Applications and Benefits of Elliptic Curve Cryptography

Krists Magons

University of Latvia, Faculty of Computing, Raiņa bulvāris 19, Riga, LV-1586, Latvia
km10054@lu.lv

Abstract. This paper covers relatively new and emerging subject of the elliptic curve crypto systems whose fundamental security is based on the algorithmically hard discrete logarithm problem.
Work includes the study of the following issues: mathematical background of the elliptic curve crypto systems, discrete logarithm problem, practical use cases in the industry, common implementation mistakes, performance comparison of elliptic curve and RSA crypto systems etc.
The conclusion contains a brief summary of the elliptic curve cryptosystem practical applications, the potential practical benefits and disadvantages with respect to the widely used RSA crypto system.

1 Introduction

The origins of asymmetric cryptography are associated with *Whitfield Diffie* and *Martin Hellman* famous 1976 publication that launched the revolution in cryptography [1, 2]. That publication pointed out a number of algorithmically hard problems such as the discrete logarithm problem. Afterwards the foundation of modern public key cryptography was defined. One of the most significant results was the discrete logarithm problem which is used in several crypto systems like *DSA*, *ECDH*, as well as virtually in any elliptic curve based crypto system design. The problem is easily to define:

G – finite cyclic group , g – generator of G, $a \in G$, find natural number s, such as $g^s = a$, if s exists [1]. It is believed that such issue is an algorithmically hard problem, which means that there is no general algorithm that solves the discrete logarithm problem in polynomial time.

In this paper the author reviews the practical use of the elliptic curve public key crypto systems which are based on the discrete algorithm problem. Elliptic curves are studied for more than a century [3] and are used not only in cryptography, but also in the fields of computer science such as coding theory, pseudo-random number generation and others [3].

The origins of the elliptic curve cryptography date back to 1985 when two scientists *N. Koblitz* and *V. Miller* came up with the idea that it is possible to use the set of points defined by an elliptic curve over finite prime field in the crypto systems whose security is based on the discrete logarithm problem. Elliptic curve based crypto systems versus those crypto systems which are based on the integer factorization problem offer significant advantages because the known methods for computing the discrete logarithm

are not feasible to be practically used on the elliptic curve based crypto systems. One of the most important practical benefits is significantly reduced key sizes compared to other crypto systems. For instance, from the security standpoint elliptic curve based crypto system with key length of 163 bits is comparable to *RSA* based cryptosystem whose key length is equivalent to 1024 bits [4].

2 Elliptic Curves Over Finite Prime Field $\mathbb{F}_p$

Let $\mathbb{F}_p$ be a finite prime field that contains exactly p elements and p is an odd prime number. For each odd prime number p exists exactly one finite prime field $\mathbb{F}_p$, however, the representation of field elements may vary [5].

If $a, b \in \mathbb{F}_p$ and $4a^3 + 27b^2 \not\equiv 0 \,(mod\, p)$ then the elliptic curve over $\mathbb{F}_p$ is the following set of points [6]:

$E(\mathbb{F}_p) = \{(x,y)\} \in \mathbb{F}_p{}^2 | \, y^2 \equiv x^3 + ax + b \,(mod\, p) \wedge 4a^3 + 27b^2 \not\equiv 0 \,(mod\, p) \wedge a,b,y,x \in \mathbb{F}_p\} \cup \{O\}$, where O is the point at infinity

The number of elements $\#E(\mathbb{F}_p)$, in $E(\mathbb{F}_p)$ is equal to the number of points of elliptic curve over $\mathbb{F}_p$. According to the *Hasse Theorem* $\#E(\mathbb{F}_p)$ belongs to the interval [5] :

$$p + 1 - 2\sqrt{p} \leq \#E(\mathbb{F}_p) \leq p + 1 + 2\sqrt{p}$$

It is proved that the elements of $E(\mathbb{F}_p)$ form abelian group. The number of elements $\#E(\mathbb{F}_p)$ in $E(\mathbb{F}_p)$ is called the order of group. The order of group can be algorithmically determined by taking full scan of elements in $O(p)$ time, however, there are more efficient algorithms available, for instance, the *Schoof's* algorithm [7, 8].

