Elliptic Curve Public-Key Cryptosystems – an Introduction

Erik De Win* and Bart Preneel**

Katholieke Universiteit Leuven, Dept. Electrical Engineering-ESAT
K. Mercierlaan 94, 3001 Heverlee, Belgium
{erik.dewin,bart.preneel}@esat.kuleuven.ac.be

Abstract. We give a brief introduction to elliptic curve public-key cryptosystems. We explain how the discrete logarithm in an elliptic curve group can be used to construct cryptosystems. We also focus on practical aspects such as implementation, standardization and intellectual property.

1 Introduction

Elliptic curves have been studied by mathematicians for more than a century. An extremely rich theory has been developed around them, and in turn they have been at the basis of new developments in mathematics, the proof of Fermat's last theorem being the most notable one. As far as cryptography is concerned, elliptic curves have been used for factoring [13] and primality proving [2].

Elliptic curve public-key cryptosystems (ECPKCs) were proposed independently by Victor Miller [17] and Neil Koblitz [12] in the mid-eighties. As with all cryptosystems, and especially with public-key cryptosystems, it takes years of public evaluation before a reasonable level of confidence in a new system is established. ECPKCs seem to have reached that level now. In the last couple of years, the first commercial implementations are appearing, as toolkits but also in real-world applications, such as email security, web security, smart cards, etc. An overview of theoretical and practical aspects of ECPKCs can be found in [14].

In the remainder of this article, we first describe the elliptic curve group and explain how public-key cryptosystems can be based on it. Then we continue with some more practical aspects, such as implementation, standardization and patents.

2 The Elliptic Curve Group

An elliptic curve is the set of solutions of an equation of the form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ .$$

* F.W.O.-Flanders research assistant, sponsored by the Fund for Scientific Research – Flanders (Belgium).
** F.W.O.-Flanders postdoctoral researcher, sponsored by the Fund for Scientific Research – Flanders (Belgium).

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

An equation of this kind can be studied over various mathematical structures, such as a ring or a field. For our purposes, we will only consider elliptic curves over a field. In this case, the coefficients a_i are elements of the field, and we look at all pairs (x, y) that satisfy the equation, with x and y in the field.

Figure 1 gives an example of an elliptic curve over the field of real numbers $\mathbb{R}$. This graph can be obtained by filling in values for x, and solving a quadratic equation in y. In this particular case, the graph consists of two distinct parts, but this is not always the case.

It turns out that the set of solutions of an elliptic curve has some interesting properties. In particular, a *group* operation can be embedded in this set, and in this way, we obtain a group, which enables us to do cryptography, as we will see in Sect. 3.

A group is a mathematical structure, consisting of a set of elements G, with a group operation $\diamond$ defined on the elements of the set. In order to have a group, the operation $\diamond$ must satisfy the following properties:

- closure: for all x, y in G, $x \diamond y$ must be in G;
- associativity: for all x, y and z in G, we must have $(x \diamond y) \diamond z = x \diamond (y \diamond z)$;
- identity: there exists an e in G such that $x \diamond e = e \diamond x = x$ for all x;
- inverse: for all x there exists a y such that $x \diamond y = y \diamond x = e$.

If on top of that, we have the abelian property:

- abelian: for all x, y in G, we have $x \diamond y = y \diamond x$,

then we say that the group is abelian.

Is it possible to turn the set of points on an elliptic curve into a group? The answer is positive, but how should we define an operation on curve points, with the properties mentioned above? This can be done as follows. Take two points on the curve, denoted with P_1 and P_2, and construct the line through these two points. In the general case, this line will always have a third point of intersection with the curve. Now take this third point and construct a vertical line through it. The other point of intersection of this vertical line with the curve is defined as the *sum* of P_1 and P_2. If P_1 and P_2 are equal, then the line to be constructed in the first step is the tangent to the curve, which again has exactly one other point of intersection with the curve. This group law is illustrated in Fig. 1. Note that the group is abelian; additive notation is often used for the group law of an abelian group.

An important question that still needs to be answered is what the identity of the group is. For example, in Fig. 1, what point should be added to P_3 to obtain P_3 as the sum? We can only find an answer to this question if an extra *point at infinity* $\mathcal{O}$ is added to the curve, which lies infinitely far on the vertical axis. This point $\mathcal{O}$ is the identity of the elliptic curve group. Now, it can be shown that this operation is a properly defined group operation, although some of the requirements (e.g., associativity) are far from trivial to prove.

