Analysis of ECIES and Other Cryptosystems Based on Elliptic Curves

V. Gayoso Martínez, F. Hernández Álvarez, L. Hernández Encinas
Department of Information Processing and Coding
Applied Physics Institute, CSIC, Madrid, Spain
{victor.gayoso,fernando.hernandez,luis}@iec.csic.es

C. Sánchez Ávila
Department of Applied Mathematics to Information Technologies
Polytechnic University, Madrid, Spain
carmen.sanchez.avila@upm.es

Abstract: Elliptic Curve Cryptography (ECC) can be used as a tool for encrypting data, creating digital signatures or performing key exchanges. Regarding the encryption procedure, the schemes currently used are known as hybrid cryptosystems, as they use both symmetric and asymmetric techniques. Among those hybrid cryptosystems based on ECC, the Elliptic Curve Integrated Encryption Scheme (ECIES) is the best known, and as such it can be found in several cryptographic standards. In this work, we present an extensive review and comparison of the versions of ECIES included in documents from ANSI, IEEE, ISO/IEC, and SECG, highlighting the main differences between them that may prevent fully interoperable implementations of ECIES. In addition, a detailed list of the functions and capabilities needed by ECIES and available in Java Card is presented, which allows to provide some conclusions about the practical limitations of a Java Card implementation of ECIES.

Keywords: ECIES, elliptic curves, encryption, hybrid cryptosystem, public key cryptography, Java Card.

1 Introduction

Since the development of public key cryptography by Whitfield Diffie and Martin Hellman in 1976 [1], several cryptosystems have been proposed. Security and efficiency are the most important features to be requested to any cryptosystem and, in general, both characteristics depend on the mathematical problem on which it is based. The list of problems that are currently considered computationally infeasible to solve includes the Integer Factorization Problem (IFP), the Discrete Logarithm Problem (DLP), and the Elliptic Curve Discrete Logarithm Problem (ECDLP).

In 1985, Victor Miller [2] and Neal Koblitz [3] independently proposed a cryptosystem based on elliptic curves defined over finite fields, whose security relies on the ECDLP [4]. In comparison with other cryptosystems (e.g. RSA), Elliptic Curve Cryptography (ECC) uses significantly shorter keys. The reason for this fact is related to the hardness of the ECDLP, which is considered by some authors to be more difficult to solve than the IFP or the DLP [3, 5].

As it is well-known, an elliptic curve, E, over a finite field, $\mathbb{F}$, can be defined by the Weierstrass equation [6], whose affine expression is as follows:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad (1)$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}$ and $\Delta \neq 0$, being Δ the discriminant of the curve [4]. The last condition assures that the curve has no points with two or more different tangent lines.

In practice, the equation (1) is not used, and the following simplified equations with affine coordinates are used depending on the characteristic of the finite field $\mathbb{F}$ where the elliptic curve is defined:

- If the finite field is a prime field, i.e. $\mathbb{F} = \mathbb{F}_p$, where $p > 3$ is a prime number, the equation defining the (non-supersingular) elliptic curve becomes:

$$y^2 = x^3 + ax + b. \quad (2)$$

- If the finite field is a binary field, i.e. $\mathbb{F} = \mathbb{F}_{2^m}$, where m is an integer number, then the equation of the (non-supersingular) elliptic curve is:

$$y^2 + xy = x^3 + ax^2 + b. \quad (3)$$

There are other representation systems that use projective coordinates and homogeneous equations (standard projective coordinates, Jacobian coordinates, Lopez-Dahab coordinates, mixed coordinates, etc.) [7, 8]. Point arithmetic in these systems is more efficient than in the affine system since inversions, the most expensive operations from a computational point of view with finite field elements, are not necessary in projective coordinates. Therefore, this is one of the research areas where more practical developments have been achieved in recent years [9, 10].

The most extended encryption scheme in ECC is the Elliptic Curve Integrated Encryption Scheme (ECIES). In this contribution, we present an extensive review and comparison of the versions of ECIES included in documents from ANSI, IEEE, ISO/IEC, and SECG, highlighting the main differences

between them that may prevent fully interoperable implementations of ECIES. In addition, a detailed list of the functions and capabilities needed by ECIES and available in Java Card is presented, providing some conclusions about the practical limitations of a Java Card implementation of ECIES.

This paper is an extended and more complete version of [11], and is organized as follows: Section 2 describes the most important cryptosystems based on elliptic curves. Section 3 presents the functional design of ECIES. Section 4 provides a comparison of the different versions of ECIES included in the ANSI X9.63, IEEE 1363a, ISO/IEC 18033-2, and SEC 1 documents. Section 5 includes the specific functions allowed in those standards. In Section 6 we describe the functions and capabilities available in the latest versions of Java Card (2.2, 2.2.1, 2.2.2, and 3.0) that are necessary in order to implement ECIES in a smart card. Finally, we will summarize the most important findings and conclusions in Section 7.

