Results of an Elliptic-Curve-Approach for Use in Cryptosystems

Reinhard Rauscher, Frank Bohnsack
University of Hamburg
Dept. of Computer Science
Hamburg, Germany
e-mail: rauscher@informatik.uni-hamburg.de

Abstract

This paper presents new research results concerning cryptosystems based upon elliptic curves. The system which has been developed is similar to that of ElGamal *[4] over a finite field F_q [7]. The investigations cover those elliptic curves with primary characteristics over F_q ($2^{16} \leq q \leq 2^{257}$, q is a prime number). The aim of the study is the determination of dependencies between calculation time required by the cryptosystem and size of the keys and length of messages. Additionally, possible areas vulnerable to attacks are identified.*

1. Introduction

As the importance of digital communication in commerce increases in the sensitive 'areas' like home banking, internet, exchange of patients' data, the security requirements increase rapidly. Apart from these high-security applications, low-security applications are also arising. These require compromises between security and system response time (in standard low-security applications real-time answers are less than 3 seconds). Examples for these applications are telephone cards and personal access cards. Though the current cryptosystems are up to the required security standards, computational speed is often seriously compromised due to increased key size, in contrast to the increasing speed of data transmission. In other words: the higher the required security, the slower the encryption.

Therefore, any research into different mathematical structures is an important part of cryptography. This article concerns the research into an alternative: cryptosystems based on elliptic curves. The aim is to explore the run time of the systems and the impact on the key size and the message transferred.

This is important in order to avoid security holes of the system.

2. State of Art

The established cryptosystems can be distinguished into two classes:

- symmetric cryptosystems (a well-known example is the DES-algorithm [5]) which are not discussed here and
- asymmetric cryptosystems (for example, the well-known *RSA*-algorithm).

Every user of an asymmetric cryptosystem needs two keys: a public key d_{public} and a private key d_{secret}. Everybody may know the public key of the users, but not the private key which is kept secret. It is important that nobody can calculate the private key d_{secret} through knowledge of the public key d_{public} (public-key-attribute). Therefore, only a restricted subset of mathematical functions is suitable as encryption or decryption functions. The safety of a cryptosystem against attacks depends essentially on the difficulty of calculating the inverse function [1].

Compared to the often used *RSA* system, the elliptic curves are a serious alternative because they offer the same security against attacks with a smaller key size [17], [20], [2].

Definition of an elliptic curve:
Let F be a finite field of characteristic unequal 2 and 3. Let $x^3 + ax + b$ $(a, b \in F)$ be a cubic polynomial. Then, the set of points (x, y) with $x, y \in F$ that satisfy the equation

$$y^2 = x^3 + ax + b, \tag{1}$$

together with an additional point O, called the point at infinity, form an elliptic curve E over F ([8], [3] and [16]).

The elliptic curve E forms an abelian group. It is relatively easy to determine the point $Q = k \cdot P$ by scalar multiplication at a given point P on the elliptic curve E.

