Issues in Elliptic Curve Cryptography Implementation

Marisa W. Paryasto, Kuspriyanto, Sarwono Sutikno and Arif Sasongko
School of Electrical Engineering and Informatics
Institut Teknologi Bandung (ITB), Indonesia

***Abstract*—This work discusses issues in implementing Elliptic Curve Cryptography (ECC). It provides a brief explanation about ECC basic theory, implementation, and also provides guidance for further reading by referring each sub topics with more specific papers or books. The future and research topics in ECC will also be discussed.**

***Index Terms*—cryptography, security**

I. INTRODUCTION

THE rapid growth of computer applications for exchanging information electronically has resulted in the elimination of physical ways for providing security through locks, sealing and signing documents. This has thus resulted in the need for techniques for securing electronic document transactions. The techniques used are usually encryption and digital signatures.

The science of keeping messages secure is called cryptography. Cryptography involves encryption and decryption of messages. Encryption is the process of converting a plaintext into cipher text by using an algorithm, while decryption is the process of getting back the encryption message. A cryptographic algorithm is the mathematical function used for encryption and decryption.

Implementing cryptography involves extensive math and effective engineering and also good algorithm to integrate both. Deep math knowledge without efficient implementation techniques and effective implementation without solid foundation on math would not result in a product that can be delivered to solve problem [14].

Cryptographic systems can be broadly divided into two kinds: *symmetric-key cryptography* and *asymmetric-key cryptography* (public-key cryptography). The major advantage of symmetric-key cryptography is high efficiency, but it has a number of significant drawbacks, namely key distribution, key management, and the provision of non-repudiation.

Public-key cryptography provides an elegant solution to the problems inherent in symmetric-key cryptography. Unfortunately, public-key operations are usually significantly slower than symmetric key operations. Hence, hybrid systems that benefit from the efficiency of symmetric-key algorithms and the functionality of public-key algorithms are often used.

The notion of public-key cryptography was introduced in 1975 by Diffie, Hellman and Merkle [3] to address the aforementioned shortcomings of symmetric-key cryptography. In contrast to symmetric-key schemes, public-key schemes require only that the communication entities exchange keying material that is authentic (but not secret). Each entity selects a single key pair *(e, d)* consisting of a public-key *e*, and a related private-key *d* (that the entity keeps secret). The keys have the property that it is computationally infeasible to determine the private key solely from knowledge of the public key.

Elliptic curve cryptography (ECC) [7][11] is an emerging type of public key cryptography that presents advantages compared to other public key algorithms.

Currently ECC is the most efficient public key cryptosystem that uses shorter keys while providing the same security level as the RSA cryptosystem [16]. The use of shorter keys implies lower space requirements for key storage and faster arithmetic operations. These advantages are important when public-key cryptography is implemented in constrained devices, such as in mobile devices.

ECC is more complex than RSA. Instead of a single encryption algorithm (as in RSA), ECC can be implemented in different ways. ECC uses arithmetic algorithms as the core operations for high level security functions such as encryption (for confidentiality) or digital signatures (for authentication).

Cryptography implementation of this kind imposes several challenges, which may require a trade-off in performance, security and flexibility. ECC can be implemented in software or hardware. Software ECC implementations offers moderate speed and higher power consumption compared to custom hardware. Additionally, software implementations have very limited physical security, specially with respect to key storage.

If security algorithms are implemented in hardware, a gain in performance is obtained at cost of flexibility. Dedicated hardware implementations of cryptographic algorithms with low power consumption are expected to outperform the software implementations due to the fact that the instruction set of a processor does not directly implement specific cryptographic functions.