2.1 The Algebraic Definition of Addition Operation in $E(\mathbb{F}_p)$

The addition operation in $E(\mathbb{F}_p)$ is defined by the following axioms [5]:

1. $O + O = O$
2. $\forall (x,y) \in E(\mathbb{F}_p) : (\mathrm{x}, \mathrm{y}) + O = O + (x,y) = (x,y)$
3. $\forall (x,y) \in E(\mathbb{F}_p) : (\mathrm{x}, \mathrm{y}) + (\mathrm{x}, -\mathrm{y}) = (\mathrm{x}, -\mathrm{y}) + (\mathrm{x}, \mathrm{y}) = O$
4. $(x_1,y_1) \in E(\mathbb{F}_p), (x_2,y_2) \in E(\mathbb{F}_p), x_1 \neq x_2, \, then \; (x_1,y_1) + (x_2,y_2) = (x_2,y_2) + (x_1,y_1) = (x_3,y_3)$
5. $x_3 \equiv \lambda^2 - x_1 - x_2 \,(mod\, p)$
6. $y_3 \equiv \lambda (x_1 - x_3) - y_1 \,(mod\, p)$
7. $\lambda \equiv \frac{y_2 - y_1}{x_2 - x_1} \,(mod\, p)$
8. $(x_1,y_1) \in E(\mathbb{F}_p) \, y_1 \neq 0 \, (x_1,y_1) + (x_1,y_1) = (x_4,y_4)$
9. $x_4 \equiv \lambda_2^2 - 2x_1 \,(mod\, p)$
10. $y_4 \equiv \lambda_2 (x_1 - x_4) - y_1 \,(mod\, p)$
11. $\lambda_2 \equiv \frac{3x_1^2 + a}{2y_1} \,(mod\, p)$

2.2 The Algebraic Definition of Scalar Multiplication Operation in $E(\mathbb{F}_p)$

Crypto systems based on elliptic curves over $\mathbb{F}_p$ largely utilize the scalar multiplication operation. Let P be a point of an elliptic curve over $\mathbb{F}_p$, $n \in N$, then the scalar multiplication is defined as addition of P n-times [8]. The result is denoted as nP. $nP = \sum_{i=1}^{n} P$, where addition is the $E(\mathbb{F}_p)$ addition operation. The scalar multiplication operation can be effectively carried out in $O(\log n)$ time by using the addition operation axioms and algorithms such as *Double and Add* algorithm [8].

2.3 Cyclic Subgroups of $E(\mathbb{F}_p)$

By examining an elliptic curve over fixed prime field $\mathbb{F}_p$, it is easy to observe that the set of all scalar multiplications of given point P forms a cyclic subgroup of $E(\mathbb{F}_p)$ [8, 9]. The point P is called the generator of subgroup. The order of subgroup is the smallest non-negative number $n \in N$ such that $nP = O$. It is not possible to use the *Schoof's* algorithm to find the order of subgroup [7, 8]. To find the order of cyclic subgroup it is required to use the *Lagrange's theorem* on subgroup order which states that for any finite group G, the order of every subgroup H of G divides the order of G [9].

In practice it is important to use cyclic subgroups with maximum possible order. In order to construct an elliptic curve based crypto systems, the curve E and underlying field $\mathbb{F}_p$ are fixed, the order of $E(\mathbb{F}_p)$ is calculated by using *the Schoof's* algorithm, then the largest maximum prime factor n of $\#E(\mathbb{F}_p)$ should be chosen as an order of cyclic subgroup [8]. To find a suitable group generator element (point on the curve E), calculate the cofactor $h = \#E(\mathbb{F}_p)/n$, randomly select an element H of $E(\mathbb{F}_p)$. If $hH \neq O$, then H is the generator of cyclic subgroup, otherwise repeat the previous step [8].