For applications in cryptography, it is not very practical to construct the result of a group operation in a graphical way. However, it is possible to find a

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

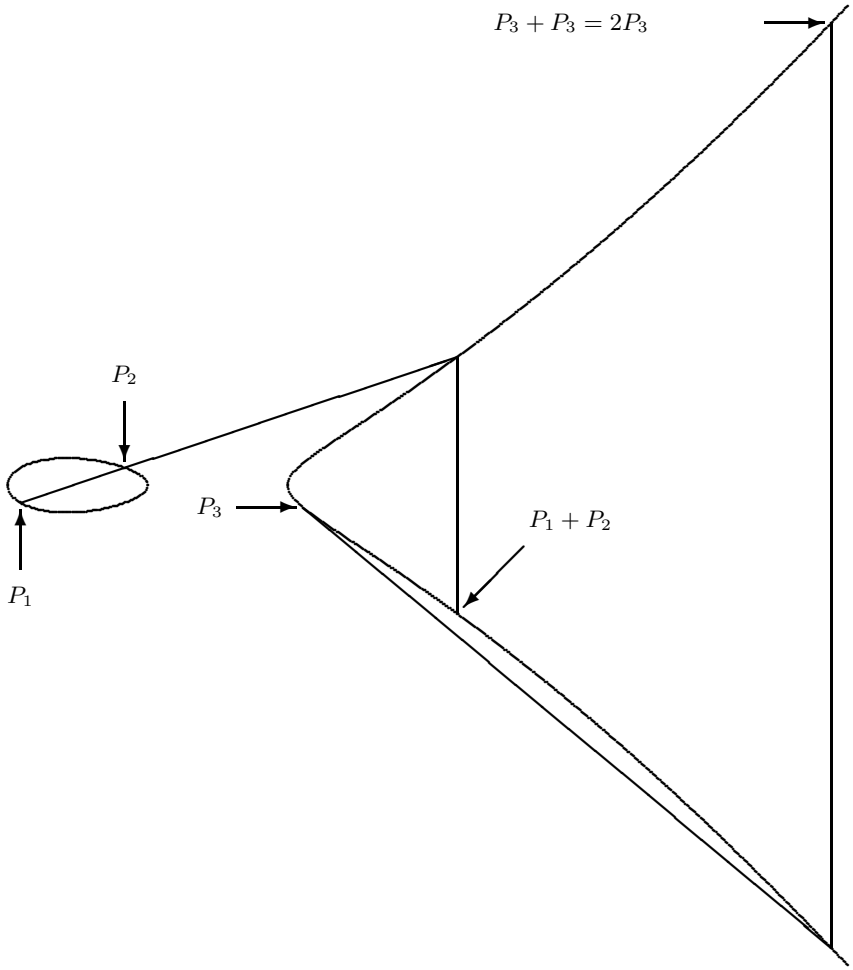

Fig. 1. Example of an elliptic curve over the reals, and demonstration of the group operation

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

closed formula that gives the coordinates (x_S, y_S) of the sum P_S of two points P_1 and P_2 as a function of their coordinates (x_1, y_1) and (x_2, y_2). For the case of real numbers, the curve equation can be simplified to the form

$$y^2 = x^3 + ax + b \ .$$

Then the formulas for the group operation are:

$$\begin{aligned} x_S &= \lambda^2 - x_1 - x_2 \ ; \\ y_S &= \lambda(x_1 - x_S) - y_1 \ ; \\ \lambda &= \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \text{ if } P_1 \neq P_2 \ ; \\ \dfrac{3x_1^2 + a}{2y_1} \text{ if } P_1 = P_2 \ . \end{cases} \end{aligned}$$

These equations hold except for the trivial cases where P_1 or P_2 are equal to the point at infinity, or where $-P_1$, which has coordinates $(x_1, -y_1)$, is equal to P_2.

A disadvantage of using the real numbers for cryptography is that it is very hard to store them precisely in computer memory, and to predict how much storage we will need for them. This problem can be solved by using *finite* fields, i.e., fields with a finite number of elements. Since the number of elements is finite, we can find a unique representation for each of them, which allows us to store and handle the elements in a manageable way. The number of elements of a finite field (or Galois field) is always a positive prime power p^n; the corresponding field is denoted $\mathrm{GF}(p^n)$. Two special cases are popular for use in ECPKCs: fields of the form $\mathrm{GF}(p)$ (or $n = 1$), and fields of the form $\mathrm{GF}(2^n)$, or $p = 2$. We will discuss both fields in more detail in Sect. 5. In a recent paper [3], the use of fields of the general form $\mathrm{GF}(p^n)$ is suggested; we will not discuss these fields here.

