PARALLELIZING GF(P) ELLIPTIC CURVE CRYPTOGRAPHY COMPUTATIONS FOR SECURITY AND SPEED

Adnan Abdul-Aziz Gutub, Mohammad K. Ibrahim*, Turki F. Al-Somani,

Computer Engineering Department, King Fahd University of Petroleum and Minerals
Dhahran 31261, Saudi Arabia
Email: gutub@kfupm.edu.sa

Abstract:

The elliptic curve cryptography can be observed as two levels of computations, upper scalar multiplication level and lower point operations level. We combine the inherited parallelism in both levels to reduce the delay and improve security against the simple power attack. The best security and speed performance is achieved when parallelizing the computation to eight parallel multiplication operations. This strategy is worth considering since it shows very attractive performance conclusions.

Keywords: ECC, GF(p) arithmetic, Projective coordinate systems, parallel elliptic curve point operations.

1. *Introduction:*

In recent years, Elliptic Curve Cryptosystem (ECC), which was originally proposed by Niel Koblitz and Victor Miller in 1985 [1-9], is seen as a serious alternative to RSA since it requires a much shorter world length. ECC with a key size of 128~256 bits is shown to offer equal security to that of RSA with key size of 1~2Kbits [2]. The strength of ECC is that it is based on the discrete logarithm problem over points on an elliptic curve. To date, no significant breakthroughs have been made in determining weaknesses in the ECC algorithm. The fact that the problem appears so difficult to crack means that key sizes can be reduced in size considerably, even exponentially [2,5,8]. This advantage of ECC is being recognized recently where it is being incorporated in many standards. In 1999, the Elliptic Curve Digital Signature Algorithm was adopted by ANSI, and it is now included in the ISO/IEC 15946 draft standards. Other standards that include Elliptic Curves as part of their specifications are the IEEE P1363 (www.stdsbbs.ieee.org), Internet Engineering Task Force (www.ietf.cnri.reston.va.us), and the ATM Forum.

Several GF(p) ECC processors have been proposed in the literature recently [15,19,20,22-25]. The advantage of using dedicated processors for encryption and decryption is that it results in a considerable higher speed when compared to a software solution on a general-purpose programmable processor. It also provides higher security than software solutions [16], which is shown increased by proper parallelization.

It is well known that adding two points over an elliptic curve would require an inversion operation, which is the most expensive operation over GF(p) [16,17]. Many proposed processors are based on representing the elliptic curve points as projective coordinate points in order to eliminate division, hence inversion, operations [4,6,15-17,22,24,25]. There are many projective coordinates systems to choose from [1,9]. Our selection is the projection of *(x,y)* to *(X/Z,Y/Z)*, which is proven in [19] and [20] to give the best performance based on the hardware architecture of four parallel multipliers.

In this letter, we assume applying the algorithms using the hardware strategy of [19] and its *(X/Z,Y/Z)* elliptic curve (EC) projection algorithm. However, we exploit the parallelization and its affect on security considering the upper level of ECC computation, i.e. the binary method to calculate the multiples of an EC point, which is introduced next.

2. *Point Operation Algorithm:*

It will be assumed that the reader is familiar with the arithmetic over elliptic curves. For a good review the reader is referred to [9]. In brief, the basic operation for ECC is Scalar multiplication, which is the algorithm used for calculating EC-point: *kP* from EC-point: *P*. Scalar multiplication in the group of points of an elliptic curve is the analogous of exponentiation in the multiplicative group of integers modulo a fixed integer. The computation of *kP* can be done with the straightforward double-and-add method, since it

* *Professor M.K. Ibrahim is now at the School of Engineering and Technology, De Montfort University, Leicester LE1 9BH, UK, Email: ibrahim@dmu.ac.uk*

is known to be efficient and practical to implement in hardware [2,5,7,9,10]. This method is based on the binary representation of $k = (k_{n-1},\ldots,k_0)$ where k_{n-1} is the most significant bit of k. In fact, several scalar multiplication methods have been proposed in the literature. A good survey is presented by Gordon in [18].