3 The Discrete Logarithm Problem

If all the points of an elliptic curve over finite prime field $\mathbb{F}_p$ form a group $E(\mathbb{F}_p)$ and P is one of the points of the curve and $n \in \mathbb{N}$, then the scalar multiplication nP is called the discrete exponent at base P and power n. Discrete exponents have significant properties similar to classic exponents. [10]. For the discrete logarithm problem based crypto systems the following property is very important:

$$(n+m)P = nP * mP, \text{ where } n, m \in \mathrm{N} \wedge P \in E$$

The question: given points $Q \in E(\mathbb{F}_p), P \in E(\mathbb{F}_p)$ find positive integer n such that $Q = nP$ if such n exists. As mentioned in previous chapters, the elliptic curve cryptography utilizes the cyclic subgroups of $E(\mathbb{F}_p)$, so it is clear that the n must be in the range $0 \leq n < \#\langle P \rangle$, where $\#\langle P \rangle$ is the order of cyclic subgroup generated by P. The number n is the discrete logarithm of base P. It is proved that discrete logarithms, like discrete exponents, have similar characteristics to the classic logarithms of real numbers [10]. We can use familiar notation: $n = log_P(Q)$, $when\, Q = nP$. Significant property of discrete logarithms [10]:

$$log_P(h * j) \equiv log_P(h) + log_P(j) \ (mod \#\langle P \rangle)$$

There is no such polynomial time algorithm which computes discrete logarithms for all cases. It should be noted that no proofs are available which states the non existence of such algorithm. In practice, exponential time algorithms are available [1, 3, 10] which compute discrete logarithms for cyclic subgroup G of $E(\mathbb{F}_p)$. For instance, *Shank's Baby-Steps, Giant-Steps* algorithm computes discrete logarithms in $\sqrt{\#G}$ steps where each step is one group addition operation [3] . Another algorithm is the *Pollard* p-method which computes discrete logarithms in approximately $\frac{\sqrt{\pi n}}{2}$ steps where n is the order of subgroup G [3]. The mentioned algorithms are two of the best known methods of computing discrete logarithms for elliptic curve based crypto systems.

To ensure maximum security of the crypto system (to increase the time required to solve the discrete logarithm problem), elliptic curve and underlying finite prime field must be properly selected. For instance, if $\#E(\mathbb{F}_p) = p$ then there is an algorithm available which computes discrete logarithms in $O(\log p)$ time [11].

Despite the fact that virtually all public key cryptography solutions can be implemented by using the popular RSA crypto system, crypto systems based on the discrete logarithm problem have several advantages over the integer factorization problem based crypto systems like RSA [10]:

- Technical advantages. If there are two algorithms which provide similar functionality, but one is based on the elliptic curve crypto system, while the other on the RSA crypto system, the first case of solving discrete logarithms is *harder* as integer factorization in the second case. The benefit is obvious. Discrete logarithm-based crypto systems have significantly smaller key sizes compared to RSA [10].
- Potential patent issues related to the RSA algorithms [10].
- Mathematical backgrounds. Possibility to improve the existing RSA crypto system security by introducing additional data-protection algorithms based on the discrete logarithm problem.

4 Crypto Systems Based on Elliptic Curves

Algorithms of cryptosystems based on elliptic curves use tuples of parameters to define an underlying curve E. E = (p, a, b, P, n, h) where p is a prime number which determines the finite prime field, a and b are coefficients of the curve, P – point on the curve, generator of cyclic subgroup, n – order of subgroup, h – cofactor of subgroup [5, 12].

As mentioned, there are classes of elliptic curves which are considered unsafe for use in cryptography. For instance, there are *weak* elliptic curves which allow calculation of discrete logarithms in polynomial time [11, 12]. To prevent malicious use of elliptic curves in the implementations of crypto systems, the curve coefficients a, b and the subgroup generator P can be randomized by using hash functions. In such cases, the set of curve parameters is extended by adding probabilistic number s, which is used to generate parameters $a, b$and$P$. An elliptic curve with probabilistic parameter s is denoted as *verifiably random elliptic curve* [12]. The generation of verifiably random curves is standardized, there are different methods of selecting the most appropriate hash function to generate the curve.

Although such approach provides relative protection against maliciously selected elliptic curves, however, there are certain risks involved, such as security and reliability of hash function being used to generate the curve parameters.

General key generation in elliptic curve based crypto systems: *For an elliptic curve* $E = (p,a,b,P,n,h)$ *, there are key pairs* (d,Q)*, where d is the private key which is natural number from the interval* $[1,n-1]$ *and Q is the point of an elliptic curve which is the public key* $Q = (x_q, y_q) = dP$ [5].