Finite fields are hard to represent in a graphical way, and hence we have to rely on formulas for the group operation. For $\mathrm{GF}(p)$, the formulas are the same as for the reals; for $\mathrm{GF}(2^n)$ they are slightly different (see Sect. 5).

3 Elliptic Curve Public-Key Cryptosystems

Many public-key cryptosystems are based on the *discrete logarithm problem* (DLP) in a group. We will give some examples for arbitrary groups, and then restrict our attention to elliptic curves.

The DLP in a group $G, \diamond$ can be defined as follows. Suppose that x and y are elements of G and that y was obtained by repeatedly applying the group operation to x, i.e., $y = x \diamond x \diamond \ldots \diamond x$, where the number n of terms in the righthand side of the equation is unknown. Alternatively, we note $y = x^n$. Then the DLP consists of finding n, knowing only x and y and the properties of the group.

How easy it is to solve the DLP depends on the properties of the group G. In a general group, without extra properties that allow for 'shortcuts', the

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

DLP is a *hard* problem, in the sense that the complexity of the best known algorithms is approximately equal to the square root of the size of the group. However, a number of interesting groups have additional structure that enable the application of faster algorithms, making these groups less suitable (but not necessarily useless) for cryptography.

The Diffie-Hellman key agreement protocol is an example of how a public-key cryptosystem can be constructed based on the DLP in a group (see also Fig. 2). Traditionally, the parties in the protocol are called Alice and Bob.

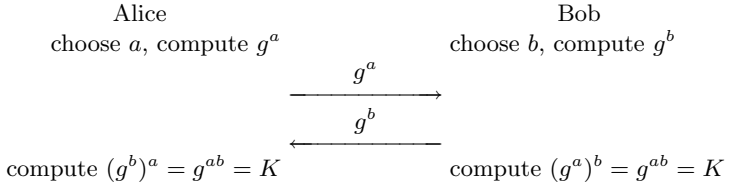

Fig. 2. Diffie-Hellman key agreement protocol

Setup Alice and Bob agree on a common group G and a common group element g. Then Alice chooses a secret number a, which serves as her secret key, and Bob chooses a secret key b.

Communication Alice computes g^a and sends it to Bob over a public channel; Bob sends g^b to Alice. Although g^a and g^b are closely related to a and b respectively, the hardness of the DLP ensures that the secret keys cannot be computed from them in a practical situation. Therefore, g^a and g^b can serve as public keys, corresponding to the private keys of Alice and Bob respectively.

Final step Alice takes Bob's public key and computes $(g^b)^a = g^{ab}$; Bob computes $(g^a)^b = g^{ab}$. As we see, Alice and Bob obtain the same result, and this result could not be computed by an adversary who only knows the public keys. Therefore Alice and Bob have agreed on a shared secret key.

Note that precautions need to be taken to make this protocol secure against both passive and active attacks. Describing them here would lead us to far; more details can be found in [16], for instance. Also, note that hardness of the DLP does not guarantee the security of the Diffie-Hellman protocol: computing g^{ab} from g^a and g^b may be easier than computing a from g^a or b from g^b.

Up to now, we have not specified the particular group we are working in. Interesting candidates are groups where the DLP is hard to solve. Examples of groups that have been used for about 20 years are the multiplicative groups of finite fields, in particular $\mathbb{Z}_p, \cdot$ and $\mathrm{GF}(2^n), \cdot$. A number of shortcuts have been found to solve the DLP in these groups in sub-exponential time, but they are still practical for use in cryptography. Examples of public-key systems based on

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

the DLP in $\mathbb{Z}_p, \cdot$ are ElGamal signatures, ElGamal encryption and DSA [16], where p is usually taken to be at least 1024 bits long for security reasons.

In the mid-eighties, Victor Miller [17] and Neal Koblitz [12] independently proposed to use the group of an elliptic curve over a finite field for public-key cryptography. No shortcuts were known to solve the elliptic curve DLP (ECDLP) faster than in time proportional to the square root of the group size, and up to now, this is still the case, except for a number of degenerated classes of curves [15, 21, 22].

All DLP-based cryptosystems can be converted to elliptic curves, and so we have ECDSA, EC Diffie-Hellman, EC ElGamal encryption, etc. Some slight technical modifications are necessary in order to adapt to the elliptic curve setting, but the underlying principles are the same as for other DLP-based systems.

