A Java implementation of the Elliptic Curve Integrated Encryption Scheme

V. Gayoso Martínez¹, L. Hernández Encinas¹, and C. Sánchez Ávila²

¹Department of Information Processing and Coding
Institute of Applied Physics, CSIC, C/ Serrano 144, 28006-Madrid, Spain
{victor.gayoso,luis}@iec.csic.es
Tel. (+34)915618806, Fax (+34)914117651

²Department of Applied Mathematics to Information Technologies
Polytechnic University, Avda. Complutense 30, 28040-Madrid, Spain
carmen.sanchez.avila@upm.es
Tel. (+34)915495700, Fax (+34)913367289

Abstract – Elliptic Curve Cryptography (ECC) is a branch of Cryptography that can be used for encrypting data, generating digital signatures or exchanging keying material during the initial phases of a secure communication. Regarding encryption, the best-known scheme based on ECC is the Elliptic Curve Integrated Encryption Scheme (ECIES). A Java implementation of ECIES is presented in this paper, showing all the options associated to the encryption scheme that can be configured by the user.

Keywords: ECIES, elliptic curves, encryption, Java implementation, public key cryptography.

1. Introduction

Since the development of public key cryptography by Diffie and Hellman in 1976 ([7]), several cryptosystems have been published. In particular, Miller ([14]) and Koblitz ([11]) proposed in 1985 a cryptosystem whose security relies on the Elliptic Curve Discrete Logarithm Problem (ECDLP). So far, no algorithm is known that solves the ECDLP in an efficient way, and some authors consider ([6] and [11]) that this mathematical problem is even more difficult to solve than other mathematical problems (e.g. the Integer Factorization Problem or the Discrete Logarithm Problem) which are used in other cryptosystems. This is the reason why the key length in ECC is significantly smaller than in other cryptosystems as RSA, as it can be observed in Figure 1, which presents a comparison of key lengths associated to the same cryptographic strength for RSA and ECC ([8] and [15]), where the cryptographic strength is interpreted as the security level provided by a symmetric encryption algorithm using a key of $n$ bits.

![Figure 1. Key length comparison](image)

An elliptic curve $E$ over the finite field (or Galois Field) $\mathbb{F}$ is defined by the following equation, known as the Weierstrass equation ([16]):

$$E : y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6,$$  

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ and $\Delta \neq 0$, being $\Delta$ the discriminant of the curve $E$. Condition $\Delta \neq 0$ assures that the curve is smooth, i.e., there are no curve points with two or more different tangent lines. In practice, instead of the Weierstrass equation, simplified versions depending on the characteristic of the finite field $\mathbb{F}$ are used. When the characteristic of the field $\mathbb{F}$ is a prime number, $p$, other than 2 or 3, the finite field (which is
represented as \( GF(p) \) or \( \mathbb{F}_p \) is said to be a prime finite field ([13]). If this is the case, the simplified equation is

\[
y^2 = x^3 + ax + b.
\]  

(2)

If the characteristic of the finite field \( \mathbb{F} \) is 2, then the finite field (represented as \( GF(2^m) \) or \( \mathbb{F}_{2^m} \)) is a binary finite field. The resulting equation in this case is

\[
y^2 + xy = x^3 + ax^2 + b.
\]  

(3)

ECIES is an integrated encryption scheme based on elliptic curves that includes public key operations, encryption algorithms, authentication codes and hash computations. More precisely, ECIES is the generic term used to identify a set of slightly different encryption schemes based on the papers of Mihir Bellare, Philip Rogaway, and Michel Abdalla ([1] and [2]), who developed the encryption scheme DHIES (Diffie Hellman Integrated Encryption Scheme) which represents the kernel of ECIES. For the sake of simplicity, we will refer to those implementations derived from DHIES using the term ECIES for all of them.

Over the years, ECIES has been included in several standards, and currently it can be found in the documents ANSI X9.63 ([4]), IEEE 1363a ([9]), ISO/IEC 18033-2 ([10]), and in some deliverables (e.g., [17] and [19]) from the Standards for Efficient Cryptography Group (SECG).