2 Elliptic Curve Cryptosystems

2.1 Early Cryptosystems

The first encryption schemes based on elliptic curves were the equivalent versions of the Massey-Omura [12] and ElGamal [13] cryptosystems, both presented by Koblitz in 1985 (and published in 1987) [3], and the Menezes-Vanstone cryptosystem [14].

One of the main disadvantages of the Massey-Omura and ElGamal versions adapted for elliptic curves is that plaintexts and encrypted messages must be represented as points of an elliptic curve E. This disadvantage, when using elliptic curves whose order $\#E$ is a high value, produces a limitation which is more theoretical than practical. However, the requirement to build tables stating the relationship between every possible message and its related elliptic curve point limits the usefulness of these cryptosystems to closed environments (enterprises, small groups, etc.) where all possible messages are previously established.

The Menezes-Vanstone cryptosystem for elliptic curves was designed precisely to overcome this limitation, as instead of matching each message with a point on the curve E, it represents the plaintexts as ordered pairs of $\mathbb{F}^* \times \mathbb{F}^*$, where $\mathbb{F}^* = \mathbb{F} \setminus \{0\}$ and those pairs do not necessarily have to represent the coordinates of an elliptic curve point. Using this cryptosystem, it is possible to divide any plaintext in blocks, where each block could be easily encoded as an ordered pair. The disadvantage of this procedure is that, instead of transforming each clear message into a single point of the curve (where the binary representation of every point of the curve has the same length), the size of the encrypted message depends directly on the length of the plaintext.

In the Menezes-Vanstone cryptosystem, the expansion factor (i.e. the ratio between the size of the cryptogram and the length of the clear message) is 2, since a plaintext $x = (x_1, x_2)$, consisting of two elements of the finite field $\mathbb{F}^*$, produces the cryptogram (Y_0, y_1, y_2), where $y_1, y_2 \in \mathbb{F}$ and Y_0^* is an elliptic curve point consisting in two coordinates that belong to the finite field, so in summary the total number of finite elements to be transmitted is 4. When using point compression in order to decrease the length of the information to be transmitted (i.e. only the first coordinate of the point is sent along with an additional byte that includes the necessary data for recovering the whole point), the expansion factor is reduced to approximately 1.5.

In comparison, the expansion factor of the variants of the Massey-Omura and ElGamal cryptosystems is 2, as the clear message is considered to be a point of the curve and, in each of those cryptosystems, it is necessary to transmit 2 elliptic curve points. In practice, a high value for the expansion factor implies that the encryption of the information generates cryptograms much larger than those produced when using a symmetric key algorithm like AES.

Another disadvantage resulting from the design of the Menezes-Vanstone cryptosystem is that it is necessary to make operations with the points of the elliptic curve in each encryption process. As depending on the plaintext length it is necessary to divide the clear message in multiple segments and perform asymmetric encryption operations with each of those segments, the performance of the Menezes-Vanstone scheme degrades much faster than with a symmetric encryption algorithm when the number of segments increases.

In addition to the previously mentioned practical disadvantages, Klaus Kiefer showed in 1998 that, under certain conditions, this cryptosystem is insecure [16]. Kiefer also demonstrated that, contrary to the terms of its specification, the Menezes-Vanstone cryptosystem cannot be considered a probabilistic encryption algorithm.

2.2 Hybrid Cryptosystems

Due to the reasons mentioned in the previous section, over the years the academic community abandoned the study of the three initial cryptosystems based on elliptic curves. As an illustrative example, while in the first edition of the work by Douglas Stinson [17] both the ElGamal and the Menezes-Vanstone cryptosystems for elliptic curves were included, in the second and third editions these schemes were replaced by ECIES. Even in one of the latest books about this subject, co-authored by Alfred Menezes and Scott Vanstone [8], the Menezes-Vanstone scheme is not included.

However, the discovery of the limitations of these early cryptosystems did not imply the abandonment of the search for a practical and secure elliptic curve cryptosystem, as it only caused a change of direction, occupying now the spotlight the hybrid encryption schemes, which bring the best characteristics of both symmetric and asymmetric cryptography. The most important hybrid schemes that use elliptic curves are ECIES, PSEC (Provably Secure Elliptic Curve encryption scheme) [18, 19], and ACE (Advanced Cryptographic Engine) [19, 20].

Of the three schemes, ECIES is available in a greater number of standards (ANSI X9.63 [21], IEEE 1363rd [22], ISO/IEC 18033-2 [23], and SECG SEC 1 [24]). PSEC can be found in ISO/IEC 18033-2 [23], IETF RFC 4051 [25], and the set of algorithms selected for the NESSIE (New European Schemes for Signatures, Integrity and Encryption) project [26, 27], while ACE is available in ISO/IEC 18033-2 [23], and the final selection of NESSIE [26, 27].