4 Comparison to Other Public-Key Cryptosystems

How do elliptic curve public-key cryptosystems compare to other popular systems such as their counterparts based on the group $\mathbb{Z}_p, \cdot$ or to systems based on integer factorization?

A very important observation is that the best known algorithms for the ECDLP have a complexity proportional to the square root of the group size, whereas DLP in $\mathbb{Z}_p, \cdot$ and integer factorization can be solved in sub-exponential time. This implies that, for a certain level of security, the sizes of the parameters can be substantially smaller. It is hard to obtain accurate security estimates for EC, but experts in the field estimate that an elliptic curve group with a 175-bit size has a security that is equivalent to RSA with a 1024-bit modulus, or to systems based on DLP in $\mathbb{Z}_p, \cdot$ with p a 1024-bit prime [24].

The smaller block size has important implications on the resources that are needed to implement an ECPKC. For example, far less chip area is needed for an elliptic curve processor than for an RSA processor. In applications where bandwidth is an issue, elliptic curves can also offer an advantage. In elliptic curve key agreement protocols, the quantities transmitted are much shorter. An elliptic curve signature is only 340 bits long, as opposed to 1024 bits for an RSA-signature. Note however that DSA-signatures are only 320 bits long and that it is possible to reduce the overhead in storage of an RSA signature by recovering part of the message from the signature [11].

For a comparison of the performance of various public-key cryptosystems we refer to Sect. 5. The EC group operation is more complex than the core operations of other systems such as RSA, and therefore the smaller block size is not reflected proportionally in the performance figures.

5 Implementation Issues

In this section we will discuss some aspects of software implementations for ECPKCs. The time consuming operation in an ECPKC is the multiplication of

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

a point on a curve by an integer, i.e., the repeated group operation, also called EC exponentiation, since it is analogous to exponentiation in other discrete logarithm-based systems. An implementation of the repeated group operation consists of two levels of hierarchy: the group operation level, and the level of calculating a point multiplication using single group operations. We will discuss both levels in separate paragraphs.

Also, in Sect. 2 we saw that in cryptography elliptic curves over two kinds of fields are used: GF(p) or GF(2^n). We will discuss the group operation for both kinds of fields in separate paragraphs.

5.1 Elliptic Curve Group Operation over GF(p)

The formulas for the group operation on elliptic curves over GF(p) are given in Sect. 2. From these, the group operation can be calculated in terms of a number of field operations, such as addition, subtraction, multiplication and inversion. The field GF(p) is usually represented by the integers modulo p, and hence the field operations are modular operations, similar to the operations needed in other public-key cryptosystems such as RSA or DSA. Therefore, parts of the software libraries for the latter can be reused for elliptic curves over GF(p).

The inversion operation deserves some extra attention. In most public-key cryptosystems, modular inversion is not very important, but in ECPKCs it turns out to be the major bottleneck if the group operation is implemented in a straightforward way. Fortunately, it is possible to reduce the number of inversions by representing the points in *projective coordinates* [9].

Implementations of elliptic curves over GF(p) have been described in [18] and [5]. Table 1 gives typical timings for the field operations and the group operations for the GF(p) case.

5.2 Elliptic Curve Group Operation over GF(2^n)

The elliptic curve equations and formulas for finite fields GF(2^n) are slightly different. The curve equation can be simplified to

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

and the addition formulas are

$$\begin{aligned} x_S &= \lambda^2 + \lambda + x_1 + x_2 + a \ , \\ y_S &= \lambda(x_1 + x_S) + x_S + y_1 \ , \end{aligned}$$

with

$$\lambda = \begin{cases} \dfrac{y_2 + y_1}{x_2 + x_1} \text{ if } P_1 \neq P_2 \ , \\ x_1 + \dfrac{y_1}{x_1} \text{ if } P_1 = P_2 \ . \end{cases}$$

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

Contrary to GF(p), a number of different representations are commonly used for the elements of a field GF(2^n). The simplest representation is in *standard basis*, where the elements are binary polynomials of degree smaller than n, and calculations are done modulo an irreducible binary polynomial of degree exactly n. Another representation is based on *optimal normal bases* [19]. In this case, elements are represented as linear combinations of the elements of the set $\{\beta^{2^0}, \beta^{2^1}, \beta^{2^2}, \ldots, \beta^{2^{n-1}}\}$, where β is a suitably chosen element of the field. A third representation, sometimes called *tower field* representation, represents the elements as polynomials over the field GF(2^r), where r is a divisor of n.

