A Survey of the Elliptic Curve Integrated Encryption Scheme

Article · January 2010

CITATIONS
52

READS
14,196

3 authors, including:

Víctor Gayoso Martínez
Spanish National Research Council
53 PUBLICATIONS **177** CITATIONS

SEE PROFILE

Carmen Sánchez Ávila
Universidad Politécnica de Madrid
109 PUBLICATIONS **1,904** CITATIONS

SEE PROFILE

A Survey of the Elliptic Curve Integrated Encryption Scheme

V. Gayoso Martínez, L. Hernández Encinas, and C. Sánchez Ávila

Abstract— Elliptic Curve Cryptography (ECC) is a relatively recent branch of cryptography based on the arithmetic of elliptic curves and the Elliptic Curve Discrete Logarithm Problem (ECDLP). Elliptic curve cryptographic schemes are public-key mechanisms that provide encryption, digital signature and key exchange capabilities. The best known encryption scheme based on ECC is the Elliptic Curve Integrated Encryption Scheme (ECIES), included in the ANSI X9.63, ISO/IEC 18033-2, IEEE 1363a, and SECG SEC 1 standards. In the present work, we offer a comprehensive introduction to ECIES, detailing the encryption and decryption procedures and the list of functions and special characteristics included in aforementioned standards.

Index Terms— Elliptic Curve Cryptography, ECIES, encryption scheme.

.

1 Introduction

THE development of public-key cryptography by Whitfield Diffie and Martin Hellman in 1976 [1] represented a revolution in the cryptographic world, overcoming some of the limitations inherent to symmetric-key algorithms such as the key distribution problem.

Public-key schemes are complex designs that, in order to be useful, must be secure and efficient. In general, both characteristics depend on the mathematical problem on which they are based. Some examples of those problems are the integer factorization problem (IFP) used in the RSA cryptosystem [2], the discrete logarithm problem (DLP) used in the ElGamal scheme [3], and the elliptic curve discrete logarithm problem (ECDLP).

In 1985, Victor Miller [4] and Neal Koblitz [5] independently proposed a cryptosystem based on elliptic curves, whose security relies on the ECDLP problem. Elliptic Curve Cryptography (ECC) can be applied to data encryption and decryption, digital signatures, and key exchange procedures.

As in the case of the IFP and DLP, no algorithm is known that solves the ECDLP in an efficient way. Moreover, the ECDLP is regarded as the hardest of these three problems ([5] and [6]). From this fact derives one of the most important benefits of ECC: the key size. Keys in ECC are significantly shorter than in other cryptosystems such as RSA. A shorter key implies easier data management, lower hardware requirements (in terms of buffers, memory, data storage, etc.), less bandwidth when transmitting the keys over a network, and longer battery life in devices where it is important, such as mobile phones.

A comparison between RSA and ECC key lengths is shown in Table 1 and illustrated in Fig. 1, with data taken from [7] and [8], where the security level is interpreted as the cryptographic strength provided by a symmetric encryption algorithm using a key of n bits.

TABLE 1
KEY LENGTH COMPARISON OF RSA AND ECC

Security level (bits)	*RSA key length (bits)*	*ECC key length (bits)*	*Approx. ratio*
80	*1024*	*160-223*	*5-6:1*
112	*2048*	*224-255*	*8-9:1*
128	*3072*	*256-283*	*11-12:1*
192	*7680*	*384-511*	*15-20:1*
256	*15360*	*512-571*	*27-30:1*