4.1 ECDH Crypto System

ECDH (*Elliptic Curve Diffie-Hellman*) is the version of *Diffie-Hellman* key exchange algorithm for elliptic curves, which determines the method how two communication participants *A* and *B* can generate key pairs and exchange their public keys via insecure channels. The algorithm determines only the method how key pairs are generated, the user defines the relation between encryption keys and data to be encrypted [7, 12]. After the keys have been exchanged it is common to use symmetric encryption methods. In practice, *ECDH* crypto system can be successfully adapted to different data security solutions [7].

The *ECDH* algorithm: *Given an elliptic curve over finite prime field* $E(\mathbb{F}_p)$*. Communication participants A and B agree on a point* $Q \in E(\mathbb{F}_p)$ *, which is freely available in the communication medium. The participant A secretly chooses probabilistic positive integer* k_A*, calculates* $k_A Q$ *and sends it to the party B. The member B also selects probabilistic positive integer* k_B*, calculates* $k_B Q$ *and sends it to the participant A. The shared secret* $P = k_A k_B Q$*.*

The participant A calculates P, by multiplying the received point $k_B Q$ *with the secret private key* k_A*. The participant B respectively calculates P by multiplying the received point* $k_A Q$ *with the secret private key* k_B [12].

After the public key exchange, one of the most typical methods how the encrypted messages are synchronized is the version of *ElGamal* crypto system for elliptic curves [1].

As the set $E(\mathbb{F}_p)$ is finite there is a limited set of information which could be encrypted by using points of the curve [1]. Also, there are no practical methods available to deterministically generate points of the given curve [1].

The problem can be solved by creating relation between the *x*-coordinate of given curve point P $\in E(\mathbb{F}_p)$ and a field $\mathbb{F}_p$ element that may not be a point on the curve [1]. There are algorithms *Compress Point* and *Decompress Point* available which allow reduction of the amount of memory required for storing the curve points in the computer memory by two times. It is possible to clearly identify the curve point $P = (xy)$ by specifiying the x-coordinate of point P and parity bit b. $b \equiv y \,(mod\, 2)$ [1].

4.2 ECDSA Crypto System

ECDSA (Elliptic Curve Digital Signature Algorithm) is *DSA* (*Digital Signature Algorithm*) digital signature standard analogue for elliptic curves. Like *DSA,* the goal of *ECDSA* is to provide verifiable digital signatures for data messages. *ECDSA* standard was approved in 2000 as *NIST FIPS 186-2* [1].

The author signs the data message by using his private key. The digital signature is added to the content of the message and can be freely validated by using the message author's public key.

Private keys are generated very much alike to *ECDH* crypto system. If P and Q are two points on an elliptic curve, then the private key is the discrete logarithm m=$\log_P$Q [1].

Algorithm of adding digital signatures works as follows: *Participants A and B agree on a set of curve* $E(\mathbb{F}_p)$ *parameters* $E = (p, a, b, P, n, h)$. *Let us suppose that the participant A wants to sign the message* $z \in \mathbb{N}$ *and the participant B wants to validate the digital signature of the received message by using the participant's A public key.*

Computing the digital signature:

1 *The participant A selects arbitrary integer k from the interval [1; n-1] where n is the order of cyclic subgroup* $\langle P \rangle$ *of* $E(\mathbb{F}_p)$.
2 *The participant A computes the point* $Q = kP$.
3 *The participant A computes* $r \equiv x_Q \pmod n$, *where* x_Q *is the* $x-$*coordinate of* Q.
4 *If* $r = 0$, *A repeats the previous step.*
5 *The participant A computes* $s \equiv k^{-1}(z + r * k_A) \pmod n$, *where* k_A *is the private key of A.*
6 *If* $s = 0$, *choose different random* $k \in$ *[1; n-1] and repeat the algorithm.*
7 *The tuple (r, s) is the digital signature* [12].

Verification of digital signature:

1. *The participant B computes* $i \equiv s^{-1} * z \pmod n$.
2. *The participant B computes* $j \equiv s^{-1} * r \pmod n$.
3. *The participant B computes the point* $Q = iP + jK'_A$, *where* k'_A *is the public key of A.*
4. *The digital signature is valid if and only if* $r \equiv x_Q \pmod n$ [12].

The *ECDSA* crypto system standard is widely used in practice. Along with the *ECDH* crypto system, *ECDSA* is being used in various network cryptographic protocols such as *SSL* and *TLS*, smart card solutions, embedded systems etc.