In addition, hardware implementations of cryptographic algorithms are more secure because they cannot be easily read

Marisa W. Paryasto is a PhD candidate (S3) at the School of Electrical Engineering and Informatics at ITB in Bandung, Indonesia. She can be reached at marisa@stei.itb.ac.id. Kuspriyanto (kuspriyanto@yahoo.com), Sarwono Sutikno (ssarwono@gmail.com) and Arif Sasongko (asasongko@gmail.com) are faculty members within the School of Electrical Engineering and Informatics at ITB.

or modified by an outside attacker. ASIC (Application Specific Integrated Circuit) implementations show lower price per unit, reach high speeds and have low power dissipation. However, ASIC implementations lack flexibility with regards to the algorithms and parameters. This leads to higher development costs when switching algorithms or schemes.

ECC interoperability is better achieved by software implementations. Software has the flexibility of allowing the switching among different ECC schemes with several security levels. However, the downside is that the performance of software implementations is lower. An approach studied in recent years combines the advantages of software (flexibility) and hardware (performance) in a new paradigm of computation referred to as *reconfigurable computing* (RC)[12]. RC involves the use of reconfigurable devices for computing purposes. The concept can be used to implement several applications but the general design methodology is not applicable to all cases.

II. Elliptic Curve Cryptography Foundation

ECC has a very unique mathematical structure that enables the process of taking any two points on a specific curve, of adding the two points and getting as a result another point on the same curve. This special feature is advantageous for cryptography due to the inherent difficulty of determining which original two points were used to get the new point. The choice of various parameters in the equation will set the level difficulty exponentially as compared to the key length.

Breaking encryption with ECC must use very advanced mathematics. However, ECC itself only require small increase in the number of bits in its keys in order to achieve a higher security.

ECC consists of a few basic operations and rules that define how addition, subtraction, multiplication, and doubling are performed.

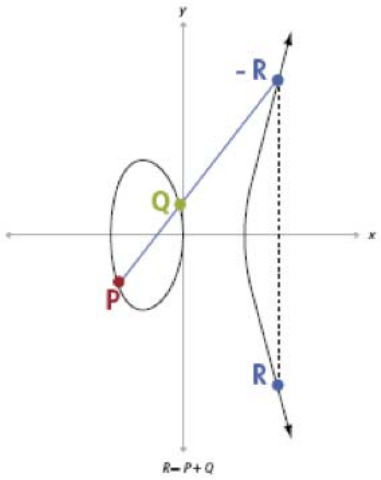

Figure 1 ECC Point Addition

Figure 1 illustrates one particular operation in ECC using real numbers. ECC point addition is defined as finding the line between two points, in this case P and Q. The result is a third point R. Point multiplication *kP* is accomplished by performing multiple additions. An example id the repeated point addition and doubling for 9P = 2(2(2P)) + P.

The public-key operation is $Q(x, y) = k\,P(x, y)$, with:
Q = public key
P = base point (curve parameter)
k = private key
n = order of P

Thus, the elliptic curve discrete logarithm is the following: given public key kP, find the private key k. The work of [6] gives a comprehensive explanation about elliptic curve mathematical foundation and its implementation.

III. ECC Implementation Issues

The most time consuming operation in ECC cryptographic schemes is the scalar multiplication. Efficient hardware/software implementations of the scalar multiplication *kP* have been the main research topic on ECC in recent years. This costly elliptic curve operation is performed according to the three layers shown in Figure 2 [12].

The scalar *k* can have different representation, and in the upper layer there are several algorithms to perform the multiplication. In the middle layer, there are several combinations for finite field representation and coordinates system. This layer covers curve operations, while the lower level is about finite field operations/arithmetic.

There are many algorithms can be applied for each layers, and the combination of algorithm used in each layer can significantly affect the performance of the scalar multiplication.

Figure 3 shows an example selection for the middle layer. An elliptic curve can be defined with different underlying fields.

An efficient implementation of an ECC over binary Galois fields in normal and polynomial bases has been proposed by Estes and Hines [4].