After reviewing the three schemes, the main differences that can be identified are the following:

- In both PSEC and ECIES, the recipient's public key is a point on the elliptic curve, $V = v \cdot G$, where v is the re-

cipient's private key, and G is the generator of the group of points of the cyclic subgroup used in the computations, while in the ACE scheme the public key consists of four elliptic curve points, so $V = (W, X, Y, Z)$.

- PSEC uses twice a key derivation function in order to obtain a pair of MAC and symmetric encryption keys, whilst ACE and ECIES use such a function only once.
- ECIES and ACE use the first coordinate of a point of the curve generated during the calculations (instead of both coordinates) as an input parameter to the key derivation function previously mentioned, while PSEC requires to use both coordinates.
- PSEC is the only scheme that uses the XOR function during the key generation process (regardless of its usage as a symmetric encryption function).
- Cryptograms in ECIES consist of three elements (the sender's ephemeral public key, the encrypted message, and a MAC code), while cryptograms in PSEC include one additional element, a binary string, and in ACE the cryptograms include two additional elliptic curve points.

Making a comparison of the three schemes ECIES offers the best-balanced set of features, providing a secure and flexible solution for the encryption of data.

3 Elliptic Curve Integrated Encryption Scheme (ECIES)

3.1 ECIES Versions

In 1997, Mihir Bellare and Philip Rogaway [28] presented the Discrete Logarithm Augmented Encryption Scheme (DLAES), which was subsequently improved by the same authors and Michel Abdalla, being first renamed as the Diffie-Hellman Augmented Encryption Scheme (DHAES) in 1998 [29] and later as the Diffie-Hellman Integrated Encryption Scheme (DHIES) in 2001 [30], in order to avoid confusions with the Advanced Encryption Standard (AES). DHIES represents an enhanced version of the ElGamal encryption scheme, using elliptic curves in an integrated scheme which includes public key operations, symmetric encryption algorithms, MAC codes, and hash computations. Because of the integration of different functions, DHIES is secure against chosen ciphertext attacks without having to increase the number of operations or the key length [30].

Figure 1 presents the version of DHIES included in [30], where M represents the clear message, g is a generator of a multiplicative cyclic group G, g^v and v are the recipient's public and private keys, respectively, and g^u and u are the sender's ephemeral public and private keys, respectively. We denote by $\mathcal{E}$ the symmetric encryption algorithm, while $\mathcal{T}$ is the MAC code generation function, and H is the hash function.

DHIES was evaluated by ANSI and included with some modifications in the ANSI X9.63 standard [21] in 2001. Independently, IEEE had released in 2000 the IEEE 1363 standard [31], so when ANSI X9.63 was made public, the group of experts of IEEE reviewed both DHIES and the version proposed by ANSI, and included a modified version in the amendment IEEE 1363a [22] released in 2004.

During those years, another ISO/IEC expert group was collaborating in the creation of the set of standards to be known as the 18033 family, so they took as starting point all the previous versions of ECIES and the existing studies about its security, and produced another slightly different version of ECIES, which was included in the ISO/IEC 18033-2 standard [23] in 2006.

Before finishing this compilation of ECIES versions, it is necessary to point out the Standards for Efficient Cryptography Group (SECG), an industry consortium, that included ECIES in the SEC 1 document [24] released in 2000 (version 1.0) and updated in 2009 (version 2.0).

In order to avoid misunderstandings, all these versions of the encryption scheme receive along this contribution the generic name of ECIES, even though the versions are not identical.

3.2 ECIES Functional Components

As it name indicates, ECIES is an integrated encryption scheme that uses the following functions:

- Key Agreement (KA): Function used by two parties for the creation of a shared secret.
- Key Derivation (KDF): Mechanism that produces a set of keys from keying material and some optional parameters.
- Hash (HASH): Digest function.
- Encryption (ENC): Symmetric encryption algorithm.
- Message Authentication Code (MAC): Information used to authenticate a message.

After comparing the ECIES versions defined by ANSI, IEEE, ISO/IEC, and SECG, two main groups of differences are clearly detectable:

- Functionality: Implementation details, usage of optional elements, binary representation conventions, etc.
- Allowed functions: Specific functions (HASH, ENC, etc.) allowed in each standard.

4 ECIES Functionality Comparison

In this section, we analyse in pairs the main differences (in terms of functionality) among the different standard implementations of ECIES.

4.1 DHIES and ANSI X9.63

The following list presents the most relevant differences between the original DHIES specification [30] and the version implemented in the ANSI X9.63 standard [21].

- DHIES does not allow arbitrary parameters in neither the KDF nor the MAC function, whilst X9.63 does allow parameters in both functions.
- DHIES uses a HASH function to produce the MAC and ENC keys. In comparison, ANSI X9.63 uses a KDF construction where the data is processed during several rounds.