5 Typical Implementation Issues

Implementation issues of elliptic curve based crypto systems can be divided into four abstract categories.

The first category includes technical errors with regard to hardware and software implementations in the form of lack of authentication, inadequate *RAM* and media protection, errors in algorithms, incorrect network infrastructure etc. For those security problems it is common that the sensitive information, including private encryption keys, can come at the disposal of third parties without attempts of breaking the fundamental security background of the crypto system but by accessing the information directly

through the hardware and software security holes. This is the most common implementation issue and is not related to the security of the fundamentals of crypto system. It is associated with insufficient software testing and computer system security audits.

The second type of implementation issues is associated with the selection of underlying elliptic curves and prime fields. As mentioned, there are classes of *weak* elliptic curves. For instance, it is possible to solve the discrete logarithm problem in polynomial time for certain class of curves where $\#E(\mathbb{F}_p) = p$ – the number of points on a curve is equal to the number of elements in a finite field [3, 11]. Also, it is important to select large enough subgroups of $E(\mathbb{F}_p)$ to avoid feasible calculation of discrete logarithms by methods such as *Pollard-p* method [1, 13]. To ensure maximum security of the crypto system it is advisable to use verifiably random elliptic curves and prime fields such that the order of group $\#E(\mathbb{F}_p)$ is divisible by a sufficiently large prime number n where $n > 2^{160}$ [3].

The third type of implementations issues is associated with the performance of $E(\mathbb{F}_p)$ group operations like addition and scalar multiplication. It is advisable to use *Mersenne* primes, which can significantly improve the performance of the scalar multiplication operation [3]. This result is related to the processor architecture that enables an effective execution of module arithmetic operations with binary representation of the number, which is close to power of two [14]. Also, it is important to select the most appropriate coordinate system to improve the performance of group operations [5]. Depending on the selected coordinate system, the performance of group operations may vary. For instance, the performance of scalar multiplication can be improved by using *Jacobi* coordinate system in cases where the scalar multiplication takes place with even number (point doubling) [14]. It is possible to apply the *Jacobi* coordinate system by using the following connection to affine coordinates: *Jacobi coordinates represent an affine point* $(X/Z^2, Y/Z^3)$ *on elliptic curve* $y^2 = x^3 + ax + b$ *as a point (X:Y:Z), where* $Y^2 = X^3 + aXZ^4 + bZ^6, Z \neq 0$ [15].

The fourth type of implementations problems is associated with the private key management. It is required to ensure that the private keys are being re-calculated and re-issued on regular basis. Usage of constant private keys seriously increases the risk of keys being intercepted by a third party. The most typical example is the interception of a private key from 2010 with *Sony PlayStation* application signature crypto system where a constant private key was used for all issued digital signatures [13].

6 Security of Elliptic Curve Based Crypto Systems Versus RSA

The fundamental security of elliptic curve crypto systems is based on the algorithmically hard discrete logarithm problem. Elliptic curve cryptography is one of the most important practical applications of discrete logarithms nowadays [10].

According to the literature, *MIPS* (*million instructions per second*) capable computer can execute 4×10^4 $E(\mathbb{F}_p)$ group addition operations per second which is approximately 2^{40} additions per year [3]. It is clear, that this assumption is hypothetical and in practice may vary due to many factors related to the computer architecture, software and elliptic curve parameters.

Koblitz, Menezes and Vanstone have published the assessment of the required computing time to solve discrete logarithms by *Pollard p-method* in cyclic subgroups of $E(\mathbb{F}_p)$ with various orders *n*. The results are summarized in the table below.

Table 1. The Assessment of the Required Computing Time to Solve Discrete Logarithms by *Pollard p-method* [3]

Size of n (bits)	**MIPS (years)**
512	3×10^4
768	2×10^8
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

Pollard p-method can be parallelized. Thus, by using multi-processor computer systems, it is possible to reduce the time required to solve discrete logarithms. Theoretical assessment: *if 10 000 computers capable of 1,000 MIPS are available and n is 2160, then the calculation of single discrete logarithm takes approximately 85,000 years* [3].

There are very specific cryptographic processor architectures available, which provide the hardware level support of execution of parallelized *Pollard p-method* [5]. However, in practice modern general computers have built-in very capable graphic cards which are easily accessible and very strong cryptographic problem solving devices.

