PDA implementation of Elliptic Curve Cipher

Diogo Miguel Costa Rodrigues (50702), MEEC, IST
Av. Rovisco Pais, 1049-001 Lisboa, Portugal

Abstract — Thanks to the widespread use of pocket devices, voice has been extended with data communication. By holding sensitive information on their devices, the fear of illegal access and data corruption from intruders was raised by the users.

This paper focuses on the software implementation of a protecting mechanism, based on Elliptic Curve cryptography, for PDA mobile device.

The work, conducted as a Master of Science thesis, depicts some of the main problems of the ECC implementation. It also provides performance results and highlight major restrictions on ECC use in PDAs.

Index Terms— C#, Cipher, Elliptic Curve cryptography, Mobile Devices, PDA

I. INTRODUCTION

In our days the use of electronic equipment for daily routines is essential. The majority of people, who lives in the industrialized world, use computers in their work and in their personal life. Moreover, almost of these computers are connected to the Internet.

The adhesion to Informatics was of such order that raised the need to integrate it with portable devices. In some cases, portable devices are substituting in some tasks the computer of secretary with the advantage of being portable.

But this versatility has raised a new problem. These devices began to keep relevant information, which could be easily stolen by others. And this is the motivation of this work, making the information on these devices safe and inviolable.

One way to protect the information is encoding it. In recent years a new model of security, faster and safer, emerged: the elliptic curve cipher. This cipher needs lesser processing power (faster encryption system) because it uses keys with fewer bits, such as the case of RSA [1], ensuring the same security level.

The elliptic curve cipher is based on a complex mathematical algorithm, using a set of arithmetic operations (addition, subtraction, multiplication and division) on the curve defined by the Weierstrass equation. The security of this cipher is guaranteed due the difficulty of finding the private key without knowing it, needing a high computation system to decrypt the data.

The constant innovation of the technology existing in these devices allows this kind of solutions can be implemented with a specific framework. This framework allows to development, test and deployment of programs into these devices. The mobile devices have a dedicated operating system (like Windows CE [2] or Symbian OS [3]) integrated with their hardware.

Since these devices have a closed architecture, it means that a developer has to take some care when he develops a program. Low memory, weak processing power and the battery issue have to be considered on the creation of software.

This work will be implemented using the dedicated framework existing for these devices (Microsoft’s systems). For compiling and editing the code it will be used the software Microsoft Visual Studio 2005, using the C# [4] as the programming language. The dedicated frameworks used are the Compact Framework 2.0 [5] for pocket devices and Framework 2.0 [6] for PC. It will be necessary that both platforms have these frameworks installed. For the communication between the computer and the PDA [7] is needed to use the application Microsoft ActiveSync [8].

For testing purpose, is used the portable ACER WLMi Travel Mate 4001 with 512MB of RAM with Intel Centrino M 1.5GHz processor and the PDA HTC p3600 Samsung 2442 Travel Mate 4001 with 512MB of RAM with Intel Centrino M 1.5GHz processor and the PDA HTC p3600 Samsung 2442 processor with a 400MHz with 64 MB RAM and 128 MB Flash.

In the next sections will be described the details, theoretical and practical, of the project.

The mathematical basis beyond the cipher and the operations that describes elliptic curves is explained in Section II. This project implements some protocols using the elliptic curve cipher explained on Section III. The last sections describe the implementation of the cipher and are divided in the description of the programming model (Section IV) and the user interface of the application (Section V). Finally, it is present the conclusions of the performance tests realized (Section VI) and the future improvements that could be made on this work (Section VIII).

II. MATHEMATICAL CONCEPTS

In this section will be describe the concepts that are need to implemented the elliptic curve.

A. Finite Fields

The encryption based on elliptic curves operates on numeric finite fields also known as Galois fields [9]. A finite field is an abstract structure where are defined the operations of addition, subtraction, multiplication and division with the particularity of having a finite number of elements. They are represented by
GF(p^n) or \(F_{p^n}\), where \(p\) is a prime number and \(n\) an integer.

The order of the group is defined by \(p^n\) and this can only be equal to a prime number or prime-power order.