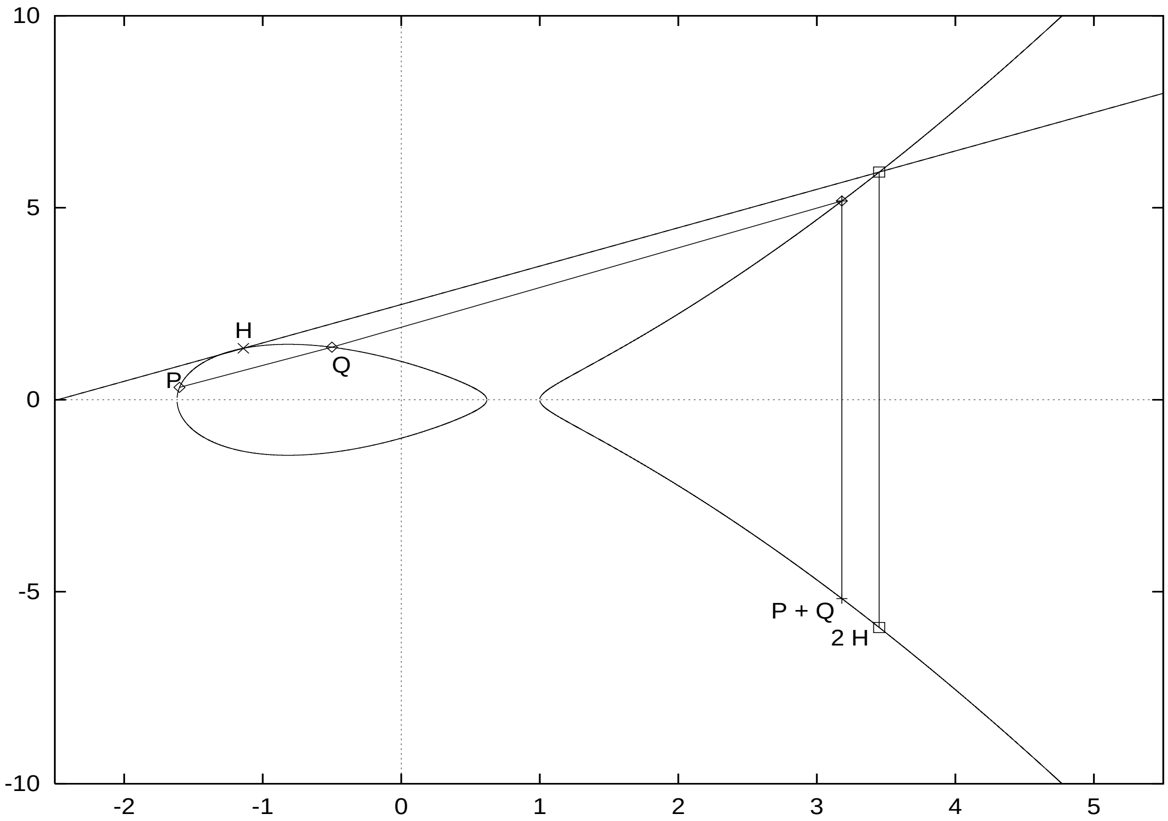

Figure 1. Addition of an elliptic curve: *Addition of two different points P and Q to $P + Q$ and doubling the point H to $2H$*

Determing the scalar k of two given points $P, Q \in E$, if it exists, is much more difficult [6].

It is assumed [6] that the solution to the so called *discrete log problem* over E is much more complex than the discrete log problem over finite fields F_q. This guarantees a smaller key size for elliptic curve cryptosystems while preserving the required safety standard.

Definition of the discrete log problem over E
Let E be an elliptic curve and $B \in E$ a point on the curve. Then to find $x \in Z$ of a given point P if such an x exists that satisfies the equation $x \cdot B = P$ is called discrete log problem over E.

At present, a 512 bit key size appears to be secure for *RSA* [15]. However, doubling the key size is now being considered and has already been partially realized [15]. In the application of elliptic curve cryptosystems the increase in key size is remarkably small in comparison to the standard *RSA* based systems [12]. Thus the increase in execution time and amount of RAM required are also comparitively minor.

Unfortunately, the scalar multiplication over elliptic curves consists of multiple additions of elliptic curve points. A geometrical demonstration of an addition of arbitrary elliptic curve points P and Q and of doubling an elliptic curve point H over the field of real numbers is shown in Figure 1):

The addition procedure is formally defined as follows:

- $O + O = O$ (the point at infinity is the neutral element of the addition)
- $(x_1, y_1) + O = (x_1, y_1)$
- $(x_1, y_1) + (x_1, -y_1) = O$
- $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ with
  1. if: $x_1 = x_2$ and $y_1 = y_2$
  $$x_3 = (\frac{3x_1^2+a}{2y_1})^2 - 2x_1$$
  $$y_3 = (\frac{3x_1^2+a}{2y_1})(x_1 - x_3) - y_1$$
  2. if: $x_1 \neq x_2$
  $$x_3 = (\frac{y_2-y_1}{x_2-x_1})^2 - x_1 - x_2$$

$$y_3 = (\frac{y_2 - y_1}{x_2 - x_1})(x_1 - x_3) - y_1$$

Doubling a point (*first if*) comprises 6 additions (subtraction included), 4 multiplications and one division in F_q. An addition of two different elliptic curve points (*second if*) consists of 6 subtractions, 2 multiplications and one division in F_q. A division is considerably more complicated than an addition or multiplication in a finite field F_q. For this reason, it is important to analyse the run time of the cryptosystem in order to discover whether elliptic curves hold the advantages mentioned above or not.

3. Implementation

The algorithm which has been chosen is the *ElGamal* algorithm, based on elliptic curves over F_q, as is demonstrated in [6]. It was programmed using C++ operating under a LINUX system on a PC platform. The source code consists of approx. 1500 lines of code, the compiled code consists of approx. 84 Kbytes. Therefore, it could be implemented within devices with a small RAM.

The overall procedure based on the *ElGamal* procedure is illustrated in Figure 2.

First, the sender and receiver start with their secret keys and generate the corresponding public keys. This phase is named 'Key construction' in Figure 2. It should be mentioned, that this task is performed within trust centers. For this task there is no need for realtime.