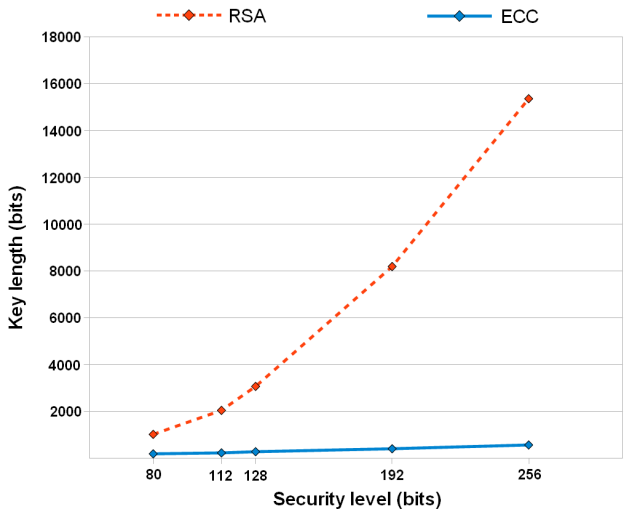

Fig. 1. Key length comparison for RSA and ECC cryptosystems.

- *V. Gayoso Martínez is with the Applied Physics Institute, Spanish National Research Council (CSIC), Madrid, Spain.*
- *L. Hernández Encinas is with the Applied Physics Institute, Spanish National Research Council (CSIC), Madrid, Spain*
- *C. Sánchez Ávila is with the Applied Mathematics to Information Technologies Department, Polytechnic University, Madrid, Spain.*

In the present work, we provide a comprehensive introduction to the ECIES encryption scheme, detailing the encryption and decryption procedures and the list of functions and special characteristics included in the ANSI X9.63, IEEE 1363a, ISO/IEC 18033-2, and SECG SEC 1 standards.

This paper is organized as follows: Sections 2 presents a brief introduction to elliptic curves and ECC. Section 3 enumerates the most important ECC implementations for key exchange, digital signatures and encryption applications. Section 4 describes in detail the ECIES scheme and the encryption and decryption steps performed during its operation. In Section 5 we offer a comparison of the ECIES allowed functions contained in the aforementioned standards. Finally, Section 6 provides a description of some of the additional options that must be taken into consideration not only when developing an ECIES implementation, but also when using this encryption scheme as a final user.

An earlier version of this work appeared in [9], where the comparison of the ECIES standads was included for the first time by the authors. The present contribution offers, in addition to what was presented in [9], an extended introduction to ECC, the fully detailed encryption and decryption processes, and the section dealing with the additional options that must taken into account when configuring ECIES.

2 Elliptic Curve Cryptography

An elliptic curve E over the finite field (or Galois Field) GF is defined by the following equation, known as the Weierstrass equation for elliptic curves in non-homogeneous form [7]:

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

where $a_1, a_2, a_3, a_4, a_6 \in GF$ and $\Delta \neq 0$, being Δ the discriminant of E calculated in the following way [10]:

$$\Delta = -d_2^2 d_8 - 8d_4^3 - 27d_6^2 + 9d_2 d_4 d_6\,,$$

being $d_2 = a_1^2 + 4a_2$, $d_4 = 2a_4 + a_1a_3$, $d_6 = a_3^2 + 4a_6$, and finally $d_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2$.

Condition $\Delta \neq 0$ assures that the curve is non-singular, and thus there are no curve points with two or more different tangent lines.

The homogeneous form of the Weierstrass equation is

$$Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3\,,$$

and this implies the existence of a special point which can only be interpreted in the projective plane: the point at infinity O. This point is paramount in the usage of elliptic curves in cryptography, as it is the identity element that, together with the rest of the points of the elliptic curve and the addition operator (which allows to add two points of the elliptic curve, P and Q, in order to generate another point, $R=P+Q$), characterizes the elliptic curve with the mathematical structure of an abelian group.

When the same point is added several times to itself in the abelian group defined by an elliptic curve, the addition operator is transformed into the scalar multiplication, which in practice allows to multiply an elliptic curve point P by a positive integer n in order to produce another elliptic curve point, $S=n\cdot P$.

The number of points of an elliptic curve (concept also known as the cardinal or the order of the curve) is represented as $\#E$. In contrast, the order of a point P that belongs to an elliptic curve E is the smaller integer n that produces the result $n\cdot P=O$.