**B. Equation of Weierstrass**

An elliptic curve is defined by the equation of Weierstrass [10] and can take various forms depending on the parameters chosen. This equation is given by:

\[
y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6
\]

(1)

Given this equation it is possible to adapt it to have a more efficient implementation by using one specific curve. There are several types of simplifications, but is particularly interesting when the K characteristic of the field is equal to two, such as:

Super-singular Curve \((a_d \neq 0)\) with the equation of the curve given by:

\[
E(x, y) : y^2 + xy = x^3 + ax^2 + b
\]

(2)

Non super-singular Curve \((a_d = 0)\), with the equation:

\[
E(x, y) : y^2 + cy = x^3 + ax^2 + b
\]

(3)

To create operations in a finite field using an elliptic curve there is a need to ensure the Group’s Law. First, we defined an identity known as point at infinity, such that:

\[
P + O_\infty = P
\]

(4)

The inverse of a point in a curve \(P = (x, y)\) is given by:

\[-P = (x, -y)
\]

(5)

The sum operation must also be defined. Given two points \(P = (x_1, y_1)\) and \(Q = (x_2, y_2)\) the sum is defined by:

If \(P \neq Q\):

\[
x_3 = \theta^2 + \theta + x_1 + x_2 + a_2, \theta = \frac{y_2 - y_1}{x_2 - x_1}
\]

(6)

And if \(P = Q = (x, y) = 2P\) (double):

\[
x_3 = \theta^2 + \theta + a_2, \theta = \frac{x + y}{2}
\]

(7)

The negative of one number \(P = (x, y)\) is given by:

\[-P = (x, y + x)
\]

(8)

**C. Base Representation**

A base is a set of vectors, linearly independent that generate a given space. It will be used in this project the polynomial basis and the normal basis in order to accelerate the process of cipher.

1) **Polynomial Basis**

The arithmetic based on polynomial representation is a simple approach because it is based on polynomials. The base polynomial represents the elements of the field in the form \(\{1, \alpha, \alpha^2 \ldots \alpha^{n-1}\}\), which \(\alpha\) is the root of the primitive polynomial of degree \(m\) on the field \(F_{p^n}\).

The operation addition is made by adding two numbers in module \(p\), where \(p\) is the characteristic of the field:

\[a + b = a + b \pmod{p}\]

(9)

In the case where the field is binary, the sum and the subtraction operations are done with the logic operation exclusive-or (XOR).

The multiplication is performed in two stages. First there is a partial adding, a sum of exponents on the basis of the field (if a binary field then the XOR operation is made into the exponents), and finally the element is reduced by module using an irreducible polynomial given by the order of the basis.

The division of polynomials is implemented using the extended Euclidean algorithm, using this algorithm to rewrite the dividend in terms of the divider and the quotient (doing the successive replacement of the rest). There are other possible methods to implement the inversion of a number, as the algorithms of almost inverse of the algorithm of Schroeppel [11] or Itoh-Tsuji inversion algorithm [12].

The advantage of the use of the polynomial basis is that the multiplication is relatively simple. There is also another basis, called normal base also very used in cryptology systems but more designed for implementations in hardware.

2) **Normal Basis**

In a Normal basis, the arithmetic operations are based on logical operations such as AND, XOR, SHIFT and ROTATE.

Considering the field \(F_{p^n}\) one element \(\beta\) can be represented as:

\[e = e_0\beta^n + \ldots + e_2\beta^2 + e_1\beta + e_0\]

(10)

The addition is made as the same way that in basis polynomial using XOR operator. The square of a number is done using the cyclical shift. In this project, the multiplication has been implemented using the algorithm presented in the paper of recommended elliptic curves from the American federal government [13]. It is an operation that needs a pre-processing of a matrix lambda with size of \(p = \text{Type} \times \text{nbits} + 1\).

**D. Auxiliary Algorithms**

In this thesis, along with the cipher of elliptic curve, other algorithms are used for performance comparison. The algorithm AES (Advance Encryption Standard) [14] is used in one of the cipher protocols implemented. It is a cipher block with keys of fixed sizes of 128, 192 or 256 bits. The use of this symmetric algorithm allows accelerating the cipher process keeping the use of elliptical curve. An alternative to the use of this algorithm in the protocol ECIES [15] is to use the algorithm Triple-DES [16].

The HMAC Algorithm [17] generates a message of integrity using a hash function. In project is used the SHA1 [18] function. The result of this algorithm is a fixed block of 160 bits. This mechanism allows the creation of a timestamp for the message. Any change made in the message causes that the result is different forcing the rejection of the data.