Within the second phase, named 'Encryption' the sender searches for the public key of the receiver, randomly chooses a number k, encodes the message and sends the ciphertext to the receiver. In the last step, the receiver decodes the message.

Referring to Figure 2, the question arises as to which system parameters of the cryptosystem are public information. Public parameters are:

- prime q of the finite field F_q
- elliptic curve E over F_q
- base point $B = (x_B, y_B) \in E$
- order of the base point $\#B = n$ (determination described in [10],[11])
- person specific keys:
  1. public key of each user $d_p = (x_p, y_p) \in E$
  2. private key of each user $d_s \in F_q$ *(kept secret)*

The following local variables are necessary:

(x_2, y_2) : point on the elliptic curve in order to encrypt the message
$k \in F_q$: randomly chosen scalar of the encryption
$d_{p_e} \in E$: public key of the receiver
(x_1, y_1) : calculated point kB transmitted with the encrypted data

The following equation is fundamental to the functioning of the cryptosystem:

$$(x_2, y_2) = kd_{p_e} = kd_{s_e}B = d_{s_e}kB = d_{s_e}(x_1, y_1)$$

The x_2-coordinate of the point (x_2, y_2) is used to encrypt the message. All succeeding decryption steps are based on finite field and bit operations.

Within the work described, a simulation program has been developed. The simulation program is divided into three blocks (see Figure 3). In the first block the system parameters of the cryptosystem are established: the elliptic curve E and thus its characteristic $\#E = prime$ (to be chosen from 16 elliptic curves, E and its characteristic is not calculated in the simulation program), the x-coordinate of the base point B and thus its order $\#B = n$, and finally the necessary public d_{p_e}, d_{p_s} and private d_{s_e}, d_{s_s} keys. If the system has z users, then the system consists of many public $d_{p_1}, \ldots, d_{p_z}$ and private $d_{s_1}, \ldots, d_{s_z}$ keys.

The second block is the encryption. The fundamental part of this is choosing the scalar k (see above) and encoding the message, followed by specific finite field and bit operations (see [6]). The third block is the decryption of the message.

The execution time is measured in all three blocks, the results of which are documented in chapter 4.

4. Results

Using 16 elliptic curves, we investigated the characteristics of the cryptosystem.

As expected, the division operation is the most time consuming operation in comparison to other operations within F_q. In the simulation program, the division is calculated by its inverse, using the extended euklidic algorithm. The time complexity of the algorithm is $O(\ln^2 q)$ [9]. In order to reduce the number of necessary additions to perform the scalar multiplication an algorithm similar to the square-and-multiply algorithm is used [9].

To estimate the CPU-time needed for encryption/decryption and evaluation of additional dependencies we performed the following experiments:

1. Execution time of the simulation program depending on the choice of q and E

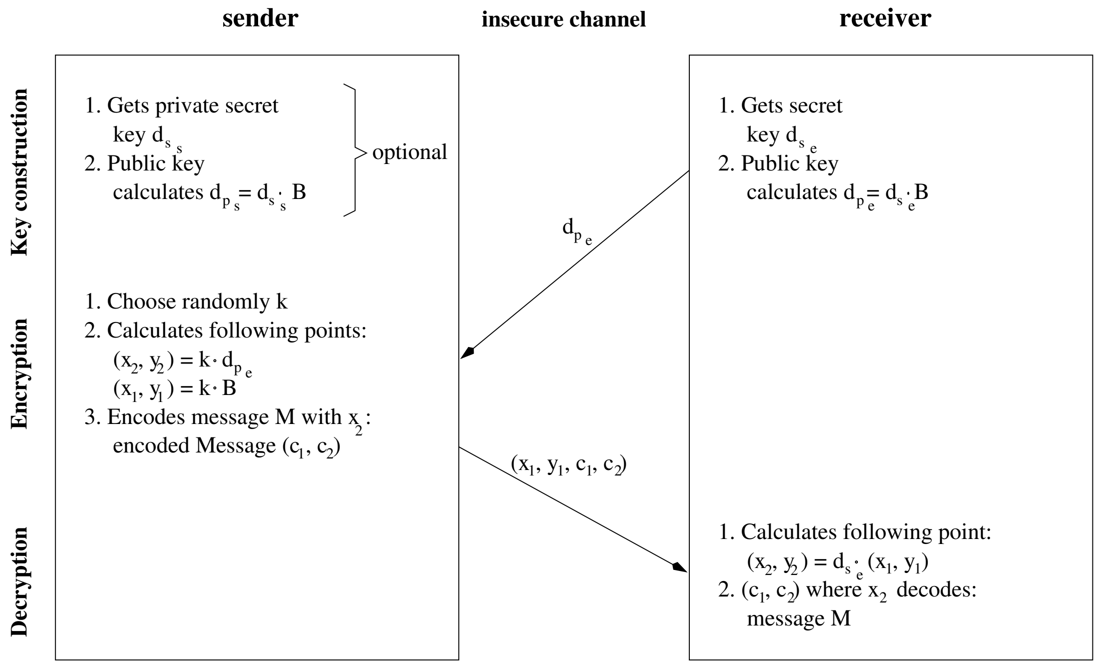

Figure 2. Run of the *ElGamal* cryptosystem based on Elliptic Curves

2. Dependency of the execution time on keys and messages

All the analyses were carried out using a LINUX operating system on a PC with a Pentium 133 MHz processor. The run times described cover only CPU calculation time.

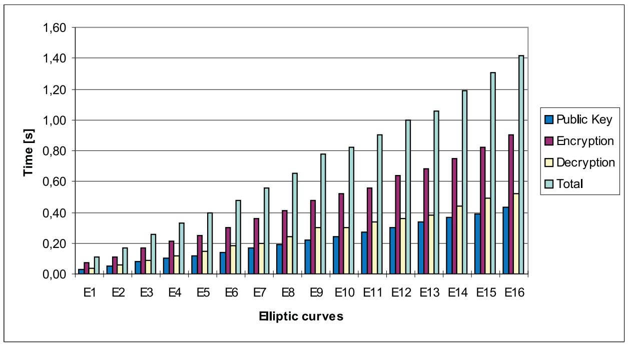

Figure 4. Different Elliptic curves, *using the same keys and the constant message M of 1 byte where q is a prime and is greater than* 10^5, 10^{10}, 10^{15}, 10^{19}, 10^{25}, 10^{30}, 10^{35}, 10^{40}, 10^{45}, 10^{50}, 10^{55}, 10^{60}, 10^{65}, 10^{70}, 10^{75}, 10^{80}

In the first experiment, randomly chosen numbers representing messages were encrypted and decrypted using ran-

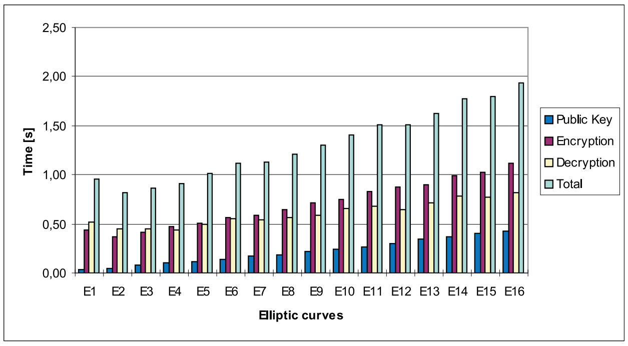

Figure 5. Different Elliptic curves, *using the same keys and the constant message M of 235 byte where q is a prime and is greater than* 10^5, 10^{10}, 10^{15}, 10^{19}, 10^{25}, 10^{30}, 10^{35}, 10^{40}, 10^{45}, 10^{50}, 10^{55}, 10^{60}, 10^{65}, 10^{70}, 10^{75}, 10^{80}

domly chosen keys. The 16 available elliptic curves were generated by the program [13], all of them have prime characteristics. Table 1 shows examples of three elliptic curves E_1, E_8 and E_{16}.

Encryption and decryption were carried out several times using these curves in order to obtain the average run time. The run time depends on the prime q of bits and can be