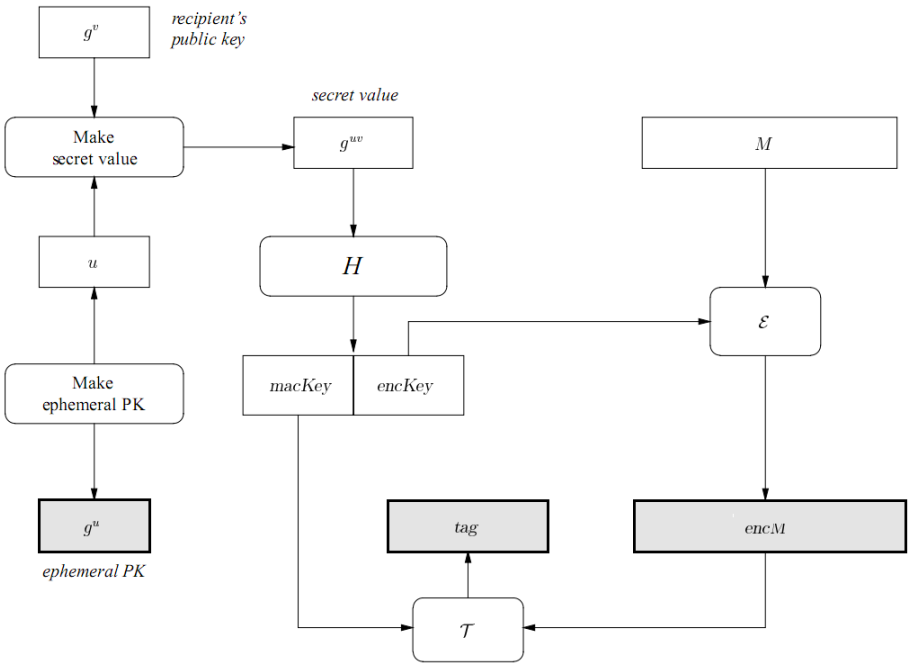

Figure 1: DHIES functional diagram.

- DHIES interprets the leftmost bits of the output of the KDF function as the MAC key, and the rightmost bits as the ENC key. In ANSI X9.63, the order is precisely the opposite.

- ANSI X9.63 only allows to use the XOR function as the symmetric encryption cipher. In comparison, DHIES offers to use either a stream or a block cipher, leaving open the option of the specific cipher to use.

4.2 ANSI X9.63 and IEEE 1363a

The list included hereafter reflects the main differences between the ECIES implementations included in the ANSI X9.63 [21] and IEEE 1363a [22] standards.

- ANSI X9.63 allows to use an arbitrary parameter as an input to the KDF function, but does not mention the content of that optional parameter. In comparison, the so-called DHAES mode in IEEE 1363a mandates to use the binary representation of the sender's public key as an input parameter.

- IEEE 1363a interprets the leftmost bits of the output of the KDF function as the MAC key, and the rightmost bits as the ENC key when using a stream cipher, and the opposite when using a block cipher. In comparison, ANSI X9.63 always interprets the ouput as $k_{ENC}||k_{MAC}$.

4.3 IEEE 1363a and ISO/IEC 18033-2

The following list presents the main differences between the IEEE 1363a [22] and ISO/IEC 18033-2 [23] standards.

- ISO/IEC 18033-2 does not allow parameters in the KDF function, whereas IEEE 1363a allows the usage of parameters in that function.

- IEEE 1363a includes the option to use either bit or byte strings, whilst ISO/IEC 18033-2 mandates the usage of byte strings.

- IEEE 1363a suggests to use always the same set of parameters and functions for a given public key. In comparison, ISO/IEC 18033-2 mandates not to change under any circumstance those parameters for the same receiver's public key.

- IEEE 1363a states that the minimum key length must be 160 bits. In contrast, ISO/IEC 18033-2 does not mention any minimum key length.

It is worth mentioning that there were more differences between IEEE 1363a and the first draft version of ISO/IEC 18033-2 [19]. These differences were removed in the final version of the document in order to obtain a higher level of compliance with previous standards and implementations. Those differences were the following:

- In the draft version of ISO/IEC 18033-2, it was mandatory to use the ephemeral public key of the sender as an input for the KDF function, but in the final version this feature is optional. In comparison, IEEE 1363a includes an option (the non-DHAES mode) that does not use that public key.

- The draft version of ISO/IEC 18033-2 stated that it was necessary to include as an input for the MAC funcion the length of the tag string that is attached to the encrypted message before the MAC computation takes place. In the final version of the standard, two modes were included (DEM2 and DEM3) not using this tag length. In IEEE 1363a, again the non-DHAES mode offers the option of not including this length.

- ISO/IEC 18033-2 presents the option of using the XOR function as a stream cipher as part of the DEM3 mode, although this possibility was clearly forbidden in the initial draft version. IEEE 1363a allows both block ciphers and stream ciphers.

