\,k_1) + 4P\,k_2)\ldots + 2^{n-1}P\,k_{n-1})$$

Using the affine method, we compute the x and y co-ordinates at each step in the computation. This requires the calculation of an inverse at each step in the squaring process; this will be the most time consuming part of the computation as each inverse generation requires several multiplications in the underlying field.

Computations Using Projective Co-ordinates

Another method of computing points on the curve involves projective coordinates. The projective equation for the non-super singular elliptic curve is written as:

$$z\,y^2 + z\,x\,y = x^3 + z\,x^2 a + b\,z^3$$

We set $Q = (x_1, y_1, z_1)$, $P = (x_2, y_2, 1)$ and define $P + Q = (x_3'', y_3'', z_3'')$.

For the case $P \neq Q$ (addition operation), let

$$x_3' = z_1^2 \{ z_1 (b + y_2^2 + x_2 (y_2 + x_2^2) + x_2^2 a) + (x_2 y_1 + x_1 (y_2 + x_2^2))\} + z_1 (x_1^2 x_2) \ .$$

Then

$$x_3'' = (x_1 + x_2 z_1) x_3'$$

$$y_3'' = (x_1 + x_2 z_1)[(y_1 + y_2 z_1) + y_1 (x_1 + x_2 z_1)] + [(y_1 + y_2 z_1) + (x_1 + x_2 z_1)]$$

$$z_3'' = z_1 (x_1 + x_2 z_1)^3$$

For the case $P = Q$ (doubling operation), let

$$x_3' = x_1^4 + z_1^4 b$$

Then

$$x_3'' = (x_1 z_1) x_3'$$

$$y_3'' = (x_1^2 + y_1 z_1) x_3' + (x_1^4 + x_3') x_1 z_1$$

$$z_3'' = (x_1 z_1)^3$$

The advantage of using this method is that only one inverse operation is required at the end of the calculation to divide out the z coordinate.

A Hardware Implementation

In our previous investigations, it was found that a very fast and efficient multiplier architecture could be developed for fields of characteristic two which had optimal normal basis representations [8], [9]. This led to the development of a single chip public key processor ([3]) which per-

formed computations in the field $GF(2^{593})$. We used this device in conjunction with a Motorola M56000 DSP to implement an elliptic curve cryptosystem. As reported in [10], this system achieved a throughput of about 5 Kbps..

Our experience with this system showed that the bottleneck in computation was the I/O section of the VLSI device. Thus, any processor designed specifically for an elliptic curve system would have to have a very fast I/O structure.

With the experience gained we set out to construct a simple, fast arithmetic processor which can be used to perform elliptic curve computations in conjunction with a control processor. Our objective was to build a device which could be fabricated in a custom gate array and yet be quite secure. Our choice was to implement an optimal normal basis multiplier for $GF(2^{155})$. The gate array device uses three registers to implement the multiplier structure and interconnection, a controller to implement the elementary operations (such as shifts, XOR's, multiplies, etc.) as well as incorporating a very fast 32 bit wide I/O structure. The device was fabricated using a 1.5 micron HCMOS gate array with a clock speed of 40 MHz. and required less than 12,000 gates. For the primary operations, the speed can be calculated as:

OPERATION	SIZE	CLOCK CYCLES
Multiplication	155 bit blocks	156
Calculation of Inverse	24 multiplications	approx. 3800
I/O	5 - 32 bit transfers per read/write to registers @ 2 clock cycles per transfer	10
Addition (XOR) and elementary register operation	155 bit parallel operation	2

To realize the elliptic curve system, a module was constructed with a 40 Mhz. Motorola M68030 as the control processor. The main reasons for this choice was the 32 bit internal bus structure and the availability of fast I/O between the 68030 and the $GF(2^{155})$ coprocessor.

Throughput Calculations

If we consider point multiplication by an integer with Hamming weight 30, this will require about 154 point doublings and 29 point multiplications. Using projective coordinates for a non-super singular curve, doubling requires 6 multiplies and point addition requires 13 multiplies. At the end of the computation, a single inverse operation followed by 2 multiplies must be

performed to return to affine coordinates. Allowing for I/O overhead in the doubling and multiplication routines, the device will be able to perform at least 145
integer multiplications per second. For use in an encryption system, each of X and Y coordinates can be used so 310 bits can be sent per point calculation for a throughput of approximately 50 Kbps..

In the above calculations, we note that a significant portion of the time is spent in doing the point doublings[2]. In elliptic curve systems, the same base point P can be used repeatedly. If this is the case, then all of the squares can be precomputed which will increase the throughput by a factor of 4 to approximately 200 Kbps.. The storage requirements for the point squarings is less than 6 Kbytes, a relatively small amount.

We have also investigated the area required to implement the elliptic curve coprocessor in a Smart Card wafer. Using new techniques, our estimates show that the registers necessary to implement the elliptic curve system will require less than 1 sq.mil. (or < 4% of the area available on the Smart Card). Design and construction of a Elliptic Curve Smart Card is currently underway and we believe that this will be the first full implementation of a fast, efficient and compact public key system on a card.

Summary

In this paper, we have described the deign and implementation of a fast processor for performing non-super singular elliptic curves over a base field of $GF(2^{155})$. This device is capable of realizing encryption speeds of up to 50 Kbps when clocked at 40 MHz. In addition to its high speed, it is exceptionally simple in implementation requiring less than 12,000 gates to fabricate. This device will have future applications in such areas as smart cards where compactness, high speed and high levels of security are desirable.

References

1. Diffie, W., M. Hellman, "New Directions in cryptography", IEEE Trans. on Info. Theory, vol. IT-22, pp.644-654, Nov. 1976.
2. Brickell, E., "A survey of hardware implementations of RSA", Proceedings of CRYPTO'89, Springer Verlag, pp. 368-370., Aug. 1989.
3. CA34C168 Data Encryption Processor Data Sheet, Newbridge Microsystems, Kanata, Ontario, Canada.
4. CY1024 Data Sheet, Cylink Corp., Sunnyvale, California, USA.

---

[2]As an aside, one of the main advantages of using normal basis representation for fields of characteristic 2 in discrete exponentiation is that squaring is effectively free, that is, a simple cyclic shift. Unfortunately, this is not the case in elliptic curve point doubling as many operations are required.

5. Koblitz, N., "Elliptic curve cryptosystems" Mathematics of computation - 48, 1987 pp.203-209.
6. Miller, V., "Use of elliptic curves in cryptography", Proceedings of CRYPTO'85, Springer Verlag, Aug. 1985, pp.417-426.
7. Menezes, A., S. Vanstone, T. Okamoto, "Reducing elliptic curve logarithms to logarithms in a finite field", STOC 1991, ACM Press, pp. 80-89, May 1991.
8. Agnew, G., T. Beth, R. Mullin, S. Vanstone, "Arithmetic operations in GF(2^n), to appear, Journal of Cryptology.
9. Agnew, G., R. Mullin, S. Vanstone, "An implementation of a fast public key cryptosystem", to appear, Journal of Cryptology.
10. Agnew, G., R. Mullin, S. Vanstone, "A fast elliptic curve cryptosystem", Lecture Notes in Computer Science #434, Proceedings of Eurocrypt'89, Springer Verlag, Apr. 1989, pp. 706-708.
11. Agnew, G., R. Mullin, S. Vanstone, "An implementation of elliptic curve cryptosystems over F_{2155}", submitted to IEEE Journal on Selected Areas in Communications.