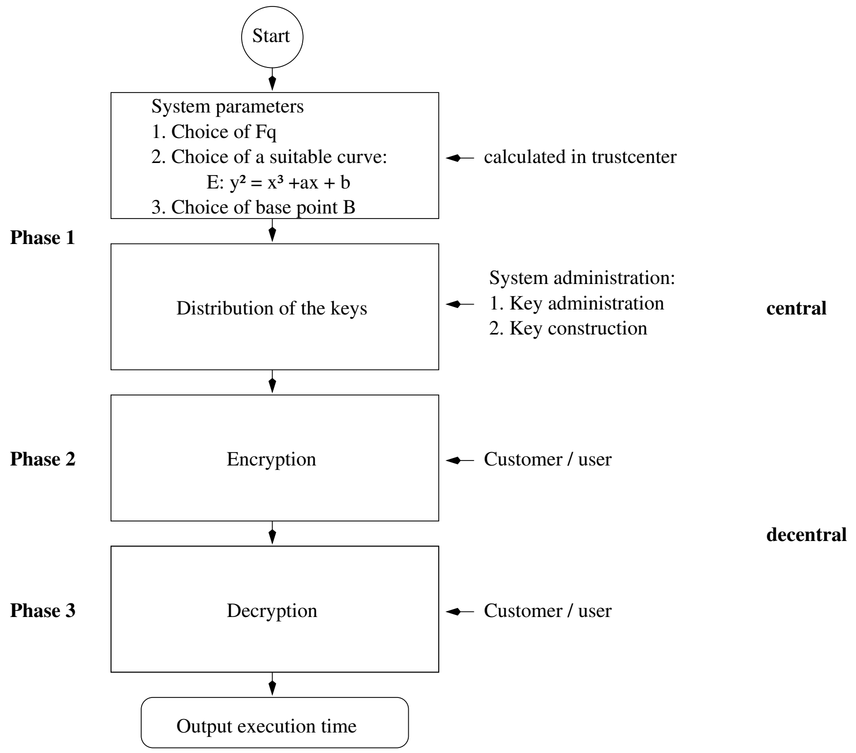

Figure 3. Run of the simulation program

equated to a parabolic curve: a linear increase in q results in a quadratic increase in execution time.

The first results are shown in Figure 4. The message to be transferred has been restricted to one byte, the keys k and d_{s_e} were kept constant for all elliptic curves; values for k and d_{s_e} have been 23451. As expected, the time for calculation of d_{p_e}, encryption, and decryption increases with the increasing prime q.

Obviously, a message to be encrypted in one step must not be larger than the minimum of $\#E$ and q. Should longer messages be transferred, the message has to be divided into small parts. For comparison we limited the length of the message to 235 bytes. Figure 5 shows the results for the case where a longer message (which has to be divided into small parts for all 16 elliptic curves) is to be transferred.

Once more, the keys k and d_{s_e} were kept constant for all the 16 elliptic curves. For example, the value of k has been chosen to be 23451 for all elliptic curves, the value for d_{s_e} has been restricted (it is dependent on the underlying elliptic curve) to be nearly the same as k, namely 23450.

The first item in Figure 5 indicates the time needed for calculation of the public key d_{p_e}.

The second and the third items describe the time needed for encryption and decryption, respectively. It should be noted that the relatively small increase of the calculation time is, on one hand, based on the fact that fewer cycles have to be performed in case of huge elliptic curves. Additionally, only small keys are used. For the case where large keys are used, the calculation time would rise. It should be noted that in all phases the value of k has been constant and therefore the value of $k \cdot B$ is determined only once.

For avoiding attacks, the changing of k within one message would be useful. In this case, additional time (as described for d_{p_e}) would be necessary.

Due to the checking procedure, necessary to avoid division by zero in the encryption, using $k \cdot d_{p_e} = (x_2, y_2)$ the calculation of the sender's dynamic public key $k \cdot B = (x_1, y_1)$ takes twice as long as that of the receiver's.

Additionally, it was discovered that the total execution time of the system is practically constant (at most a factor of 2) when applying different elliptic curves. The reason for this is that each elliptic curve has a similar complexity when calculating the $k \cdot d_{p_e} = (x_2, y_2)$ required for encryption. All additional operations of the cryptosystem consist of simple logic operations (see [6]) like string concatenation or bitwise XOR.

The run time relationship between the size of the key and the time for encryption is important to avoid successful attacks. On the other hand there may be applications, for which only encryption or only decryption is necessary in realtime.

The relationships are shown in Figures 6 and 7. Two different procedures using elliptic curves over F_q with $q \geq 2^{15}$, 2^{127} and 2^{256} are analysed maintaining all parameters

E_1		
q	:	33013
a	:	2155
b	:	16008
$\#E$	:	33247
E_8		
q	:	170141183460469231731687303715884105757
a	:	105373295517709951970325062096778779566
b	:	34674159133580410663812342760227529126
$\#E$	:	170141183460469231746769722985284162023
E_{16}		
q	:	115792089237316195423570985008687907853269984665640564039457584007913129640233
a	:	95270011154113989137874045642563197904974382476028229496939583044928154917770
b	:	67197498288843822956561798249438724627559132613538319448066573513035828 2482
$\#E$	:	115792089237316195423570985008687907852926634160420486457496696723664680664331

Table 1. Elliptic curves. *Three analysed elliptic curves E_1, E_8 and E_{16}*

mentioned above:

1. fixed receiver's private key d_{s_e} and variable dynamic sender's privat key k (Figure 6)
2. fixed sender's private key k and variable receiver's privat key d_{s_e} (Figure 7)

Obviously, the calculation time of the receiver's public key d_{p_e} and of d_{p_s} is constant and not shown in Figure 6. In comparison to encrytion/decryption time they can be neglected. Additionally, the calculation time for determining d_{p_e} or $k \cdot B$ is constant due to the fact that either k or d_{s_e} are constant (value is 23451). These values are 0,03 sec. for E_1, 0,18 sec. for E_8, and 0,43 sec. for E_{16}.