It is also used the key derivation function KDF2 [19]. These methods are based on hash functions and create an extra security mechanism. From the keys, common information or
shared secrets is created an additional key increasing the level of security. The implementation is done according to the document IEE P1363A [20].

The hash function SHA1 permits from a set of data to generate a chain of bits of a fixed size. It is a function that works upon data blocks.

Finally, there is also the algorithm not adjacent form (NAF), [21] used in the operator multiplication. Separating in parts the multiplier, it is possible to reduce the number of multiplications needed. The NAF algorithm transforms a number, starting in the binary representation and produces a result in terms of sum and subtraction where the results (in binary) have no consecutive bits.

III. PROTOCOLS

This section deals with the explanation of the various protocols used, for example, how will be done the exchange of keys and what the messages necessary prior to transfer data.

A. Elliptic Curve Integrated Encryption Scheme

The protocol Elliptic Curve Diffie Hellman [22] is used for a rapid exchange of public keys between two entities. First, the two parts decide what are the parameters of that curve will be used (size of the finite field, coefficients of the curve, base of representation). Each entity generates their public and private key (\( Q_A = d_A G \) and \( Q_B = d_B G \)). After exchanging the public keys, both entities can encrypt the data with the key shared:

\[
K = d_A Q_B = d_A d_B G = d_B d_A G = d_B Q_A = K_B.
\]  

B. Elliptic Curve Integrated Encryption Scheme

The protocol Elliptic Curve Integrated Encryption Scheme (ECIES) helps to accelerate the generation of the cipher compared with the traditional method. The ECIES generates a set of data (V, C, T) that will be sent, where V is public key of the sender, C is the encrypted message (using XOR, 3DES or AES), and T is a tag of authentication generated from the message M and the public key of the recipient (MAC).

In this implementation will be used the algorithms KDF2, HMACSHA1 and AES specified in Section 2. AES algorithm may be replaced by XOR operation (small messages with a lower level of security) or the triple-DES algorithm.

The ECIES encryption is given by the following steps:

1) \( Z = \text{Diffie-Hellman Primitive} \)
2) \( K = \text{KDF} (Z) \)
3) \( \text{MaskedEncData} = (\text{message} \oplus \text{K1}) \) (or \( \text{AES(message,K1)} \) or \( \text{AES(message,K2)} \))
4) \( \text{MacTag} = \text{HMACSHA1} (K2, \text{MaskedEncData}) \)
5) \( \text{Send} (Z \parallel \text{MaskedEncData} \parallel \text{MacTag}) = (V, C, T) = (\text{publicKey/secret, AES, MAC}) \)

The ECIES decryption is given by

1) \( Z = \text{Diffie-Hellman Primitive} \)
2) \( K = \text{KDF2} (Z) \)
3) Decrypt using the algorithm AES (ou XOR ou Triple-DES)
4) \( \text{MacTag1} = \text{HMACSHA1} (K2, \text{MaskedEncData}) \)
5) \( \text{Confirm the MacTag2 as the correct one.} \)

C. Elliptic Curve Digital Signature Algorithm

It is possible also use the elliptic curve cipher to create digital signatures. The digital signatures are used to guarantee the authenticity and integrity of a document, detecting any change that is made to the data.

The ECDSA [23] is created by the following steps:

1) Define ECDSA protocol parameters: hash function, elliptic curve (size of the key and basic point of the curve) and the private key (called s).
2) The second step is to Calculate the public key \( Q = s.P \)
3) Calculate the \( R \) point given by the expression: \( R = k.P \) where \( k \) is a random number
4) Evaluate \( c = x \mod n \), where \( x \) is a entire part of \( R \) point and \( d = k^{-1}(e + s.c) \)
5) Send \( (c, d) \)

The signature is verified by the following steps:

1) Using the same parameters from the creation of the signature, calculate \( e = \text{hash(mensagem)} \), \( h = d^{-1} \mod n \), \( h_1 = e'.h \) and finally \( h_2 = (c.h) \mod n \)
2) With the values obtained in the previous point, calculate \( R' = h_1.P + h_2.Q = (x', y') \) and \( c' = x' \mod n \).
3) The digital signature is valid if \( c = c' \)