4.4 ISO/IEC 18033-2 and SECG SEC 1

The following list provides the main differences between the ECIES implementations included in the ISO/IEC 18033-2 standard [23] and the SEC 1 document [24].

- ISO/IEC 18033-2 does not allow input parameters in the KDF function, whilst SEC 1 allows to include this additional information, even though in the test vectors included in the GEC 2 document [32] no additional parameters have been used.
- SEC 1 does not explicitly include the sender's ephemeral public key in the KDF computation. However, it mentions that the public key could be one of the elements used as input parameters in that function.
- ISO/IEC 18033-2 does not mention minimum key lengths, whereas SEC 1 states that the selection of the field must be guided by the following requirements:

$$\mathbb{F}_p\text{: } \lceil \log_2 p \rceil \in \{192, 224, 256, 384, 521\}.$$
$$\mathbb{F}_{2^m}\text{: } m \in \{163, 233, 239, 283, 409, 571\}.$$

5 Allowed Functions Comparison

This section presents the comparison of allowed KA, KDF, HASH, ENC, and MAC functions included in the aforementioned standards.

Table 1 shows the different KA functions allowed in ECIES. Here DH denotes the Diffie-Hellman key agreement function, whilst DHC is the Diffie-Hellman function using the cofactor of the elliptic curve.

X9.63	1363a	18033-2	SEC 1
DH	DH	DH	DH
DHC	DHC	DHC	DHC

Table 1: KA functions.

The KDF functions considered in ECIES are presented in Table 2, where X9.63-KDF is the KDF function defined in the ANSI X9.63 standard, KDF1 and KDF2 are functions defined by the ISO/IEC 18033-2 document, and NIST-800-56 is the KDF concatenacion function specified in [33].

X9.63	1363a	18033-2	SEC 1
X9.63-KDF	X9.63-KDF	KDF1	X9.63-KDF
		KDF2	NIST-800-56

Table 2: KDF functions.

Table 3 presents the HASH functions used in ECIES. SHA-1 is the well-known digest function included in [34]; SHA-2 represents the family composed by SHA-256, SHA-384, and SHA-512 [34]; SHA-2* is the SHA-2 family with the addition of the SHA-224 hash algorithm [34]; RIPEMD is the set of hash algorithms defined in [35]; and WHIRLPOOL is the function defined in [36].

The symmetric ciphers considered in ECIES are shown in Tables 4 and 5, where XOR$^\perp$ represents the ISO/IEC 18033-2 method that uses the XOR function together with a KDF function in order to derive a XOR key of variable length; TDES is the Triple DES algorithm in CBC mode [37]; AES represents the Advanced Encryption Standard family, that is, AES-128, AES-192, and AES-256; and MISTY1, CAST-128, Camellia, and SEED are the algorithms specified in [38], [39], [40], and [41], respectively.

X9.63	1363a	18033-2	SEC 1
SHA-1	SHA-1	SHA-1	SHA-1
SHA-224	SHA-256	SHA-256	SHA-224
SHA-256	SHA-384	SHA-384	SHA-256
SHA-384	SHA-512	SHA-512	SHA-384
SHA-512	RIPEMD-160	RIPEMD-128	SHA-512
		RIPEMD-160	
		WHIRLPOOL	

Table 3: HASH functions.

ANSI X9.63	IEEE 1363a
XOR	XOR
	TDES/CBC/PKCS5
	AES/CBC/PKCS5

Table 4: ENC functions (I).

ISO/IEC 18033-2	SECG SEC 1
XOR	XOR
XOR$^\perp$	TDES/CBC
TDES/CBC/PKCS5	AES/CBC
AES/CBC/PKCS5	AES/CTR
MISTY1/CBC/PKCS5	
CAST-128/CBC/PKCS5	
Camellia/CBC/PKCS5	
SEED/CBC/PKCS5	

Table 5: ENC functions (II).

Tables 6 and 7 show the allowed MAC functions. HMAC-SHA-1 and HMAC-RIPEMD are defined in [42]; HMAC-SHA-2 represents the family of HMAC algorithms, i.e., HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512, described in [43]; HMAC-SHA-2* is the same as HMAC-SHA-2 with the addition of the HMAC-SHA-224 function [43]; and CMAC-AES is the set of AES-related HMAC functions, that is, CMAC-AES-128, CMAC-AES-192, and CMAC-AES-256, included in [44].

Finally, Table 8 presents all the cryptographic functions and algorithms allowed simultaneously in the four standard versions of ECIES cited along this document.

6 ECIES in Java Card

Smart cards are considered tamper-resistant, portable storage devices that can enhance the security of different systems and procedures (e.g. client authentication [45]). Java Card is a framework for the programming and execution of applications in smart cards developed by Sun with the support from several leading smart card providers.