Let's start discussing Figure 6: choice of the private key has logarithmic time complexity. The reason for this is that the above mentioned scalar multiplication is similar to the square-and-multiply algorithm. In addition, increasing the key size extends the calculation time of the encryption. This is caused by the calculation of $(x_1, y_1) = k \cdot B$ and $(x_2, y_2) = k \cdot d_{p_e}$. If k increases, the calculation time extends approximately logarithmically for (x_2, y_2). In contrast to the encryption, the time for the decryption is nearly constant due to the receiver's constant private key. The variations result from the different numbers of modulo reductions in F_q while calculating the point $(x_2, y_2) = d_{s_e} \cdot (x_1, y_1)$. Thus, the times vary around an average execution time which depends on the numbers of reductions.

In Figure 7 the results of varying the receiver's private key are shown. The calculation of the public key d_{p_s} is constant due to $d_{p_s} = d_{s_s} \cdot B$.

Due to the checking procedure ($d_{s_s} \cdot d_{p_e} = (x_2, y_2)$), necessary to avoid division by zero in the decryption the calculation of d_{p_s} takes approximately twice as long as the calculation of d_{p_s} in the first case (see Figure 6).

In comparison, the calculation of the receiver's public key d_{p_e} now takes only half the time as it does in the first case. The encryption time depends only on d_{s_s} which is constant. Compared to the encryption, the decryption takes longer the larger the key size of the receiver's private key d_{s_e} is. The main reason for this is based on the calculation of the point $(x_2, y_2) = d_{s_e} \cdot (x_1, y_1)$.

	Message 1	Message 2	Message 3
Key d_{p_e}	$0,71$ s	$0,73$ s	$0,73$ s
$k \cdot B$	$0,73$ s	$0,73$ s	$0,73$ s
Encryption	$1,51$ s	$1,51$ s	$1,53$ s
Decryption	$0,81$ s	$0,81$ s	$0,81$ s
Total	$3,05$ s	$3,05$ s	$3,07$ s

Table 2. Execution time depending on the message M over E_{16} with $q > 2^{256}$

In contrast to the encryption in the first case (Figure 6), the decryption in the second case (Figure 7) takes longer, due to the intensive calculation of the inverse over the finite field F_q.

It can concluded that, in order to reduce the execution time of the cryptosystem, it is more convenient to choose a large private key d_{s_s} and the smallest possible multiplier of the base point B for the public key. This might be a disadvantage due to the possible compromising of cryptosystem security.

Many cryptosystems have the disadvantage that attackers are able to deduce the message through the analysis of the time used [19].

Therefore, the important question arises as to whether the developed cryptosystem demonstrates the same characteristics. The implemented cryptosystem requires only an XOR operations and a finite field multiplication which are dependent on the message. Compared to the scalar multi-