Two scalar multiplication forms of binary methods are considered. Both techniques have identical interface and take similar number of iterations as summarized below:

Define: *n:* number of bits in k; k_i: the i^{th} bit of k
Input: k and P (a point on the elliptic curve).
Output: $Q=kP$ (another point on the curve).

Left to Right Algorithm (LtR-Alg)

```
1. if $k_{k-1} = 1$, then $Q:=P$ else $Q:=0;$
2. for $i = n-2$ down to $0;$
3.     { $Q := Q+Q$ ;
4.       if $k_i = 1$ then $Q:= Q+P$ ; }
5. return $Q$;
```

Right to Left Algorithm (RtL-Alg)

```
1. $Q:=0;$
2. for $i = 0$ to $n-1;$
3.     if $k_i = 1$ then $Q:= Q+P$ ;
4.     $P := P+P$ ;
5. return $Q$;
```

The basic operations in all scalar multiplication algorithms are point addition and point doubling over an elliptic curve. Both binary methods need to double a point in all n iterations, where n is the number of binary bits of k. The algorithms scan the bits of k; whenever, a particular bit of k is found to be one, an extra operation is needed. This extra operation is EC point addition $(Q+P)$. The left to right algorithm (LtR-Alg) has received more attention due to its efficiency for sequential computation. The problem is that it becomes open to Simple Power Attacks (SPA) that will reduce the security of ECC. However, the right to left algorithm (RtL-Alg) is more secure and suitable for our parallel execution since the doubling can be carried out in parallel with the point addition [5,8] as will be clarified in the following section.

3. *Simple Power Attacks:*

Now, here we show that monitoring power consumption during the computation of kP knowing P may enable to recover k. Power consumption enables to visually identify large features, such as, the main iteration loop in the binary algorithms LtR-Alg and RtL-Alg. Power consumption analysis may also enable to distinguish between instructions being executed, i.e., it is possible to distinguish between point doubling and point addition in the algorithms, thus revealing the bits of the exponent k.

Coron in [10] showed that for algorithms LtR-Alg and RtL-Alg, in order to be unaffected by SPA, the instructions performed during a cryptographic algorithm should not depend on the data being processed. There should not be any branch instructions conditioned by the data. This could be done by performing the addition and doubling each time and then at the end of the loop decide either to accept the result or to eliminate the addition part according to k_i value. In this work, we adopt this idea by using RtL-Alg. However, we parallelize the EC point addition and doubling to gain the best speed, which will also insure the system immunity against SPA.

The next section describes EC projective coordinates point operations. Since, as mentioned earlier, point operations (EC point doubling and adding) requires complex inversion [16,17], it can almost be eliminated using projective coordinates[1,9], as clarified in the following section.

4. *Projective Coordinate Procedures:*

For elliptic curve defined over $GF(p)$, two different forms of formulas are found [1,9] for point addition and doubling. One form projections $(x,y)=(X/Z^2, Y/Z^3)$ [9], while the second projects $(x,y) = (X/Z,Y/Z)$ [1]. The projection of $(x,y) = (X/Z,Y/Z)$ is found better for speed, especially when four parallel multipliers are adopted [19,20]. The dataflow graphs for EC point adding is shown in Fig.1. Each EC point addition operation needs four steps with full utilization of all multipliers. Similarly, the EC point doubling is shown in Fig. 2, where it needs three steps for each doubling operation.

5. *Vulnerable Parallelization:*

As stated previously, using the right to left binary algorithm (RtL-Alg), point doubling and point additions can be carried out in parallel. However, as will be shown in this section, higher degree of parallelism can be achieved when the concurrency within each doubling and point additions operations is exploited. To achieve this, the dataflow graphs of point addition and doubling operations is to be revisited.