Other implementations and analysis over polynomial basis and ONB have also been done earlier by Choi *et al.* [2]. A non-conventional basis of finite fields for implementing a fast communication between two elliptic curve cryptosystems in software and hardware has been proposed by Sang Ho Oh *et al.* in [13]. This was done to address the problem of different choices of the basis. Sunar, Savas and Koc have constructed composite field representations for efficient conversion between binary and composite fields by deriving the change of the basis matrix [19].

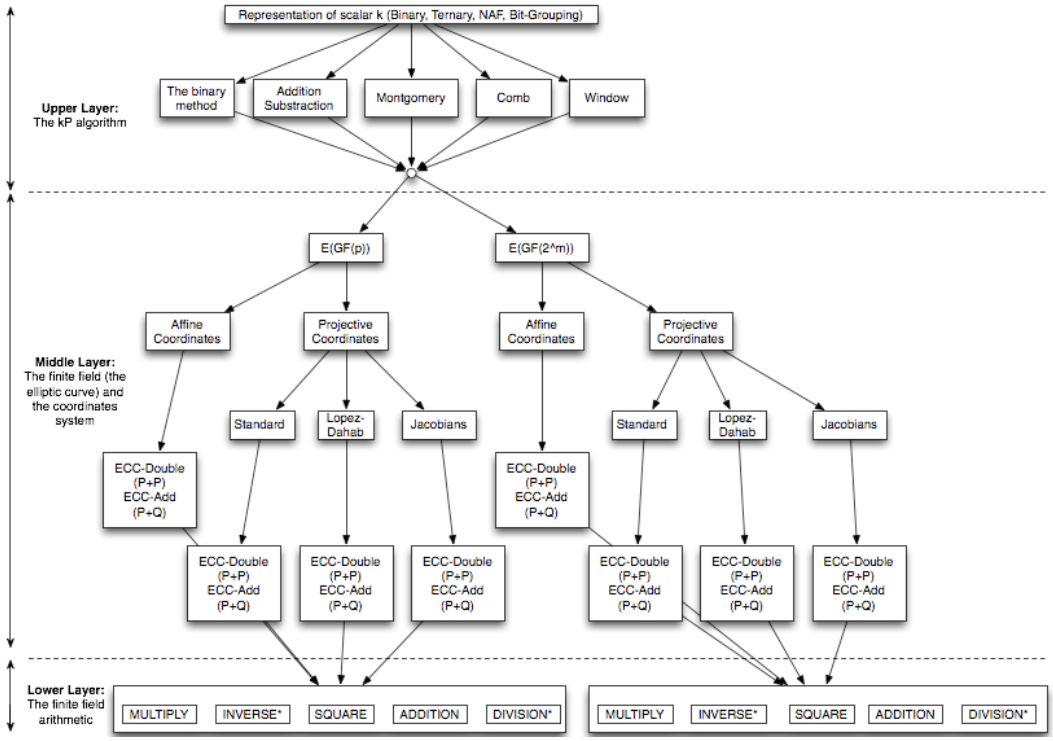

Figure 2 Various Methods of Scalar Multiplication kP

For the lower layer multiple works have been reported: efficient multiplication beyond optimal normal bases [15], Montgomery multiplication in *GF* (2^m) [9], Mastrovito multiplier for all trinomials [8], hardware implementation of *GF* (2^m) arithmetic using normal basis [20] and systematic design of original and modified Mastrovito multipliers for general irreducible polynomials [21].

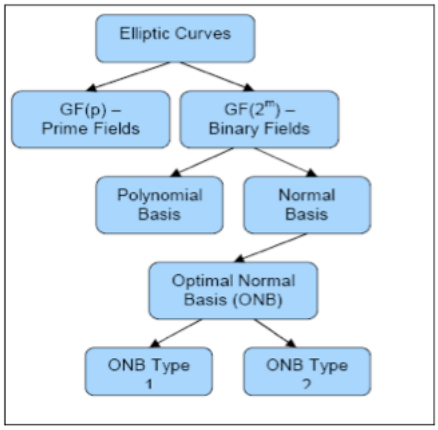

Figure 3 Taxonomy of Elliptic Curves

A. Software Implementations

ECC implementations in general purpose processor and embedded systems can meet some security requirements in some applications, but on the other hand hardware implementations are needed due to the application requirements such as throughput, power consumption, area constrains and physical security.