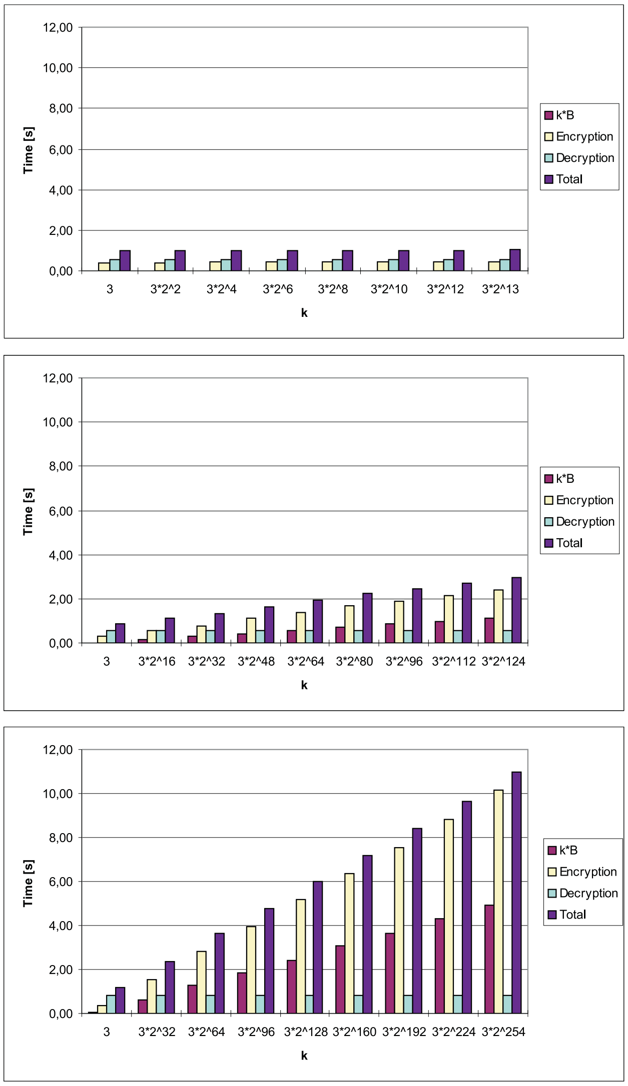

Figure 6. Execution Time, *varying the sender's private dynamic key k over E_1 with $q > 2^{15}$ (top), E_8 with $q > 2^{127}$ (middle) and E_{16} with $q > 2^{256}$ (bottom). Value of d_{s_e}: 23451.*

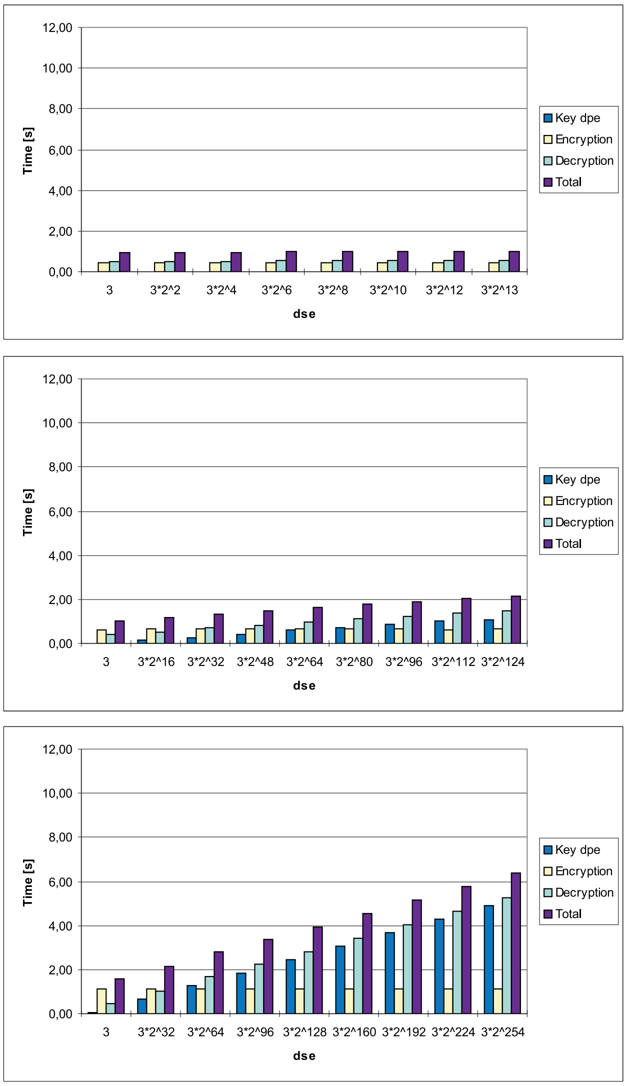

Figure 7. Execution time, *varying the receiver's private key d_{s_e} over E_1 with $p > 2^{15}$ (top), E_8 with $p > 2^{127}$ (middle) and E_{16} with $p > 2^{256}$ (bottom). Value of k: 23451.*

plication of elliptic curves, these two operations can be neglected. In Table 2 the elliptic curve E_{16} with $q > 2^{256}$ is analysed with the following messages of maximum length:

- Message 1: $2^{256} = 100000\ldots_2$
- Message 2: $2^{256} - 1 = 111111\ldots_2$
- Message 3: alternate $101010101\ldots_2$

The message which is the most influential on the encryption execution time is the bitwise XOR operation resulting in $1111\ldots_2$. This means that the final number of multiplications are at a maximum. As can be seen in Table 2, the time is nearly independent of the structure of the message which will be encoded. The differences shown in Table 2 are less than 1%. A similar result can be found during the decryption process.

So, the execution time of an *ElGamal* cryptosystem based on elliptic curves over a finite field F_q is dependent mainly on two factors:

- the choice of the finite field F_q
- the choice of the keys (k and d_{s_e})

The greater the prime q, the more resources are used during the finite field operations. The same applies to the choice of the private keys in the encryption and decryption. A large private key will increase the number of additions of the elliptic curves.