A number of implementations of ECPKCs over GF(2^n) have appeared in literature, based on these three representations; see [4, 5, 7, 8, 23]. Table 1 gives typical timings for the field operations and the group operations for GF(2^n) using a standard basis.

Table 1. Timings for field operations over GF(p) and GF(2^n) on a PPro200. A standard basis is used for the latter. The field size is approximately 191 bits for both. All times in μs.

	GF(p)	GF(2^n)
addition	1.6	0.6
multiplication	7.8	39
squaring	7.6	2.6
inverse	180	126
EC addition	103	215
EC double	76	220

5.3 Point Multiplication

As soon as we have algorithms for the group operation available, we want to look for a method that computes a point multiplication using as little group operations as possible. This corresponds to the problem of computing an exponentiation using multiplications in other cryptosystems such as RSA or DSA.

A very simple algorithm for this is the square-and-multiply algorithm, see e.g., [10]. A number of optimizations to this algorithm are possible; [6] gives a good overview. An advantage of elliptic curves is that the inverse of a point can be computed very efficiently (see Sect. 2). A comparison of different algorithms using this optimization can be found in [5].

With the algorithms for point multiplication, we can start building various public-key schemes. Table 2 compares the efficiency of signature schemes based on elliptic curves to other signature schemes.

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

Table 2. Comparison of ECDSA to other signature algorithms. For EC, the field size is approximately 191 bits. The modulus for RSA and DSA is 1024 bits long; the RSA public exponent is 3. All times in ms, unless otherwise indicated. The timings were done on a PPro200.

times in ms	GF(p)	GF(2^n)	RSA	DSA
key generation	5.5	11.7	1 s	7
sign	6.3	11.3	43.3	7
verify	26	60	0.65	28.3
general point multiplication	21.1	56		

6 Standardization

It took a number of years before manufacturers of cryptographic products attained confidence in elliptic curve systems, but we seem to have reached that point now. A number of standardization bodies are undertaking efforts to create standards about elliptic curves. Although none of them have been officially approved yet, most are quite mature and not expected to change much before approval.

A broadly known initiative for a public-key standard, including ECPKCs, is the IEEE P1363 working group, to which the authors have also delivered a number of contributions. More information, including the latest draft and directions on how to join the mailing list, can be found at the web site [20].

The ANSI X9F1 working group is also working on a number of documents including elliptic curve-based systems, in particular X9.42 (DH/MQV key agreement), X9.62 (elliptic curve signatures) and X9.63 (elliptic curve key agreement/transport). More information can be found at [25].

Other organizations working on elliptic curve standards are ISO/IEC and the Internet Engineering Task Force.

7 Intellectual Property Issues

Contrary to RSA, the basic idea of ECPKCs has not been patented, and in the beginning this seemed to be an important advantage. In the past two years, a number of patents have been applied for on techniques that mostly aim at improving the efficiency. It should be noted that the situation is rather unclear since most of these patents are still pending. In principle, it should still be possible to construct a secure, albeit not extremely efficient, ECPKC without licensing patents.

A number of these techniques are being considered for inclusion in standards and this will potentially make it hard to implement interoperable elliptic curve systems without licensing patents. On the other hand, some standardization organizations require the holders of patents on standardized techniques to guarantee 'reasonable' licensing conditions. In summary, elliptic curves have lost

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

many of their advantages as far as patents are concerned, but the situation is probably not worse than for other popular public-key systems. However, this may change when the RSA patent expires on Sept. 20, 2000.

8 Conclusion

Elliptic curve public-key cryptosystems are well on their way to being a serious alternative to older public-key cryptosystems, in particular RSA. They can be implemented efficiently, and have a number of advantages that can make them the best choice in a range of applications (such as on-line communications and wireless communications). With a number of standards being in preparation, interoperability between the different products will be much easier to obtain. The patent situation is not as nice as it first appeared to be, but it should not be an important drawback either. One factor that still remains largely uncertain is the security: as with all practically used public-key cryptosystems, their security has not been proven but is based on the inability to find attacks. If attacks exist on any of these systems, they might be discovered sooner or later. In this case it is vital to be able to quickly switch to an alternative.