One of the comprehensive software implementations of ECC is [17]. Other ECC software implementation has been done by [18]. Software implementation in prime fields has been done by [1], while implementation in binary fields has also been done by [5].

B. Hardware Implementations

ECC hardware implementations are aimed to optimize each stage of scalar multiplication *kP* in Figure 2:

(1) Field optimization is performed by choosing fields with fast multiplication and inversion;

(2) Coordinates and scalar multiplication optimizations are performed by reducing the number of field inversions (projective coordinates), reducing the number of point additions (windowing) and by replacing point doubles (endomorphism methods).

Several works reported in the literature have used reconfigurable devices, FPGAs, to implement ECC algorithms. To compute the scalar multiplication *kP* as fast as possible is the main objective of these works. Hardware architectures for computing *kP* reported in the literature can be divided into processor or co-processor approaches. In the former, there exist a number of specialized instructions which the processor decodes and executes; most of them are for

elliptic curve and finite field arithmetic. In the latter, there are no such instructions because the algorithms are implemented directly on specialized hardware. In general, both kinds of implementations are based on a regular structure.

Many considerations must be taken into account when these blocks are implemented in hardware, especially for wireless applications where area/performance trade off is important. While a custom implementation can perform the operation *kP* faster, such custom work is difficult to change in order to support a different algorithm.

There is a diversity of technologies used to implement elliptic curve cryptography in hardware. So far a generic architecture suitable for mobile devices has not been reported. What has been reported are the differences regarding resources and timing achieved for different selections of the ECC parameters [12].

Recent works also show different approaches to implementing the three layers of *kP* computation. It has been reported that several multipliers have been used at different levels of parallelism. It is not clear how these choices will impact the resources of the coprocessor and the advantages gained by using one of the reported multipliers (namely Karatsuba, LFRS, Massey Omura, and D-serial). The same is also true for the inversion and square algorithms.

IV. Future of ECC and Research Topics

The implementation of cryptographic systems presents several requirements and challenges, particularly for constrained environments (memory and area requirements). An important aspect is the power and energy consumption relating to public key algorithms. This is especially a challenge in pervasive devices running on their own energy storage and which are placed in the field for long periods of time without any maintenance or possible physical access. For example, devices like RF-ID makes replacing the batteries a highly cumbersome process. RF-ID tag applications derive the required power from the electromagnetic field of the reader to run its applications. Such systems also have to be extremely power efficient. Therefore, real world estimates of the power requirements for cryptographic processes are extremely important. This includes systems running public-key cryptography on processors with extensions. The underlying arithmetic algorithms could then be chosen and fine-tuned more efficiently for a low power ECC design.

Unlike traditional systems that cannot be physically accessed by an attacker, pervasive systems must also consider physical security as they are placed in insecure surroundings easily accessible for tampering. Therefore, storing the private key securely on such devices remains a big challenge, with the usual solutions remaining too expensive for such low-cost devices.

Even when physically secure, these devices can be passively attacked using side-channel (time and power) methods. Well know side-channel resistant algorithms normally require almost double the execution time, with larger memory and hardware resources. These measures are unsuitable for such low-end devices that require highly optimized implementation (in time, memory and power) and therefore are an open problem that need further investigation [10].

V. Conclusion

ECC is a promising candidate for the next generation public key cryptosystem. Although ECC's security has not been completely evaluated, it is expected to come into widespread use in various fields in the future because of its compactness and high performance when it is hardware-implemented. In general we can conclude that the reliability, maturity and difficulty of a mathematical problem are very important factors.