Additional results can be found in [18].

5. Further Activities

The aim of further analysis is to accelerate the system. As the results show, the existing system - as implemented - is only suitable for low-security applications. A possible improvement is to reduce the number of divisions. This could be achieved by choosing a nominator and denominator representation. After the seperate calculation of the nominator and denominator the final division could be performed. In addition to this, using a dedicated hardware structure based on large numbers or utilizing that parts of the system can be parallelized system speed could be enlarged. Analyses over the finite field F_{2^k} also have to be investigated.

6. Acknowledgements

The authors thank Dr. Volker Müller from the university of Darmstadt for supplying the analysed elliptic curves. Without them an analysis would have been very difficult. Also for his interesting discussions and explanations that were most useful in realising some problems, and recognizing some new starting points. Special thanks to Philips Semiconductors, Hamburg, especially to Dr. Steffen Drews and Helmut Hörner, for their practical suggestions, discussions and support. Further thanks to Mr. Manfred Grove for giving us support concerning mathematical problems.

References

[1] T. Beth, *Kryptographie: Eine Einführung in die Methoden und Verfahren der geheimen Nachrichtenübermittlung*, Teubner, Stuttgart 1983

[2] A. Beutelspacher, J. Schwenk, K.-D. Wolfenstetter *Moderne Verfahren der Kryptographie*, Vieweg Verlag, Braunschweig, Wiesbaden 1995

[3] H. Cohen, *A Course in Computational Algebraic Number Theory*, Graduate Texts in Mathematics, Springer-Verlag, Berlin, Heidelberg 1993

[4] T. Elgamal, *A public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms*, IEEE Trans on Information Theory, Vol. IT-31 1985

[5] P. Horster, *Kryptologie*, Bibliographisches Institut Mannheim Wien Zürich, Zürich 1985

[6] A. Menezes, M. Qu, S. A. Vanstone, *IEEE P1363 Standard Part 6 Elliptic Curve Systems*, Ontario, 1995

[7] D. Jungnickel, *Finite Fields*, Bibliographisches Institut & F.A. Brockhaus AG, Mannheim 1993

[8] N. Koblitz, *A Course in Number Theory and Cryptography*, Graduate Texts in Mathematics, Springer-Verlag, New York 1987

[9] D. E. Knuth, *The Art of Computer Programming - Seminumerical Algorithms*, Addison Wesley, 1981

[10] R. Lercier, F. Morain, *Counting the Number of Points on Elliptic Curves over Finite Fields: Strategies and Performances*, *Advances in Cryptology - EUROCRYPT '95*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg 1995, S. 79 - 94

[11] F. Morain, *Building Cyclic Elliptic Curves Modulo Large Primes*, *Advances in Cryptology - EUROCRYPT '91*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg 1991

[12] A. Menezes, T. Okamoto, S.A. Vanstone, *Reducing elliptic curves logarithms to logarithms in a finite field*, *Proceedings 23rd Annual ACM Symposium on Theory of Computing (STOC)*, ACM Press, S. 80 - 89

[13] V. Müller, *Ein Algorithmus zur Bestimmung der Punktanzahl elliptischer Kurven über endlichen Körpern der Charakteristik größer drei*, Dissertation, Saarbrücken 1995

[14] R. L. Rivest, A. Shamir, L. Adleman, *A Method for Obtaining Digital Signatures and Public key Cryptosystems*, Communications of the ACM, Vol. 21, No. 2, 1978, S. 120 - 126

[15] M. J. B. Robshaw, *Security Estimate for 512-bit RSA*, RSA Laboratories, 1995

[16] J. Silverman, *The Arithmetic of Elliptic Curves*, Graduate texts in Math. 106, Springer-Verlag, New York 1986

[17] R. Schroeppel, H. Orman, S. O'Malley, O. Spatscheck, *Fast Key Exchange with Elliptic Curve Systems*, *Advances in Cryptology - CRYPTO '95*, Lecture Notes in Computer Science, Springer-Verlag, Berlin, Heidelberg 1995, S. 43 - 56

[18] F. Bohnsack, *Untersuchung von Elliptische Kurven für die Tauglichkeit zur Hardwareakzeleration von Kryptoverfahren*, Diploma Thesis, Univ. of Hamburg, June 1997

[19] P. C. Kocher, *Timing Attacks on Implementations of Diffie-Hellmann, RSA, DSS, and Other Systems*, Cryptography Consultant

[20] S. Hamdy, *Elliptische Kurven - neue Wege in der Public-Key-Kryptographie* in Proc. 6. Workshop "Sicherheit in vernetzten Systemen", Hamburg, March 1999