IV. THE CODE IMPLEMENTATION

Elliptic curve cipher was implemented by using the programming language C# and the Microsoft’s platforms Framework 2.0 and Compact Framework. The code was designed to be as more abstract as possible, to make possible easily improvements. The UML diagram is shown in the Fig. 1 and below is a brief description of each class.
Base Class: stores the common parameters used during the execution. These parameters are generated according to the desired configuration. This class was created because during the execution of the algorithms it is necessary to use constants by the different methods. After choosing the mathematics basis, the protocol and the bit number, the Base Class initializes the lambda array (used by the Normal multiplication), the common parameters (like the word size, size array, last bit or upper shift) and the elliptic curve parameters (base point, curve, co-factor, irreducible polynomial and the order of the curve). Base Class defines the constants TYPEDISPLAY (Hexadecimal by default), TYPEPROTOCOL (ECIES, ECDSA or ECDH), TYPEBASIS (Normal or Polynomial) and TYPECOMMUNICATION (USB [24] or other) specified by the user. This classes defines the location of the XML files (config.xml and ec_parameters.xml) that have the constants mentioned before and the elliptic curve parameters, each one was random generator using a seed and its values follow the American federation recommendation for elliptic curves (the ec_parameters.xml file defines the base point, curve, co-factor, irreducible polynomial and the order of the curve constant parameters for each basis related to the number of bits used in the curve).

Communication Class: is responsible for sending and receiving bytes. This abstract class defines the necessary interfaces (send, receive, sendpublickey e receivepublickey) that must be implemented for each communication protocol (USB, Bluetooth, Infrared or other). For example, the class that implements the communication through USB cable uses the TCP/IP protocols provided by the platform Framework (under a client-server model) to send and receive byte arrays. In the transmission, first byte is reserved to store the number of bytes used in the cipher process in a manner to maintain consistency.

Protocol Class: is an abstract class that serves as a base for implementations of cipher protocols. It has the interfaces of the methods encrypt and decrypt, which are implemented by the different classes (such as ECDH or ECIES).

ECDF Subclass: implements the protocol Diffie-Hellman for the exchange of keys. After choosing the desired configuration, it generates a curve and creates the public and private keys.

ECIES Subclass: implements the ECIES protocol. This class also implements HMAC_SHA1 and ANSI_X9_93 algorithms.

Element Class: is responsible for the generic creation of an element of a base. This abstract class has all the methods necessary to be implemented in any base representation. If the programmer wants to create another base, he has to extends this class and implement all the abstract methods. Using this abstract class, it is possible to make the arithmetic operation without knowing the math base. Element Class also contains methods of bytes manipulation (xor, shift, rotate, setbit, getbit, one, reset, trace, random, new element). Finally, this class implements NAF algorithm.

Normal Subclass: In this class are defined methods needed to perform operations under normal basis. Normal Subclass implements algorithms of multiplication and inversion of two elements and contains the construction method of lambda matrix, which is used in multiplication.

Polynomial Subclass: Implements the algorithms for multiplication (composed by the partial multiplication and reduction of degree using the irreducible polynomial) and the inversion operation (using the Extended Euclidean Algorithm for Inversion in F2m) for the polynomial basis.

Form1 Class: is responsible for the graphical part of the application. It also contains the methods for the selection and initialization of the parameters. The event Load initializes the global variables necessary for the implementation. It also contains the methods associated with buttons “start”, “end”, “send” and “receive”.

BigInteger Class: created by Chew Keong TAN, it is used to operate large dimension math operations.

Curve Class: represents the Weierstrass curve with the ‘a’ and ‘b’ parameters.

Point Class: represents one point which is one coordinate of one curve. It is composed with two Elements that represent the coordinates.

V. THE GRAPHIC USER INTERFACE (GUI)

It was developed two versions of the same program using the same core of code, one version for Windows/PC (using the platform Framework 2 from Microsoft) and another for Pocket PC (with the Compact Framework 2 for this kind of devices). The program implements two tabs that correspond to the different areas: the main and the options area. The “Main tab” is the tab that shows the data to be sent and received. The ciphered text it is divided by color in segments. Each segment corresponds to the keys, data and verification tags. “Main tab” is depicted in Fig. 2.