From a cryptographic point of view, not every elliptic curve is useful. Cryptographers are interested in elliptic curves that form cyclic abelian groups, and also in elliptic curves with cyclic subgroups, so that the cofactor is a small number (e.g. 2, 4, etc.). As a consequence of Lagrange's theorem (which states that for any finite group M, the order of every subgroup N of M divides the order of M), the order of the generator (i.e. the elliptic curve point that generates all the points of the cyclic subgroup) always divides the order of the elliptic curve (which not necessarily is a prime number).

Two types of finite fields $GF(q)$, with $q = p^m$ elements, are used in ECC: prime finite fields $GF(p)$ (where p is an odd prime and $m = 1$) and binary finite fields $GF(2^m)$ (where $p = 2$ and m can be any integer greater than 1). When working with finite fields, using the proper change of variables it is possible to simplify the Weierstrass equation, obtaining new equations less general (they are adapted to specific finite fields) but easier to manage.

If the characteristic of the finite field is 2, then $GF(q)=GF(2^m)$. If $a_1 \neq 0$, the equation (1) can be reduced to the form

$$y^2 + xy = x^3 + ax^2 + b\,, \qquad (2)$$

where the discriminant is $\Delta = b$.

If $a_1 = 0$, then the equation (1) is transformed into

$$y^2 + cy = x^3 + ax + b\,, \qquad (3)$$

where the discriminant is $\Delta = c^4$.

Moreover, if the characteristic of the finite field is 3, then two cases appear. If $a_1^2 \neq -a_2$, the equation (1) is reduced to

$$y^2 = x^3 + ax^2 + b\,, \qquad (4)$$

where the discriminant is $\Delta = -a^3b$.

In contrast, if $a_1^2 = -a_2$, then equation (1) is reduced to

$$y^2 = x^3 + ax + b\,, \qquad (5)$$

where the discriminant is $\Delta = -a^3$.

Finally, if the characteristic of $GF(q)$ is neither 2 nor 3, using the proper change of variables the equation (1) can be transformed into

$$y^2 = x^3 + ax + b\,, \qquad (6)$$

where the discriminant is $\Delta = -16\left(4a^3 + 27b^2\right)$.

The set of parameters to be used in any ECC implementation depends on the underlying finite field. When

the field is $GF(p)$, the set of parameters that define the curve is (p,a,b,G,n,h), whereas if the finite field is $GF(2^m)$, the set of parameters is $(m,f(x),a,b,G,n,h)$. The meaning of each element in both sets is the following:

- p is the prime number that characterizes the finite field $GF(p)$.
- m is the integer number specifying the finite field $GF(2^m)$.
- $f(x)$ is the irreducible polynomial of grade m defining $GF(2^m)$.
- a and b are the elements of the finite field $GF(q)$ taking part in the equations (2), (3), (4), (5), and (6).
- $G=(Gx,Gy)$ is the point of the curve that will be used as a generator of the points representing public keys.
- n is the prime number whose value represents the order of the point G (i.e. $n \cdot G = O$).
- h is the cofactor of the curve, computed as $h=\#E/n$, where n is the order of the generator G.

3 ECC Standards

Theoretical findings related to either RSA or ECC cannot be used directly, as it is necessary to define data structures and procedures to manage the information. Currently there are three immediate applications for ECC in cryptography, as it is described in this section.

3.1 Elliptic Curve Diffie-Hellman

The main objective of key exchange protocols is to put in contact two or more entities communicating through an open and insecure channel, sharing a secret key that will provide data confidentiality and integrity to any information exchanged using that channel.

ECDH denotes the generic key exchange scheme based on the Diffie-Hellman mechanism applied to elliptic curves. Some practical implementations can be found in ANSI X9.63 [11], IEEE 1363 [12], NIST SP 800-56A [13], and SEC 1 [14] documents.

3.2 Elliptic Curve Digital Signature Algorithm

FIPS 186-2 [15] describes all the algorithms and digital signature schemes that can be used by any agency of the U.S. government. Currently those algorithms are DSA, RSA and ECDSA. ECDSA is the elliptic curve variant of the Digital Signature Algorithm (DSA).