ECC has been proven to involve much less overheads when compared to RSA. The ECC has been shown to have many advantages due to its ability to provide the same level of security as RSA yet using shorter keys. However, its disadvantage – which may lessen its attractiveness – is its lack of maturity, as mathematicians believe that not enough research has been done in the ECDLP.

References

[1] M. Brown, D. Hankerson, J. Lopez, and A. Menezes. Software Implementation of the NIST Elliptic Curves over Prime Fields.

[2] Yong-Je Choi, Moo-Seop Kim, Hang-Rok Lee, and Ho-Won Kim. Implementation and analysis of elliptic curve cryptosystems over polynomial basis and onb. In Proceedings of World Academy of Science, Engineering and Technology, volume 10, December 2005.

[3] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. IEEE International Symposium on Information Theory, June 1976.

[4] Matthew Estes and Philip Hines. Efficient implementation of an elliptic curve cryptosystem over binary galois fields in normal and polynomial bases. Technical Report GMU EE-746 Fall 2006, George Mason University, 2006.

[5] Darrel Hankerson, Julio Lopez Hernandez, and Alfred Menezes. Software Implementation of Elliptic Curve Cryptography Over Binary Fields.

[6] Darrel Hankerson, Alfred Menezes, and Scott Vanstone. Guide to Elliptic Curve Cryptography. Springer-Verlag New York, Inc., 2004.

[7] Neal Koblitz. Elliptic curve cryptosystems. Mathematics of Computation, 48(177): 203–209, January 1987.

[8] C. K. Koc and Berk Sunar. Mastrovito multiplier for all trinomials. IEEE Transactions on Computers, 1999.

[9] Cetin K. Koc and Tolga Aca. Montgomery Multiplication in GF(2k), volume 1, pages 57–69. Kluwer Academic Publishers, Boston, April 1998.

[10] Sandeep S. Kumar. Elliptic Curve Cryptography for Constrained Devices. Verlag Dr. Muller, 2008.

[11] Victor S. Miller. Use of elliptic curve cryptography. Technical report, Exploratory Computer Science, IBM Research, P.O. Box 218, Yorktown Heights, NY 10598.

[12] Miguel Morales-Sandoval. An interoperable and reconfigurable hardware architecture for elliptic curve cryptography. PhD thesis, National Institute for Astrophysics, Optics and Electronics - Tonantzintla, Puebla - Mexico, September 2006.

[13] Sang Ho Oh, Chang Han Kim, Joong Chul Yoon, Hee Jin Kim, and Jong In Lim. Non-conventional basis of finite fields - implementing a fact communication between two elliptic curve cryptosystems in software and hardware.

[14] Marisa W. Paryasto, Kuspriyanto, Sarwono Sutikno, and Arif Sasongko. ECC implementation: towards math and engineering integration. To be published, March 2009

[15] Arash Reyhani-Masoleh and M. Anwar Hasan. Efficient multiplication beyond optimal normal bases. IEEE, 52(4), April 2003.

[16] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21 (120–126), 1978.
[17] Michael Rosing. Implementing Elliptic Curve Cryptography. Manning Publications Co., 1999.
[18] Zhijie Jerry Shi and Hai Yan. Software implementations of elliptic curve cryptography. International Journal of Network Security, 7(2): 157–166, September 2008.
[19] Berk Sunar, Erkay Savas, and Cetin K. Koc. Constructing composite field representations for efficient conversion. IEEE Transactions on Computers, 52(11): 1391, November 2003.
[20] Alaaeldin Amin Turki F. Al-Somani. Hardware implementation of gf(2m) arithmetic using normal basis. Journal of Applied Sciences 6, 6:1362–1372, 2006.
[21] Tong Zhang and Keshab K. Parhi. Systematic design of original and modified mastrovito multipliers for general irreducible polynomials. IEEE Transactions on Computers, 50(7), July 2001.