ANSI X9.63	IEEE 1363a
H-SHA-1	H-SHA-1
H-SHA-224	H-SHA-256
H-SHA-256	H-SHA-384
H-SHA-384	H-SHA-512
H-SHA-512	H-RIPEMD-160

Table 6: MAC functions (I).

ISO/IEC 18033-2	SECG SEC 1
H-SHA-1	H-SHA-1-80/160
H-SHA-256	H-SHA-224-112/224
H-SHA-384	H-SHA-256-128/256
H-SHA-512	H-SHA-384-192/384
H-RIPEMD-128	H-SHA-512-256/512
H-RIPEMD-160	CMAC-AES-128/192/256
H-WHIRLPOOL	

Table 7: MAC functions (II).

Even though Java Card version 2.1 included some cryptographic capabilities, the support for ECC was not included until version 2.2, so this section is focused in the comparison of the functionality needed by ECIES and available in Java Card versions 2.2, 2.2.1, 2.2.2, and 3.0.

6.1 Java Card 2.2 and 2.2.1

Java Card 2.2 implements the following functionality related to ECIES:

- HASH function: SHA-1.
- KA function: DH and DHC, with the peculiarity that, instead of obtaining the first coordinate of the product of the sender's ephemeral private key and the recipient's public key, these functions provide the SHA-1 output of that result.
- ENC function: AES with key lengths of 128, 192, and 256 bits in CBC and ECB modes, both without padding.
- ECC key length: 113, 131, 163, and 193 bits in binary fields, and 112, 128, 160, and 192 bits in prime fields.

Java Card 2.2.1 did not present any change in the list of functions and available key lengths related to ECIES.

6.2 Java Card 2.2.2

The main novelties in Java Card 2.2.2 were the HMAC and SHA-2 group of functions. The complete list of functions and key lengths related to ECIES in this version of Java Card is the following:

- HASH function: SHA-1, SHA-256, SHA-384, and SHA-512.
- KA function: DH and DHC, with the peculiarity already present in Java Card 2.2 and 2.2.1.
- ENC function: AES with key length 128, 192, and 256 bits in CBC and ECB mode, both without padding.
- MAC function: HMAC-SHA-1-64, HMAC-SHA-256-64, HMAC-SHA-384-128, and HMAC-SHA-512-128.
- ECC key length: 113, 131, 163, and 193 bits in binary fields, and 112, 128, 160, and 192 bits in prime fields.

HASH	*KA*	KDF	*ENC*	*MAC*
SHA-1	DH	KDF2	XOR	HMAC-SHA-1
SHA-256	DHC			HMAC-SHA-256
SHA-384				HMAC-SHA-384
SHA-512				HMAC-SHA-512

Table 8: Common functions allowed in the standards.

6.3 Java Card 3.0

Java Card 3.0 has extended the number of padding modes in the ENC function, as well as the possible key lengths available in the prime fields $\mathbb{F}_p$. The complete list of Java Card 3.0 functionalities related to ECIES is the following:

- HASH function: SHA-1, SHA-256, SHA-384, and SHA-512.
- KA function: DH and DHC, with the modes included in the previous versions and also a new mode that provides directly the product of the sender's ephemeral private key and the recipient's public key.
- ENC function: AES with key lengths of 128, 192, and 256 bits in CBC and ECB modes, both either without padding or with PKCS#5 padding.
- ECC key length: 113, 131, 163, and 193 bits in binary fields, and 112, 128, 160, 192, 224, 256, and 384 bits in prime fields.
- MAC function: HMAC-SHA-1, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512.

6.4 Java Card Summary

Tables 9 and 10 show a summary of the information presented in the previous sections, where DHC y DH are the Diffie-Hellman functions with and without cofactor, respectively, and DHC* y DH* are the same functions with the particularity explained in §6.1.

	Java Card 2.2	Java Card 2.2.1
HASH	SHA-1	SHA-1
KA	DH* DHC*	DH* DHC*
ENC	DES TDES AES-128	DES TDES AES-128
MAC		
$\mathbb{F}_p$	112,128,160,192	112,128,160,192
$\mathbb{F}_{2^m}$	113,131,163,193	113,131,163,193

Table 9: ECC functionality in Java Card (I).

	Java Card 2.2.2	Java Card 3.0
HASH	SHA-1 SHA-256 SHA-384 SHA-512	SHA-1 SHA-224 SHA-256 SHA-384 SHA-512
KA	DH* DHC*	DH* DHC* DH DHC
ENC	DES TDES AES-128	DES TDES AES-128 AES-192 AES-256
MAC	HMAC-SHA-1 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512	HMAC-SHA-1 HMAC-SHA-256 HMAC-SHA-384 HMAC-SHA-512
$\mathbb{F}_p$	112,128,160,192	112,128,160,192, 224,256,384
$\mathbb{F}_{2^m}$	113,131,163,193	113,131,163,193

Table 10: ECC functionality in Java Card (II).

