Advances in Computational & Experimental Engineering & Science
Copyright 2004 Tech Science Press

Public Key Algorithm Comparison

M. de-Miguel-de-Santos [1],C.Sanchez-Avila [1], R.Sanchez-Reillo [2]

Summary

The main target of this work is to expose the capacities that make elliptic curve cryptography the most suitable one to be implemented in environments with several constrains related to processor speed, bandwidth, security and memory.

Introduction

In order to present the advantages of using elliptic curves in cryptography, we have compared several public key algorithm characteristics with elliptic curve ones. We have made a comparison among different public key cryptosystems: ElGamal for encryption, Diffie-Hellman for key exchanging and the corresponding ones based on elliptic curve theory, as well as the RSA algorithm. In order to achieve the purpose of this study, several performing features have been tested: memory usage in the execution process and speed. We have used M. Rosing subroutines implemented in C to obtain the different elliptic curve algorithm characteristics. The underlying fields chosen are Galois fields GF(2n), using polynomial basis as well as normal ones.

Diffie Hellman Protocol over Elliptic Curves

In order to obtain the same secret key, the two peers involved in the key agreement protocol, combine the other peer's public key with its private key. Both parties must share certain *domain parameters* [1] which are used in each cryptographic protocol over elliptic curves. The parameters that must be known by each party are the listed below:

Table 1: Domain Parameters

Domain Parameters	**Meaning**
$GF(\mathbb{F}_q)$	Finite Field
q	Prime number different than 2 or 2^m, where m is a positive integer
$a,b \in GF(\mathbb{F}_q)$	Curve parameters
r	Prime positive integer, that divides the curve order, $r \mid (\#E(\mathbb{F}_q))$
$G \in E(\mathbb{F}_q)$	Point which belongs to the curve, whose order is r

[1]E.T.S.I. Telecomunicacion -Dpt. Matematica Aplicada Ciudad Universitaria, s/n; E-28040 - Madrid; SPAIN e-mail: mariademiguel@wanadoo.es

[2]Universidad Carlos III de Madrid - Dpt. Tecnologia Electronica c/ Butarque, 15; E-28911 - Leganes (Madrid); SPAIN e-mail: rsreillo@ing.uc3m.es

Besides, when $q = 2^m$, the way to represent the elements in $GF(\mathbb{F}_q)$ (normal basis or polynomial ones), is another parameter . Another one is the cofactor $k = \#E/r$.

Apart from the domain parameters, each peer must own a pair of keys: the private key $s \in [1, r-1]$ and the public one W, which is a point belonging to the elliptic curve E, such that $W = sG$ where $G \in E$ whose order is r. The methods used to generate public keys as well as the ones to validate them are described in [1]. Below these lines, it is briefly described the basic tests that a public key $Q = (x_q, y_q)$, which belongs to the other peer, must fulfill in order to be considered a right one:

- Check that $Q \neq O$
- Verify that x_q e y_q are elements of the field $\mathbb{F}_q$ considered
- Verify that Q belongs to the elliptic curve E chosen
- Check that $nQ = O$

Sometimes, the last check is omitted, as the multiplication operation produces a high computational cost, so if a volatile public key is computed each time a certain protocol is carried out, then the validations procedure would cost a extra high load during execution time.

It was in 1985 when for the first time, the possibility to use the group of points of an elliptic curve in the Diffie-Hellman Key Agreement Protocol appeared [2]. ECKAS-DH1 is the Diffie-Hellman Key Agreement Protocol over elliptic curves, in which each party contributes with a unique pair of keys. In this protocol, either the ECSVDP-DH or the ECSVDP-DHC primitives may be chosen, as well as a key derivation function —SHA-1 or RIPEMD-160—. In order to carry out this protocol it is also necessary to choose the domain parameters, a private key according to those domain parameters, obtain the public key of the other party —and optionally, check that it is a valid key–, then obtain the secret shared value, and finally, execute either SHA-1 or RIPEMD-160 [1] over the secret shared value to obtain the secret shared key .

The IEEE proposed another version of this protocol to avoid several problems that the Diffie-Hellman protocol deals with. In this case, the two parties have two key pairs, which produces the generation of two secret shared values.

In general, the main disadvantage of the Diffie-Hellman protocol is that it may suffer the “man-in-the-middle” attack.

There is also a Menezes-Qu-Vanstone Key Agreement Protocol version over elliptic curves based on the primitives ECSVDP-MQV or ECSVDP-MQVC.