Figure 2 presents a graphic description of the ECIES encryption process, including the elements and functions involved in the procedure. The steps that must be taken in order to complete the encryption of a clear message are:

1) The sender must create a pair of temporary keys. The temporary private key will be denoted as \( u \), and the temporary public key as \( U \).

2) The sender will use the key agreement function, \( KA \), in order to create a shared secret value, which is the product of the sender’s temporary private key and the recipient’s public key, \( V \).

3) The sender will take the shared secret value (and optionally other parameters) as input data for the key derivation function, denoted as KDF. The output of this function is the concatenation of the encryption key, \( k_{ENC} \), and the MAC key, \( k_{MAC} \).

4) The sender will encrypt the clear message, \( m \), using the ENC symmetric algorithm and the encryption key \( k_{ENC} \). The encrypted message will be represented as \( c \).

5) Taking the encrypted message and the MAC key (and optionally other parameters), the sender will use the selected MAC function in order to produce a \( tag \).

6) Finally, the sender will take the temporary public key, the encrypted message and the tag, and will send the cryptogram consisting of those three concatenated elements, \( U || c || tag \), to the recipient of the message.

To the extent of our knowledge, there is no ECIES software implementation that offers the possibility to set up all the different options and functionalities that must be selected by the user following the recommendations of the most relevant standards. In this paper, we present a Java ECIES implementation where the number of combinations of parameters and functions is certainly high, with the aim to test this encryption scheme and be able to try the different combinations that could be relevant in different deployments.

The rest of this paper is organized in the following way: In Section 2 we present a Java implementation of ECIES developed for PC platforms, the security of ECIES is commented in Section 3, an example of this implementation is included in Section 4 and, finally, Section 5 contains our conclusions.

2. A Java PC implementation of ECIES

The ECIES implementation presented in this contribution is composed of a menu bar and four panels, as it can be seen in Figure 3. The menu bar includes the following items:

- **Program**: The options belonging to this menu entry are Look & Field (which allows the user to select either the Nimbus or the Windows graphical themes), Help (an HTML file with a comprehensive description of the program), About (basic information about the software version and the authors), and Exit.

- **Mode**: This menu option allows the user to choose among the Standard and the Advanced items that represent a different display view. In the Standard view, the Configuration panel and the Profiles and Test menu options are hidden, as in this view a fixed set of parameters for the encryption and decryption processes is used, whereas in the Advanced view those panels and menu options are available to the user.

- **Profiles**: This option includes frequently used sets of parameters (e.g. ISO/IEC and SECG typical configurations).

- **Test**: The selection of one of the options that belong to this menu entry loads a fixed set of parameters and information corresponding to one of the following configurations:

  - Nimbus: This configuration includes the Nimbus graphical theme, a fixed set of parameters, and information about the software implementation.
  - Windows: This configuration includes the Windows graphical theme, a fixed set of parameters, and information about the software implementation.

- **Advanced**: This menu option allows the user to choose among the Standard and the Advanced items that represent a different display view. In the Standard view, the Configuration panel and the Profiles and Test menu options are hidden, as in this view a fixed set of parameters for the encryption and decryption processes is used, whereas in the Advanced view those panels and menu options are available to the user.
eral test cases included in ISO/IEC 18033-2 ([10]) and SECG GEC 2 ([17]).

- **Curves**: By means of this option, the user can load the parameters of several elliptic curves defined over $F_p$ or $F_{2^m}$. These curves have been proposed and published by ANSI ([3]), Brainpool ([5]), NIST ([15]) and SECG ([18]).
- **Tools**: In this menu entry several tools are included, where the most important one is the generator of key pairs that are stored by the program with a proper format as local files.
- **Field**: This menu element allows to work with either the finite field $F_p$ or $F_{2^m}$, using the switching option $GF(p)/GF(2^m)$.

The four panels (*Configuration*, *Parameters*, *Encryption*, and *Decryption*) are described in detail in the next sections.