7 Conclusions

After reviewing the different versions of ECIES included in the standards, it is possible to obtain the following conclusions:

1. ECIES is the best known encryption scheme based on elliptic curves, and it is included in ANSI X9.63, IEEE 1363a, ISO/IEC 18033-2, and SECG SEC 1. Even though those versions are ultimately based on the DHIES scheme, there are important differences regarding the functionality and the specific functions allowed by each standard.

2. All the standards allow to use the XOR function as the symmetric encryption algorithm. However, in order to avoid security problems when using this function, the message length should have a fixed value, which limits dramatically the practicality of the encryption scheme [19]. The best solution, given these circumstances, consists in using a symmetric block cipher instead of the XOR function.

3. Besides, the implementation in devices such as Java Cards faces another important problem: the limitation in the functions available to the developer in the programming interface.

4. The direct consequence of the previous comments is that, when implementing ECIES, the first step should be to evaluate the capabilities provided by the final platform and, from that point, decide which version of ECIES to implement. Regarding this point, even though the newer versions (e.g. ISO/IEC 18033-2 and SEC 1) may not be fully compatible with legacy devices, they provide access to the most recent and secure functions (e.g. SHA-2, AES, etc.), so it should be recommended to use one of these versions.

Acknowledgment

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.

References

[1] W. Diffie and M. Hellman. New Directions in Cryptography, IEEE Transactions on Information Theory, 22 (6), pp. 644–654, 1976.

[2] V. Miller. Use of Elliptic Curves in Cryptography, Lecture Notes in Computer Science, 218, pp. 417–426, 1986.

[3] N. Koblitz. Elliptic Curve Cryptosystems, Mathematics of Computation, 48 (177), pp. 203–209, 1987.

[4] A. Menezes. Elliptic Curve Public Key Cryptosystems, Kluwer Academic Publishers, Boston, 1993.

[5] BSI TR-03111. Elliptic Curve Cryptography (v. 1.11), Bundesamt für Sicherheit in der Informationstechnik, 2009.

[6] J. H. Silverman. The Arithmetic of Elliptic Curves (2nd ed.), Springer-Verlag, New York, 2009.

[7] H. Cohen et al. Handbook of Elliptic and Hyperelliptic Curve Cryptography, Chapman & Hall/CRC, Florida, 2006.

[8] D. Hankerson, A. Menezes, S. Vanstone. Guide to Elliptic Curve Cryptography, Springer-Verlag, New York, 2004.

[9] Q. Abu Al-Haija' and M. Al-Khatib. Parallel Hardware Algorithms & Designs for Elliptic Curves Cryptography to Improve Point Operations Computations Using New Projective Coordinates, Journal of Information Assurance and Security, 5 (6), pp. 588–594, 2010.

[10] Q. Abu Al-Haija' and L. Tawalbeh, Efficient Algorithms & Architectures for Elliptic Curve Crypto-Processor Over GF(p) Using New Projective Coordinates Systems, Journal of Information Assurance and Security, 6 (1), pp. 63–72, 2011.

[11] V. Gayoso Martínez et al. A Comparison of the Standardized Versions of ECIES. In Proceedings of the 6th International Conference on Information Assurance and Security (IAS 2010), pp. 1–4, Atlanta, 2010.

[12] OMNET Associates. Method and Apparatus for Maintaining the Privacy of Digital ;essages Conveyed by Public Transmission. Inventors: J. L. Massey, J. K. Omura. Filling date: 1982-09-14. Issue date: 1986-01-28. United States patent 4.567.600.

[13] T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms, IEEE Transactions on Information Theory, 31 (4), pp. 469–472, 1985.

[14] A. Menezes and S. Vanstone. Elliptic Curve Cryptosystems and Their Implementation, Journal of Cryptology, 6 (4), pp. 209–224, 1993.

[15] H. Pietiläinen. Elliptic Curve Cryptography on Smart Cards, Ph.D. Thesis, Faculty of Information Technology, Helsinki University of Technology, Finland, 2000.

[16] K. Kieffer. A Weakness of the Menezes-Vanstone Cryptosystem, Lecture Notes in Computer Science, 1361, pp. 201–206, 1998.

[17] D. Stinson. Cryptography: Theory and Practice (3rd ed.), Chapman & Hall/CRC, Boca Raton (Florida), 2006.

[18] NTT Corporation. PSEC-KEM Specification (v. 2.2), 2008.

[19] V. Shoup. A Proposal for an ISO Standard for Public Key Encryption (v. 2.1), preprint, 2001.

[20] R. Cramer and V. Shoup. Design and Analysis of Practical Public-key Encryption Schemes Secure Against Adaptive Chosen Ciphertext Attack, SIAM Journal on Computing, 33 (1), pp. 167–226, 2003.