References

1. ANSI X9.62-199x: *Public-Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA),* November 11, 1997.
2. A. Atkin and F. Morain, "Elliptic curves and primality proving," *Mathematics of Computation*, Vol. 61 (1993), pp. 29–68.
3. D. Bailey and C. Paar, "Optimal Extension Fields for Fast Arithmetic in Public-Key Algorithms," *Advances in Cryptology, Proc. Crypto'98, LNCS 1462,* H. Krawczyk, Ed., Springer-Verlag, 1998, pp. 472–485.
4. E. De Win, A. Bosselaers, S. Vandenberghe, P. De Gersem and J. Vandewalle, "A fast software implementation for arithmetic operations in GF(2^n)," *Advances in Cryptology, Proc. Asiacrypt'96, LNCS 1163*, K. Kim and T. Matsumoto, Eds., Springer-Verlag, 1996, pp. 65–76.
5. E. De Win, S. Mister, B. Preneel and M. Wiener, "On the performance of signature schemes based on elliptic curves," *Proceedings of the ANTS III conference, LNCS 1423,* J. Buhler, Ed., Springer-Verlag, 1998, pp. 252–266.
6. D.M. Gordon, "A survey of fast exponentiation methods," *Journal of Algorithms,* Vol. 27, pp. 129-146, 1998.
7. J. Guajardo and C. Paar, "Efficient algorithms for elliptic curve cryptosystems," *Advances in Cryptology, Proc. Crypto'97, LNCS 1294*, B. Kaliski, Ed., Springer-Verlag, 1997, pp. 342–356.
8. G. Harper, A. Menezes and S. Vanstone, "Public-key cryptosystems with very small key length," *Advances in Cryptology, Proc. Eurocrypt'92, LNCS 658*, R.A. Rueppel, Ed., Springer-Verlag, 1993, pp. 163–173.
9. IEEE P1363: Editorial Contribution to Standard for Public-Key Cryptography, July 27, 1998.
10. D. Knuth, *The Art of Computer Programming, Vol. 2, Semi-Numerical Algorithms,* 2^{nd} *Edition*, Addison-Wesley, Reading, Mass., 1981.

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.

11. ISO/IEC 9796-2, *"Information technology – Security techniques – Digital signature schemes giving message recovery, Part 2: Mechanisms using a hash-function,"* 1997.
12. N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, Vol. 48, no. 177 (1987), pp. 203–209.
13. H.W. Lenstra Jr., "Factoring integers with elliptic curves," *Annals of Mathematics*, Vol. 126 (1987), pp. 649–673.
14. A. Menezes, *Elliptic Curve Public-Key Cryptosystems*, Kluwer Academic Publishers, 1993.
15. A. Menezes, T. Okamoto and S. Vanstone, "Reducing elliptic curve logarithms to logarithms in a finite field," *IEEE Transactions on Information Theory*, Vol. 39 (1993), pp. 1639–1646.
16. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
17. V.S. Miller, "Use of elliptic curves in cryptography," *Advances in Cryptology Proc. Crypto'85, LNCS 218*, H.C. Williams, Ed., Springer-Verlag, 1985, pp. 417–426.
18. A. Miyaji, T. Ono and H. Cohen, "Efficient elliptic curve exponentiation," *Proceedings of ICICS'97, LNCS 1334,* Y. Han, T. Okamoto and S. Qing, Eds., Springer-Verlag, 1997, pp. 282–290.
19. R. Mullin, I. Onyszchuk, S. Vanstone and R. Wilson, "Optimal normal bases in GF(p^n)," *Discrete Applied Mathematics*, Vol. 22 (1988/1989), pp. 149–161.
20. `http://grouper.ieee.org/groups/1363/`.
21. T. Satoh and K. Araki, "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves," *Commentarii Math. Univ. St. Pauli,* Vol. 47 (1998), pp. 81–92.
22. N. Smart, "The discrete logarithm problem on elliptic curves of trace one," preprint, 1998.
23. R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck, "Fast key exchange with elliptic curve systems," *Advances in Cryptology, Proc. Crypto'95, LNCS 963*, D. Coppersmith, Ed., Springer-Verlag, 1995, pp. 43–56.
24. M. Wiener, "Performance comparison of public-key cryptosystems," *CryptoBytes*, Vol. 4., No. 1, Summer 1998, pp. 1–5.
25. `http://www.x9.org/`.

Appeared in *State of the Art and Evolution of Computer Security and Industrial Cryptography*, Lecture Notes in Computer Science 1528, B. Preneel, and V. Rijmen (eds.), Springer-Verlag, pp. 132–142, 1998.