Both FIPS 186-2 [15] and ANSI X9.62 [16] state a minimum key size of 1024 bits for RSA and DSA and 160 bits for ECC, which provides an equivalent security to a symmetric block cipher with a key size of 80 bits (see Table 1).

As a comparison, texts signed with a 1024 bits RSA key produce a digital signature of 128 bytes, whilst the same text signed with a 192 bits ECDSA key generates a digital signature of 48 bytes.

3.3 Elliptic Curve Integrated Encryption Scheme

The most extended encryption and decryption scheme based on ECC is the Elliptic Curve Integrated Encryption Scheme (ECIES). This scheme is a variant of the ElGamal scheme proposed by Abdalla, Bellare, and Rogaway in [17] and [18].

Slightly different versions of ECIES can be found at ANSI X9.63 [11], IEEE 1363a [19], ISO/IEC 18033-2 [20] and SEC 1 [14] standards.

As an example, any standard symmetric key encrypted with a 1024 bits RSA key produces an output of 128 bytes compared with the output of 84 bytes if the encryption is performed with one of the possible configurations of ECIES.

4 ECIES

As its name properly indicates, ECIES is an integrated encryption scheme which uses the following functions:

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

In order to describe the steps that must be taken in order to encrypt a clear message, we will follow the tradition and will assume that Alice wants to send a message to Bob. In that scenario, Alice's ephemeral private and public keys will be represented as u and U, respectively. Similarly, we will refer to Bob's private and public keys as v and V, respectively.

In ECC, private keys are elements of the finite field, either $GF(p)$ or $GF(2^m)$, whilst public keys are points belonging to the elliptic curve and calculated as the product of the private key and the generator G of the elliptic curve. The steps (shown in Fig. 2) that Alice must complete are the following:

1) Alice must create an ephemeral key pair consisting in the finite field element u and the elliptic curve point $U=u \cdot G$. That key pair should be generated pseudo-randomly exclusively for the current process.
2) After the ephemeral keys u and U are generated, Alice will use the Key Agreement function, KA, in order to create a shared secret value, which is the result of the escalar multiplication $u \cdot V$, considering as input values Alice's ephemeral private key u and Bob's public key V.
3) Then, Alice must take the shared secret value $u \cdot V$ and optionally other parameters (e.g. the binary representation of the ephemeral public key U) as input data for the Key Derivation Function, KDF. The output of this function is the concatenation of the symmetric encryption key, k_{ENC}, and the MAC key, k_{MAC}.
4) With the element k_{ENC} and the clear message, m, Alice will use the symmetric encryption algorithm, ENC, in order to produce the encrypted message, c.
5) Taking the encrypted message c, k_{MAC} and optionally other parameters, such as a text string previously agreed by both parties, Alice must use the selected MAC function in order to produce a *tag*.
6) Finally, Alice will take the temporary public key U, the