[21] ANSI X9.63. Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography, American National Standards Institute, 2001.

[22] IEEE Std 1363a. Standard Specifications for Public Key Cryptography - Amendment 1: Additional Techniques, Institute of Electrical and Electronics Engineers, 2004.

[23] ISO/IEC 18033-2. Information Technology – Security Techniques – Encryption Algorithms – Part 2: Asymmetric Ciphers, International Organization for Standardization, 2006.

[24] SECG SEC 1. Elliptic Curve Cryptography (v. 2.0), Standards for Efficient Cryptography Group, 2009.

[25] D. Eastlake. Additional XML Security Uniform Resource Identifiers (URIs), IETF RFC 4051, 2005.

[26] NESSIE Consortium. Portfolio of Recommended Cryptographic Primitives (v. 1.0), 2003.

[27] NESSIE Consortium. Final Report of European Project Number IST-1999-12324 Named New European Schemes for Signatures, Integrity, and Encryption (v. 0.15), 2004.

[28] M. Bellare and P. Rogaway. Minimizing the Use of Random Oracles in Authenticated Encryption Schemes, Lecture Notes in Computer Science, 1334, pp. 1–16, 1997.

[29] M. Abdalla, M. Bellare, P. Rogaway. DHAES: An Encryption Scheme Based on the Diffie-Hellman Problem, contribution to IEEE P1363a, 1998.

[30] M. Abdalla, M. Bellare, P. Rogaway. The Oracle Diffie-Hellman Assumptions and an Analysis of DHIES, Lecture Notes in Computer Science, 2020, pp. 143–158, 2001.

[31] IEEE Std 1363. Standard Specifications for Public Key Cryptography, Institute of Electrical and Electronics Engineers, 2000.

[32] SECG GEC 2. Test Vectors for SEC 1 (v. 0.3), Standards for Efficient Cryptography Group, 1999.

[33] NIST SP 800-56A. Recommendation for Pair-wise Key Establishment Schemes Using Discrete Logarithm Cryptography, National Institute of Standards and Technology, 2007.

[34] NIST FIPS 180-3. Secure Hash Standard, National Institute of Standards and Technology, 2008.

[35] H. Dobbertin, A. Boselaers, B. Preneel. RIPEMD-160: A Strengthed Version of RIPEMD, Lecture Notes in Computer Science, 1039, pp. 71–82, 1996.

[36] ISO/IEC 10118-3. Information Technology – Security Techniques – Hash-functions – Part 3: Dedicated Hash-functions, International Organization for Standardization, 2004.

[37] ANSI X9.52. Triple Data Encryption: Modes of Operation, American National Standards Institute, 1998.

[38] M. Matsui. Specification of MISTY1 - A 64-bit block cipher, preprint, 2000.

[39] C. Adams. The CAST-128 Encryption Algorithm, IETF RFC 2144, 1997.

[40] K. Aoki et al. Camellia: A 128-bit Block Cipher Suitable for Multiple Platforms - Design and analysis, Lecture Notes in Computer Science, 2012, pp. 39-56, 2001.

[41] H. J. Lee et al. The SEED Encryption Algorithm, IETF RFC 4269, 2005.

[42] H. Krawczyk, M. Bellare, R. Canetti. HMAC: Keyed Hashing for Message Authentication, IETF RFC 2104, 1997.

[43] NIST FIPS 198-1.The Keyed-Hash Message Authentication Code (HMAC), National Institute of Standards and Technology, 2008.

[44] NIST SP 800-38B. Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, National Institute of Standards and Technology, 2005.

[45] Z. Jia and Y. Zhang. An Elliptic Curve Based User Authentication Scheme with Smart Cards, Journal of Information Assurance and Security, 1 (4), pp. 283–292, 2006.

Author Biographies

Víctor Gayoso Martínez obtained his Ph.D. in Telecommunication Engineering from the Polytechnic University of Madrid in 2010. Since 1998, he has been working in topics related to smart cards, Java technology, and public key cryptography.

Fernando Hernández Álvarez obtained his Master Degrees in Telecommunication Engineering from the Polytechnic University of Madrid in 2009 and in Electrical Engineering from the Royal Institute of Technology of Stockholm in 2009. Since then, he has been working in topics related to Biometrics, the Java language, and public key cryptography.

Luis Hernández Encinas obtained his Ph.D. in Mathematics from the University of Salamanca, in 1992. He is a researcher at the Department of Information Processing and Coding, Spanish Council for Scientific Research (CSIC). His current research interests include cryptography, algebraic curve cryptosystems, image processing, and number theory.

Carmen Sánchez Ávila received the Ph.D. in Mathematical Sciences from the Polytechnic University of Madrid in 1993. At present she is Professor in the Department of Applied Mathematics, where during the last years she has been teaching different undergraduate courses as well as graduate courses in Biometric and Cryptography.