Observe the dataflow graphs of Fig. 1 and Fig. 2, when four multipliers are used to perform RtL-Alg, the total time for each RtL-Alg computation is approximated: $T_{total} = 3\,n + 4\,x$, where n is the number of binary bits of k, and x is the number of bits in k with values one. Note that only the GF(p) multiplication time is considered while the addition and subtraction

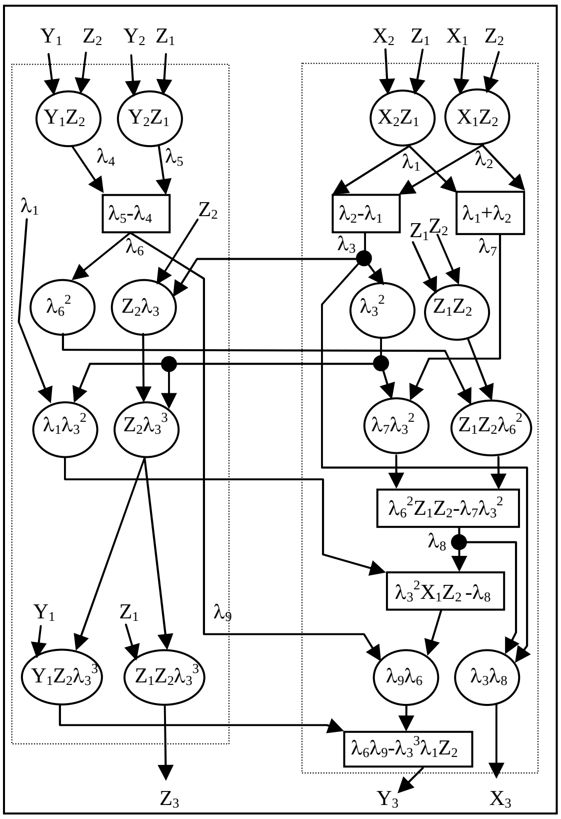

Fig. 1 Adding two points data flow

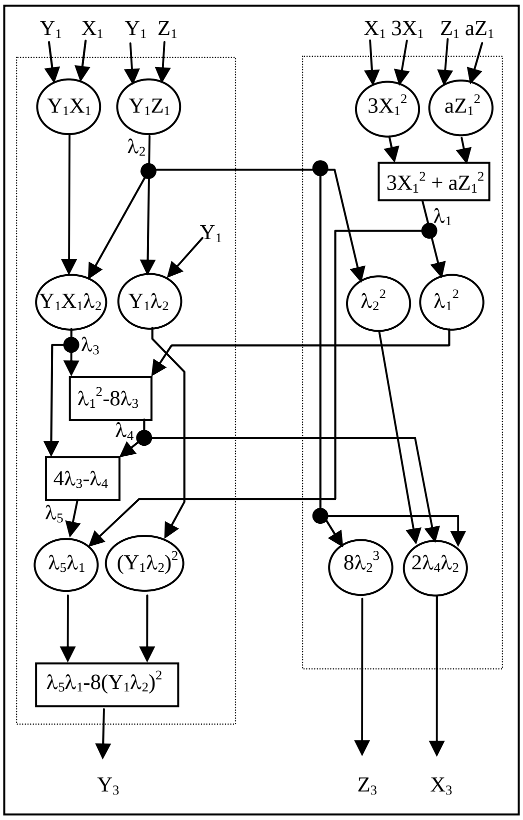

Fig. 2 Doubling a point data flow graph

time is ignored as clarified in [19,20]. We can form the normalized computation time for a single bit per one RtL-Alg operation as: $T = 3 + 4\tau$, where τ is the ratio x/n and has a value $0 < \tau < 1$.

For the worst case scenario: $\tau = 1$ ➔ $T = 7$. For the average case assuming half the bits of k are ones: $\tau = 0.5$ ➔ $T = 5$. Since the time depends on the data factor τ, someone can study the time and power of each iteration within RtL-Alg and extract the value k which is making the system not very secure.