![Fig. 2 The Main tab (Windows/PC version). Legend: Data to be send (1), Data received (2), Curve’s parameters (3), Irreducible polynomial (4), Type of the Normal Basis (5), Base Point, Private and public key (6), Buttons start, send, receive and end (7), Type of that data to be send (8), Name of the file (9), Open the file dialog (10), Status for debug (11)
The second tab is the “Options Tab”. Here it is possible to change the parameters used by the program (protocol, communication type, math basis and others options) as depicted in Fig. 3.

With the same approach it was developed for the pocket Pc platform another program that uses the same classes as the PC version (despite the GUI interface). Because of the display’s resolution, this version is divided in five tabs. The Curve tab (showing the parameters of curve chosen in the options tab and the data type), the Send and Receive tab (they are similar with the plain text/encrypted data, with the difference of the send/receive button), the Options tab (here the user can choose the Basis, the number of bits and protocol) and finally the Log (shows the system information and errors).

VI. PERFORMANCE TESTS

In this section it is provide one example of the application. In the end of the section is comparing the time results for the different parameters between a PC and a PDA. This test is based on sending one string and one file.

First test sends, through the cable, one string with the characters “THIS IS A TEST” from one PC/Windows to the PDA.

Before sending the information, we need to configure the options in the option tab. This test uses the following parameters, in both devices.

- Basis: Polynomial
- Number of bits: 163,233,283 and 409
- Protocol of the cipher: ECIES

These times results are depicted in the table 1:

<table>
<thead>
<tr>
<th>Bits</th>
<th>PC (T\text{Windows})</th>
<th>ECIES</th>
<th>PocketPC (T\text{WinCE})</th>
<th>T\text{diff} = \frac{T\text{WinCE}}{T\text{Windows}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>163</td>
<td>Public key generation</td>
<td>2.2</td>
<td>43</td>
<td>19.44</td>
</tr>
<tr>
<td></td>
<td>Encrypt the message</td>
<td>2.3</td>
<td>42</td>
<td>18.59</td>
</tr>
<tr>
<td></td>
<td>Decrypt the message</td>
<td>2.2</td>
<td>42</td>
<td>19.35</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>6.6</td>
<td>127</td>
<td>19.12</td>
</tr>
<tr>
<td>233</td>
<td>Public key generation</td>
<td>7.2</td>
<td>149</td>
<td>20.67</td>
</tr>
<tr>
<td></td>
<td>Encrypt the message</td>
<td>7.2</td>
<td>148</td>
<td>20.52</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>21.6</td>
<td>445</td>
<td>20.63</td>
</tr>
<tr>
<td>283</td>
<td>Public key generation</td>
<td>14.2</td>
<td>242</td>
<td>17.10</td>
</tr>
<tr>
<td></td>
<td>Encrypt the message</td>
<td>13.9</td>
<td>243</td>
<td>17.44</td>
</tr>
<tr>
<td></td>
<td>Decrypt the message</td>
<td>14.1</td>
<td>243</td>
<td>17.29</td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>42.1</td>
<td>727</td>
<td>17.28</td>
</tr>
<tr>
<td>409</td>
<td>Public key generation</td>
<td>45.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Encrypt the message</td>
<td>45.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Decrypt the message</td>
<td>46.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total</td>
<td>137.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As expected, the algorithms in one PC are faster than Pocket PC, about 20 times.

The limited resources of PDAs show that, in the current implementation, keys are limited to 283 bits. The security for this key size limitation in EEC cryptography is similar to the security of 1024-bit RSA. Despite heavier computational requirements, EEC does not suffer from known attacks to RSA such as Fermat factorization and Pollard $\rho$-1.

VII. CONCLUSIONS AND FUTURE IMPROVEMENTS

In this work it was demonstrated the viability of EEC implementation on portable devices.

The implementation of new algorithms and new protocols is possible with the minimum of work due to the structure of the application, which uses a level of abstraction to allow future improvements.

There is still a long way to go in the optimization of the various algorithms, this path that was not possible through this work due to lack of time.

As for the performance itself for a key size bigger than 163 bits the cipher is impractical. The multiplication is the heaviest operation and this algorithm should be optimized, since this operation runs several times during the program.
It is also important to note that most of the processing takes place when the keys are generated, not in the encryption process itself.

It is possible to conclude that the elliptic curve is a viable solution for devices with limited processing power but only if they are applied algorithms to speed up the cipher process. Doing this, it is possible to give more fluidity to the cipher / decipher process.

REFERENCES

[13] “RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE”; 1999
[16] NIST, Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, 2004