It is obvious that the computation of a single discrete logarithm could lead to the leakage of a single private key. Easy to conclude that in general case the illegitimate private key computation of elliptic curve based crypto systems is extremely expensive operation.

RSA (*Rivest, Shamir, Adleman*) is the most widely used public-key crypto system [4]. Since its introduction in 1977 [2] it is used around the world on wide range of security system solutions scaling from private users to global corporations.

Opposite to elliptic curve crypto systems, which are fundamentally based on the algorithmically hard discrete logarithm problem, RSA fundamental security is based on the algorithmically hard integer factorization problem which could be defined as follows: *given the product n of two primes q, p, find p and q such that* $n = qp$ [1].

The mathematical background of RSA is based on elementary modular arithmetic, which is relatively long known and well studied.

Currently, one of the most effective methods of solving the integer factorization problem is *General Number Field Sieve* algorithm [1]. This algorithm works in sub-exponential time [16]. As mentioned, the best known algorithms for solving the discrete logarithm problem are capable of exponential running time. For comparisons, *Koblitz, Menezes and Vanstone* have also published the assessment of the required computing time to solve the integer factorization problem by *General Number Field Sieve* algorithm [3], the results are summarized in the table bellow.

Table 2. The Assessment of the Required Computing Time to Solve the Integer Factorization Problem by *General Number Field Sieve* Algorithm [3]

Size of n (bits)	MIPS (years)
512	3×10^4
768	2×10^8
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

It is obvious that in general case the discrete logarithm problem is algorithmically *harder* than the integer factorization problem. Easy to notice, compared to elliptic curve based crypto systems, RSA keys can be obtained in relatively shorter time. In order to maintain the reasonable level of security, RSA keys must have longer bit length compared to elliptic curve based crypto systems.

6.1 Comparison of Key Generation

Taking into account the relatively high computing resources required to compute discrete logarithms, elliptic curve crypto systems allow to significantly reduce size of the encryption keys. The small key size enables faster execution of various cryptographic operations. According to the literature, it is concluded that RSA key generation takes place substantially slower than elliptic curve based crypto systems of comparable level of security [4]. The results are listed on the table below (Please see the publication for details of the experiment):

Table 3. Comparison of Key Generation [4]

Key Size (bits)		Generation time (seconds)	
ECC	**RSA**	**ECC**	**RSA**
163	1024	0.08	0.16
233	2240	0.18	7.47
283	3072	0.27	9.89
409	7680	0.64	133.90
571	15360	1.44	679.06

Easy to notice, the key generation of elliptic curve based crypto systems is significantly faster than RSA due to smaller key size. In addition, increasing level of security significantly increases the generation time ratio.

According to the literature, to ensure sufficient protection against elliptic curve crypto system key cracking, it is required to use keys with length of at least 150 bits for temporary security solutions and 180 bits for long term security solutions [3]. To meet the equivalent level of security, RSA keys must be at length of 1024 bits for short

term solutions and 2240 bits for long term solutions. Such RSA keys are not only 6 to 9 times longer, but also their generation is 2 to 40 times slower.

There is a study available that compares elliptic curve based crypto systems and RSA on implementations for 8-bit processor architectures. The authors experimentally observed that there is a fundamental relationship between the processor word length and the key length of crypto system: *The relative performance of ECC over RSA increases as the word size of the processor decreases* [14].

7 Summary

Despite the several decades long history of the elliptic curve cryptography, there is still a lack of research. The popular *RSA* crypto system is more widely studied. A significant lack of research is one of the main reasons why elliptic curve based crypto systems have showed low popularity nowadays. It is possible to conclude that the lack of research is related to the relatively complex mathematical foundation of elliptic curves and lack of interest from the systems developers.

It is expected that elliptic curves will play a growing role in various implementations. As mentioned, the discrete logarithm problem is algorithmically *harder* than the integer factorization problem, allowing a significant reduction in the public key cryptographic key size, thus speeding up a variety of cryptographic operations. Elliptic curve based crypto systems can be effectively used on low resources and power system solutions such as smart cards, mobile devices, sensors and so on.

The vast majority of implementation issues of elliptic curve based crypto systems are not directly related to the fundamental security backgrounds. These issues are related to the factors such as faulty software, inappropriate system components, inadequate private key protection, usage of defective random number generators and cryptographic hash functions etc.