ElGamal Protocol over Elliptic Curves

Public key encryption algorithms are mainly used, not for data encryption, but for key exchange; this is due to the fact that public key encryption is much more slow than private one. The encryption protocol described in this article is ElGamal, which is widely spread, not just over elliptic curve, but also over integer number finite fields $\mathbb{F}_q$.

There are other public key encryption algorithms implemented over elliptic curves, such as: Massey-Omura (not included in [1]), which is not as spread as ElGamal, but has several advantages as its simplicity; and the ECIES —Elliptic Curve Integrated Encryption Scheme—, which is a secure protocol under different circumstances, for example, chosen adaptive encrypted text attacks.

ElGamal protocol may be used in data encryption as well as key exchange, due to its flexibility. It is not a secure protocol, when referring to active attacks; so the use of this protocol is just recommended with other extra security elements, as for example, the mixture of the encryption algorithm as well as a digital signature one and a hash function, as studied by Schnorr and Jackobsson. When using this protocol, it is not necessary to know the curve order. Another advantage is that it does not exist a patent over the protocol.

The description of this protocol is shown below:

1. $\mathcal{A}$ and $\mathcal{B}$ generate the key pairs (a,P_a) y (b,P_b), where a and b are random numbers which belong to $\mathbb{F}_q$
2. $\mathcal{B}$ sends the key pair (b,P_b) to $\mathcal{A}$
3. $\mathcal{A}$ generates the random number $k \in \mathbb{F}_q$
4. $\mathcal{A}$ calculates the point $P_r = kG$.
5. $\mathcal{A}$ calculates the point $P_h = P + kP_b$, where P is the elliptic curve point to be encrypted
6. $\mathcal{A}$ sends to $\mathcal{B}$ the points (P_r,P_h)

An attacker, in order to obtain the private key b from the public one P_b, should solve the logarithm problem, as both keys are matched by the equation $P_b = bG$. An advantage of this protocol, is that each time it is executed, a key pair is generated, which is more secure; on the other hand, the encrypted message is twice the length of the plain text, which implies an storage increment and a constraint in bandwidth.

Experimental Results

The results shown in the following figures are several analysis which have been carried out in order to compare different public key cryptographic protocols. Normal basis are considered in the evaluated elliptic curve protocols.

We have obtained that Diffie-Hellman and ElGamal protocols over elliptic curve have similar temporal complexity.

Comparing RSA, ElGamal and ECElGamal execution time (depending on the key length in bits) we may observe that the elliptic curve one is the lowest, as it is shown in figure 1.

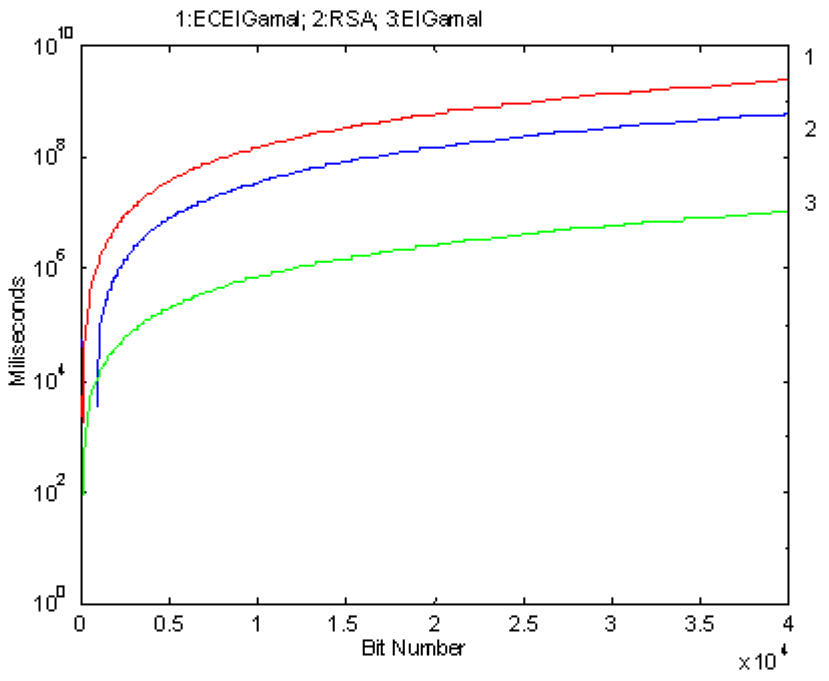

Figure 1: Time complexity among the different algorithms measured

Nevertheless, it must be taken into account that figure 1 must be evaluated considering that the protocols used supply with the same security level. A 1000-bit-length key in RSA provides the same security level than a 150-bit-length one in elliptic curve cryptography algorithms, refer to table 2 obtained from [3] in order to get more information.