Even if two multipliers are used instead of four, to compromise in area as suggested in [20], the critical paths will be affected and the normalized computation time for a single bit per one multiplication is going to be: $T=6+8\tau$, This two multipliers hardware worst case scenario will be: $\tau = 1$ ➔ $T = 14$, and the average case: $\tau = 0.5$ ➔ $T = 10$. Making the design with two multipliers and running the procedures normally will just double the time, which is a usual time area trade-off. However, it is making the security worse because the factor of data dependency τ increased allowing the SPA to be simpler.

6. *Secure Parallelization:*

The complete security benefit of the RtL-Alg is obtained if the data dependency factor τ is omitted. One way to perform this is to use the four multipliers design of [20] but dedicating two multipliers for each EC operation, i.e. two multipliers for point addition and two for doubling.

Both operations are performed in parallel, then, the k_i bit is checked to control the use of the EC point addition result. The normalized computation time for a single bit per one RtL-Alg operation will be: $T = 8$; since the point addition operation will dominate the timing.

To gain the full benefit of parallelization speed and security, two hardware designs with four multipliers each are used in parallel, i.e. an eight multiplier design. The critical path in this hardware will be dominated be EC point addition dataflow graph. The normalized computation time for a single bit per one RtL-Alg operation will be: $T = 4$, which is fixed and not affected by the k value.

7. *Security and Speed Trade-off:*

In the full eight multipliers parallelization attempt, we are performing the EC point addition in every RtL-Alg iteration although it may be unneeded, i.e. even if k_i is zero. This is to insure security against SPA. If this security constraint is released, the eight multiplier hardware can have better speed by checking k_i bit immediately after completing the doubling operation. If k_i is zero, the addition operation can be terminated. The normalized computation time for a single bit per one RtL-Alg operation will be: $T = 3 + \tau$, which is a compromise between speed and security.

8. Conclusion

This letter explores the ability to gain security from performing GF(p) elliptic curve cryptography computation in parallel methods. The best security can be achieved when making the design computation independent to the data, which can be obtained both elliptic curve addition and doubling are performed in parallel. The best security and speed performance is achieved when using full parallelization of the elliptic curve point operations, which can be attained with eight parallel multipliers in hardware. Implementing these parallelisms seems complex because of its large hardware requirement, however, it worth consideration especially because of its interesting benefits in security and speed performance.

Acknowledgments

Thanks to King Fahd University of Petroleum and Minerals for supporting all research work.

References

[1] Miyaji A.,: Elliptic Curves over F_P Suitable for Cryptosystems, Advances in cryptology-AUSCRUPT'92, Australia, December 1992.

[2] Stallings, W. "Cryptography and Network Security: Principles and Practice", Second Edition, Prentice Hall Inc., New Jersey, 1999.

[3] Chung, J., Sim, S., and Lee, P.,: Fast Implementation of Elliptic Curve Defined over GF(p^m) on CalmRISC with MAC2424 Coprocessor, Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000, Massachusetts, August 2000.

[4] Okada, S., Torii, N., Itoh, K., and Takenaka, M.,: Implementation of Elliptic Curve Cryptographic Coprocessor over GF(2^m) on an FPGA, Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000, Massachusetts, August 2000.

[5] Crutchley, D. A.,: Cryptography and Elliptic Curves, Master Thesis under Supervision of Prof. Gareth Jones, submitted to the Faculty of Mathematics at University of Southampton, England, May 1999.

[6] Orlando, G., and Paar, C.,: A High-Performance Reconfigurable Elliptic Curve Processor for GF(2^m), Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000, Massachusetts, August 2000.

[7] Stinson, D. R.,: Cryptography: Theory and Practice, CRC Press, Boca Raton, Florida, 1995.