### 2.1. Configuration panel

Figure 4 presents the elements of this panel. The list of options available in this version of the software for each element is the following:

- KDF function: KDF1 and ANSI.
- Encryption function: XOR, AES in CBC and ECB modes, both with PKCS#5 padding, and 3DES in CBC mode with either PKCS#5 padding or without padding.
- MAC key length in bytes.
- Encryption key length in bytes.
- Option to include the temporary public key of the sender as an input to the KDF function.
- Selection of the first coordinate of the shared secret or its SHA-1 hash value as an input to the
KDF function.
• Interpretation of the KDF function output, which can be either $k_{MAC}||k_{ENC}$ or $k_{ENC}||k_{MAC}$.
• Binary representation of the elliptic curve points (compressed or uncompressed).

2.2. Parameters panel

The Parameters panel includes the values related to the elliptic curve that must be initialized in order to be able to perform the encryption/decryption procedure. Depending on the type of finite field selected, this panel presents a different set of parameters. If the option $GF(p)$ is selected (i.e. the user decides to work with prime finite fields), the elements that are included in the panel represented by the left image in Figure 5 are the following:

- $p$: Prime number characterizing the finite field $\mathbb{F}_p = \{0, 1, 2, \ldots, p-1\}$.
- $a, b$: Elements of the field $\mathbb{F}_p$ that define the elliptic curve whose equation is given by (2).
- $Gx, Gy$: Coordinates of the generator $G$, which is the point of the curve that will be used as a generator of the points representing public keys, where $G = (G_x, G_y)$ and $G_x, G_y \in \mathbb{F}_p$.
- $n$: Prime number whose value represents the order of the point $G$.
- $h$: Cofactor of the curve which is computed as $h = \#E(\mathbb{F}_p)/n$.
- $u, v$: Sender’s temporary private key and receiver’s permanent private key, respectively, with $u, v \in \mathbb{F}_p$.
- $Ux, Uy$: Coordinates of the sender’s temporary public key, where $U = (U_x, U_y)$ and $U_x, U_y \in \mathbb{F}_p$.
- $Vx, Vy$: Coordinates of the receiver’s permanent public key, where $V = (V_x, V_y)$ and $V_x, V_y \in \mathbb{F}_p$.

If the selected option is $GF(2^m)$, then the elements included in the panel (and shown in the right image in Figure 5) are:

- $m$: Exponent that characterizes the finite field $\mathbb{F}_{2^m}$.
- $k_1, k_2, k_3$: Exponents of the irreducible polynomial $f(x) = x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$ used in the operations with polynomial basis.
- $a, b$: Elements of the field $\mathbb{F}_{2^m}$ that define the elliptic curve specified by equation (3).
- $Gx, Gy$: Coordinates of the generator $G$, where $G_x, G_y \in \mathbb{F}_{2^m}$.
- $n$: Order of the point $G$.
- $h$: Cofactor of the curve.
- $u, v$: Sender’s temporary private key and receiver’s permanent private key, where $u, v \in \mathbb{F}_{2^m}$.
- $Ux, Uy$: Coordinates of the sender’s temporary public key, where $U_x, U_y \in \mathbb{F}_{2^m}$.
- $Vx, Vy$: Coordinates of the receiver’s permanent public key, where $V_x, V_y \in \mathbb{F}_{2^m}$.

Independently of the considered finite field, this panel includes the following elements:

- **Format:** Option to present the data strings in either hexadecimal or decimal format.
- **Information:** Output of the process.
- **Generate:** Button that allows the user to compute the coordinates of both points $U$ and $V$ using the generator and the values $u$ and $v$.
- **Delete:** Button for erasing the information of this panel.
2.3. Encryption panel

The Encryption panel (left image in Figure 6) includes the following elements needed to compute the cryptogram corresponding to a given message:

- **Target public key**: Receiver’s permanent public key. If the user clicks on the Text option, the Parameters panel will be activated so the user can enter the recipient’s public key data. On the other hand, if the user clicks on the File option, the user will be prompted to select the file storing the certificate that contains the public key.

- **Plaintext**: Message that the sender wants to transmit to the receiver. The message can be entered either manually or uploaded from a file. Once the message is typed or uploaded, it is shown in the corresponding window.

- **Tag input**: Additional data that is optionally used as input in the MAC function.

- **Cryptogram**: Output of the encryption process, consisting of the concatenation $U||c||tag$, where $U$ is the sender’s temporary public key, $c$ is the message encrypted with the chosen symmetric encryption algorithm, and $tag$ is the output of the MAC function.