Table 2: Key size comparison in *bits*

Reference	**RSA or ElGamal (bits)**	**ECC (bits)**
1	512	106
2	768	132
3	1024	160
4	2048	210
5	21000	600

So, taking into account the security level (from 1-5 in 2), the algorithm speed comparison is depicted in figure 2. It is shown that RSA algorithm is the lowest one, and that ECElGamal is faster than ElGamal.

The results obtained in the experiments have been obtained with a Pentium 133 MHz and a 16 MB RAM, so relative results will be kept when using another platform. On the other hand, execution time and speed depends on the platform chosen.

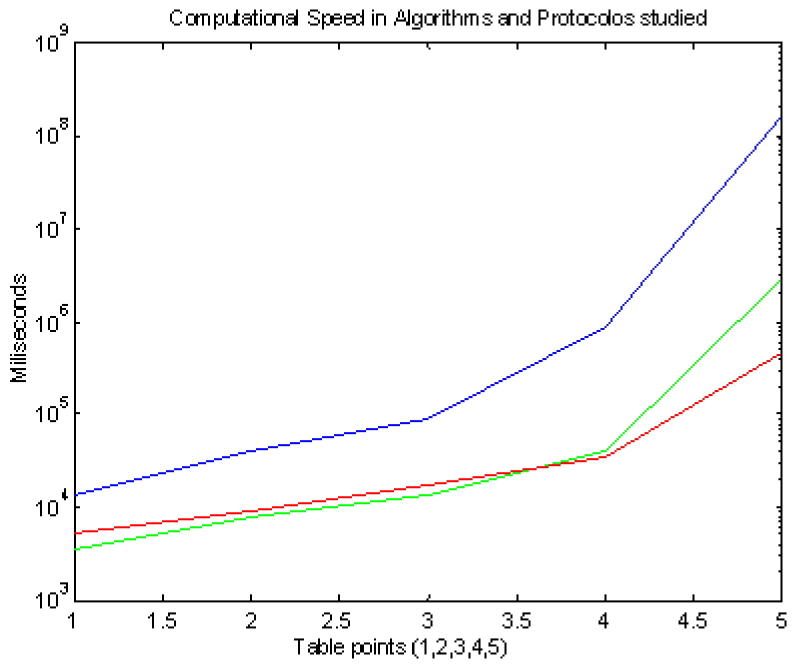

Figure 2: Speed comparison among the different algorithms measured

Conclusions

From the results obtained we may conclude that, among the studied algorithms, the ones which use elliptic curve cryptography have a higher performance, a better time response. On the other hand, when using polynomial basis, memory requirements are higher; nevertheless, when choosing normal basis to represent the different elements in elliptic curve algorithms, storage memory is reduced. Obviously, as in elliptic curve cryptography, key size is smaller than in other public key algorithms, memory requirements are also reduced.

Elliptic curves have been studied for over a hundred of years but it was in 1985 when N. Koblitz and V. Miller proposed, independently, to use the group of points of an elliptic curve defined over a finite field, in cryptography. The youth of public key cryptosystems based on elliptic curves has brought about an important controversy about its safety, nevertheless, the fact is that elliptic curve properties have been studied for over 150 years, but not their application in cryptography. The youth of public key cryptosystems based on elliptic curves has brought about an important controversy about its safety. This skepticism comes from the idea that elliptic curve cryptography has not been tested, cryptoanalized, as much as RSA -based on integer factoring problem (IFP)-, or DSA -based on the discrete logarithm problem (DLP)-. However, algorithms such as ECDSA (digital signature), ElGamal elliptic curve version (encryption) and ECDH (key agreement) are based on ECDLP, a mathematical problem without subexponential-time solution by now (unless for certain kind of elliptic curves, such as supersingular ones).

Reference

1. IEEE P1363a (2001): "Draft Standard Specifications for Public-Key Cryptography" —Amendment 1: Additional Techniques—".
2. Koblitz, N. (1987): "Elliptic Curve Cryptosystems" *Mathematics of Computation*, pp. 203–209.
3. Mohammed, E. and Emarah, E. (2001): "Elliptic Curve Cryptosystems on Smart Cards" *IEEE 7803/01*, pp. 213–222.