[8] Paar, C., Fleischmann, P. and Soria-Rodriguez, P.,: Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents, IEEE Transactions on Computers, Vol. 48, No. 10, October 1999.

[9] I. Blake, G. Seroussi, N. Smart,: Elliptic Curves in Cryptography, Cambridge University Press: New York, 1999.

[10] D. Hankerson, J. Hernandez, and A. Menezes, Software Implementation of Elliptic Curve Cryptography Over Binary Fields, Workshop on Cryptographic Hardware and Embedded Systems, CHES 2000, Massachusetts, August 2000.

[11] G. A. Orton, M. P. Roy, P. A. Scott, L. E. Peppard, and S. E. Tavares,: VLSI implementation of public-key encryption algorithms, Advances in Cryptology -- CRYPTO '86, Vol 263 of Lecture Notes in Computer Science, (1987) pp. 277-301.

[12] Norman R. Scott,: Computer Number Systems and Arithmetic, Prentice-Hall Inc., NJ, 1985.

[13] R. J. Tocci, and N. S. Widmer, Digital Systems: Principles and Applications, Eighth Edition, Prentice-Hall Inc., New Jersey (2001).

[14] M. D. Ercegovac, T. Lang, and J. H. Moreno,: Introduction to Digital System, John Wiley & Sons, Inc., New York (1999).

[15] G. Orlando and C. Paar,: A scalable GF(p) elliptic curve processor architecture for programmable hardware, Cryptographic Hardware and Embedded Systems, CHES 2001.

[16] Adnan Gutub, Tenca, and Koc,: Scalable VLSI architecture for GF(p) Montgomery modular inverse computation, IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2002.

[17] Adnan Abdul-Aziz Gutub and Alexandre F. Tenca,: Efficient Scalable VLSI Architecture for Montgomery Inversion in GF(p), Integration, the VLSI Journal, Vol. 37, No. 2, May 2004.

[18] Gordon D.: A Survey of Fast Exponentiation Methods, Journal of Algorithms, 1998.

[19] Adnan Abdul-Aziz Gutub,: VLSI Core Architecture For GF(P) Elliptic Curve Crypto Processor, IEEE 10th International Conference on Electronics, Circuits and Systems (ICECS 2003), pages 84-87, University of Sharjah, United Arab Emirates, December 14-17, 2003.

[20] Adnan Abdul-Aziz Gutub and Mohammad K. Ibrahim,: High Radix Parallel Architecture For GF(p) Elliptic Curve Processor", IEEE Conference on Acoustics, Speech, and Signal Processing, ICASSP 2003, Hong Kong, April 6-10, 2003.

[21] Coron, J.-S.,: Resistance against differential power analysis for elliptic curve cryptosystems, In Cryptographic Hardware and Embedded Systems - CHES '99, 1999.

[22] Adnan Abdul-Aziz Gutub, "Fast 160-Bits GF(p) Elliptic Curve Crypto Hardware of High-Radix Scalable Multipliers)", *International Arab Journal of Information Technology (IAJIT),* Vol. 3, No. 4, Pages: 342-349, October 2006.

[23] Turki F. Al-Somani, M. K. Ibrahim and Adnan Gutub, "High Performance Elliptic Curve GF(2^m) Crypto Processor", *Information Technology Journal (ITJ)*, Vol. 5. No. 4, Pages: 742-748, 2006.

[24] Turki F. Al-Somani, M. K. Ibrahim and Adnan Gutub, "Highly Efficient Elliptic Curve Crypto-Processor with Parallel GF(2^m) Field Multipliers", *Journal of Computer Science (JCS)*, Vol. 2, No 5. Pages: 395-400, 2006.

[25] Adnan Abdul-Aziz Gutub, "Merging GF(p) Elliptic Curve Point Adding and Doubling on Pipelined VLSI Cryptographic ASIC Architecture", *International Journal of Computer Science and Network Security (IJCSNS)*, Vol.6, No.3A, March 2006.