- **Information**: Data referred to the output of the encryption process.

The information related to the plaintext and the tag can be displayed (in their respective text boxes) either as regular ASCII text or in hexadecimal format. In comparison, the format options related to the cryptogram are Base64 and hexadecimal.

2.4. Decryption panel

The Decryption panel (right image in Figure 6) includes the following elements needed for the recovery of a plaintext from a cryptogram:

- **Target’s private key**: Receiver’s permanent private key. If the user clicks on the Text option, the Parameters panel will be activated so the user can enter manually his private key data. Besides, if the user clicks on the File option, the private key information will be recovered from a file.

- **Cryptogram**: Data received from the sender, representing the concatenation of elements $U||c||tag$.

- **Plaintext**: Original message that the sender tries to transmit to the receiver.

- **Tag**: Optional data that is used as input in the MAC function.

- **Information**: Data referred to the output of the decryption process.

The data presented in the text boxes related to the plaintext and the tag can be displayed either as regular
ASCII text or in hexadecimal format, while the display options for the cryptogram text box are Base64 and hexadecimal.

3. Attacks and security recommendations

As described in [12], the known attacks on ECIES can be classified as follows:

- Benign and malign malleability attacks.
- Small subgroup attacks.
- Subexponential attacks with supersingular and anomalous curves.

Besides, when implementing ECIES, it is necessary to consider the security implications of decisions such as the usage of the compressed format, the keying material interpretation, the type and number of the optional parameters fed to the KDF and MAC functions, the dynamic selection of parameters for the same public key, etc.

Accordingly to [12], the configuration (functions and parameters) of ECIES used in the example described in Section 4 ensures that no currently known attack can be performed on this implementation of ECIES.

4. Example

Figure 6 provides an encryption and decryption example, where the relevant elements for both processes are the following:

- Target’s public key (in encryption): Retrieved from the file SECG 160 Fp.pub.
- Plaintext (in encryption): Obtained from the file Treasure Island intro.txt, whose content can be seen in the plaintext text box of the left image in ASCII format.
- Tag input (both in decryption and decryption): Text consisting in the sentence “Beginning of Chapter 1”, represented as ASCII text in the left image and in hexadecimal format in the right image of Figure 6.
- Cryptogram (in encryption): File where the output of the encryption process is stored. In this case, the file is Encrypted Treasure Island.bin, and the content can be seen in the cryptogram text box of the left image in Base64 format.
- Target’s private key (in decryption): Uploaded from the file SECG 160 Fp.pri.
- Cryptogram (in decryption): Obtained from the file Encrypted Treasure Island.bin, whose content can be seen in the cryptogram text box of the right image in hexadecimal format.
- Plaintext (in decryption): File where the output of the decryption process is stored. In this case, the file name is Clear intro.txt, and the
content can be seen is the plaintext box of the right image in hexadecimal format.

The combination of parameters used in this encryption/decryption process (hash function, etc.) is presented in Figure 4. The public and private keys (defined in the example over a prime field $\mathbb{F}_p$), and the rest of data elements related to the elliptic curve that are necessary in the computations, are shown in the left image of Figure 5.

5. Conclusions

As we have commented during this contribution, ECIES is an encryption scheme whose implementation details vary depending on the standard where it is included. The nature and number of differences prevent a developer from building a software implementation compatible with all the relevant security standards, so the designer must take some decisions based on the availability and efficiency of the specific functions included in the target platform.

We have developed a Java encryption software that can be used to test different parameter combinations and decide which is the best one in a particular deployment scenario of the ECIES encryption scheme. Performance tests can be conducted by any third party using this tool in order to determine the proper finite field, key length, KDF function, etc.

Additionally, our ECIES software can be used to check that a certain implementation adjusts to one or several of the standards mentioned in this contribution.

Acknowledgements. This work has been partially supported by Ministerio de Ciencia e Innovación (Spain) under the grant TEC2009-13964-C04-02 and Ministerio de Industria, Turismo y Comercio (Spain), in collaboration with CDTI and Telefónica I+D under the project Segur@ CENIT-2007 2004.

The authors would like to thank the anonymous reviewers for their helpful comments.

References