Implementation options:

- The most used crypto systems such as *ECDH* and *ECDSA* are standardized and patent free. They are free to use.
- There are available *NIST* standardized elliptic curves for various security requirements.
- Free access to the extensive information on algorithms for elliptic curves based crypto systems.

Benefits of elliptic curve based crypto systems versus RSA crypto system:

- Key size. The key of an elliptic curve based crypto system takes significantly less memory. The ratio increases rapidly with the increase of security levels. For instance, RSA crypto system with the key length of 1024 bits, is equivalent to an elliptic curve crypto system with the key length of 163 bits.
- Cryptographic operations performance. Thanks to the smaller size of keys, the cryptographic operations such as key and digital signature generation are carried out significantly faster. For instance, an elliptic curve crypto system with the key length of 233 bits corresponds to RSA crypto system with the key length of 2240 bits. In the first case the key is generated approximately 40 times faster.

- Resource savings. Due to the smaller key sizes, algorithms of an elliptic curve based crypto systems can be executed on very limited resources.

Disadvantages of elliptic curve based crypto systems versus RSA crypto system:

- Significantly more complex mathematical backgrounds.
- Relatively large group of *weak* elliptic curves.
- Lack of research.

References

1. Stinson, D.R.: Cryptography Theory And Practice. 3th edition, Chapman & Hall/CRC, New York (2006)
2. Maurer, U.M., Wolf, S.: The Diffie–Hellman Protocol. In: "Towards a Quarter-Century of Public Key Cryptography", Kluwer Academic Publishers, pp. 147–171, Boston (2000)
3. Koblitz, N., Menezes, A., Vanstone, S.: The State of Elliptic Curve Cryptography. In: "Towards a Quarter-Century of Public Key Cryptography", Kluwer Academic Publishers, pp. 173–193, Boston (2000)
4. Arrendondo, B., Jansma, N: Performance Comparison of Elliptic Curve and RSA Digital Signatures. IPCSIT vol. 4, IACSIT Press, Singapore (2011)
5. Brown, D.R.L.: SEC 1: Elliptic Curve Cryptography. Certicom Corp (2009)
6. Novotney, P.: Weak Curves In Elliptic Curve Cryptography (2010) `http://ftp.mpir.org/edu/2010/414/projects/novotney.pdf`
7. Schoof, R.: Elliptic Curves Over Finite Fields and the Computation of Square Roots. Mathematics of Computation vol. 44, pp. 483–494 (1985)
8. Corbellini, A: Elliptic Curve Cryptography: Elliptic Curve Cryptography: finite fields and discrete logarithms (2015) `http://andrea.corbellini.name/2015/05/23/elliptic-curve-cryptography-finite-fields-and-discrete-logarithms`
9. Robinson, J.S.D.: An Introduction to Abstract Algebra. Walter de Gruyer GmbH & Co (2003)
10. Odlyzko, A.: Discrete logarithms: The past and the future. In: "Towards a Quarter-Century of Public Key Cryptography", Kluwer Academic Publishers, pp. 129–145, Boston (2000)
11. Silverman, H.S.: An Introduction to the Theory of Elliptic Curves. University of Wyoming (2006)
12. Corbellini, A.: Elliptic Curve Cryptography: ECDH and ECDSA (2015) `http://andrea.corbellini.corbellini.name/2015/05/30/elliptic-curve-cryptography-ecdh-and-ecdsa/`
13. Bos, J.W., Halderman, J.A., Heninger, N., Moore, J., Naehrig, M., Wustrow, E.: Elliptic Curve Cryptography in Practice (2014) `https://eprint.iacr.org/2013/734.pdf`
14. Gura, N., Patel, A., Wander, A., Eberle, H., Shantz, S.C.: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. LNCS, vol. 3156, pp. 119–132. Springer, Heidelberg (2004)
15. Brown, M., Hankerson, D., Lopez, J., Menezes, A.: Software Implementation of the NIST Elliptic Curves Over Prime Fields LNCS, vol. 2020, pp. 250–265. Springer, Heidelberg (2001)
16. Schaefer, E.: An introduction to cryptography and cryptanalysis (2011) `http://math.scu.edu/\textasciitildeeschaefe/book.pdf`