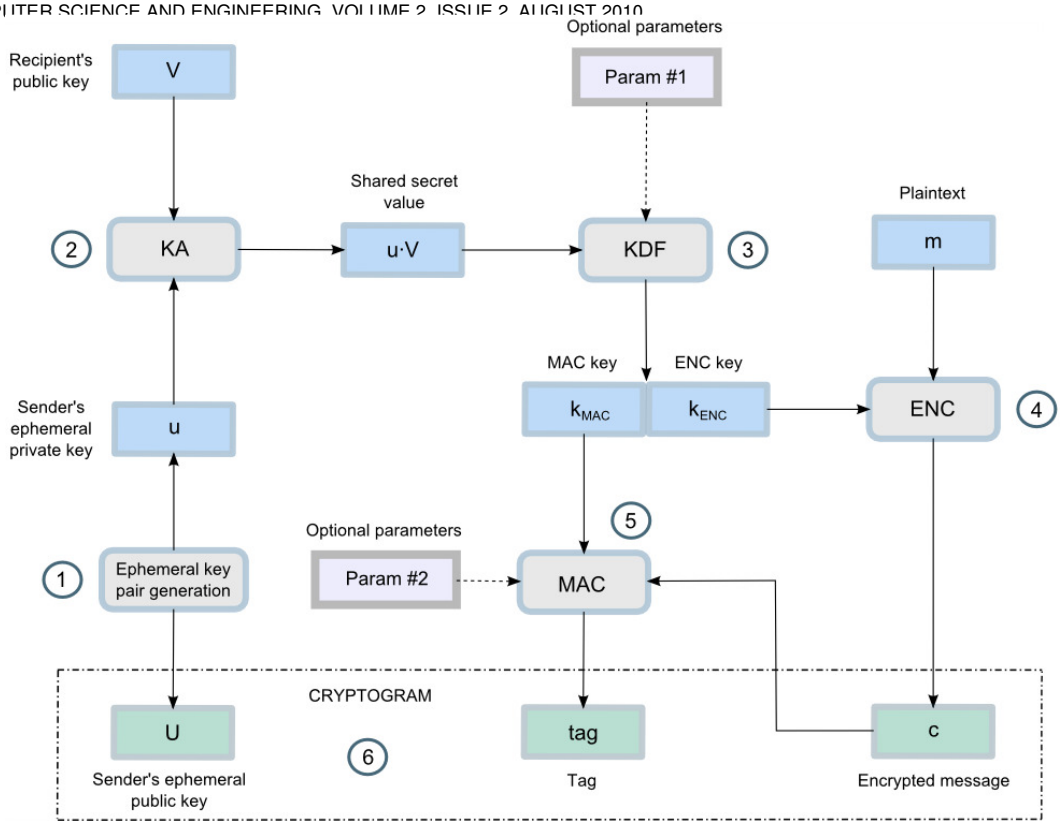

Fig. 2. ECIES encryption functional diagram.

tag, and the encrypted message c, and will send the cryptogram $(U||tag||c)$ consisting of those three concatenated elements to Bob.

Regarding the decryption process, the steps that Bob must perform (shown in Fig. 3) are the following:

1) After receiving the cryptogram $(U||tag||c)$ from Alice, Bob must retrieve the ephemeral public key U, the *tag*, and the encrypted message c, so he can deal with those elements separately.
2) Using the retrieved ephemeral public key, U, and his own private key, v, Bob will multiply both elements in order to produce the shared secret value $v \cdot U$, as the result of this computation is the same that the product $u \cdot V$, which is the core of the Diffie-Hellman procedure ([1] and [7]).
3) Taking as input the shared secret value $v \cdot U$ and the same optional parameters that Alice used, Bob must produce the same encryption and MAC keys by means of the KDF procedure.
4) With the MAC key k_{MAC}, the encrypted message c, and the same optional parameters used by Alice, Bob will first compute the element *tag**, and then he will compare its value with the *tag* that he received as part of the cryptogram. If the values are different, Bob must reject the cryptogram due to a failure in MAC verification procedure.
5) If the tag value generated by Bob is the correct one, then he will continue the process by deciphering the encrypted message c using the symmetric ENC algorithm and k_{ENC}. At the end of the decryption process, Bob will be able to access the plaintext that Alice intended to send him.

5 ECIES Allowed Functions Comparison

This section presents the comparison of allowed KA, KDF, HASH, ENC, and MAC functions that appear in the ANSI X9.63 [11], IEEE 1363a [17], ISO/IEC 18033-2 [20], and SECG SEC 1 [14] standards.

Table 2 shows the different KA functions allowed in ECIES. In the context of ECIES, DH denotes the Diffie-Hellman key agreement function [1] whose standard procedure was described in Section 4, whilst the term DHC refers to the Diffie-Hellman variant that, in addition to the sender's and recipient's keys, includes the cofactor in the computation of the shared secret value by means of the products $h \cdot u \cdot V$ and $h \cdot v \cdot U$ [7].

TABLE 2
ECIES KA Functions per Standard

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

The KDF functions considered in ECIES are presented in Table 3, 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 concatenation function specified in NIST SP 800-56A [13].

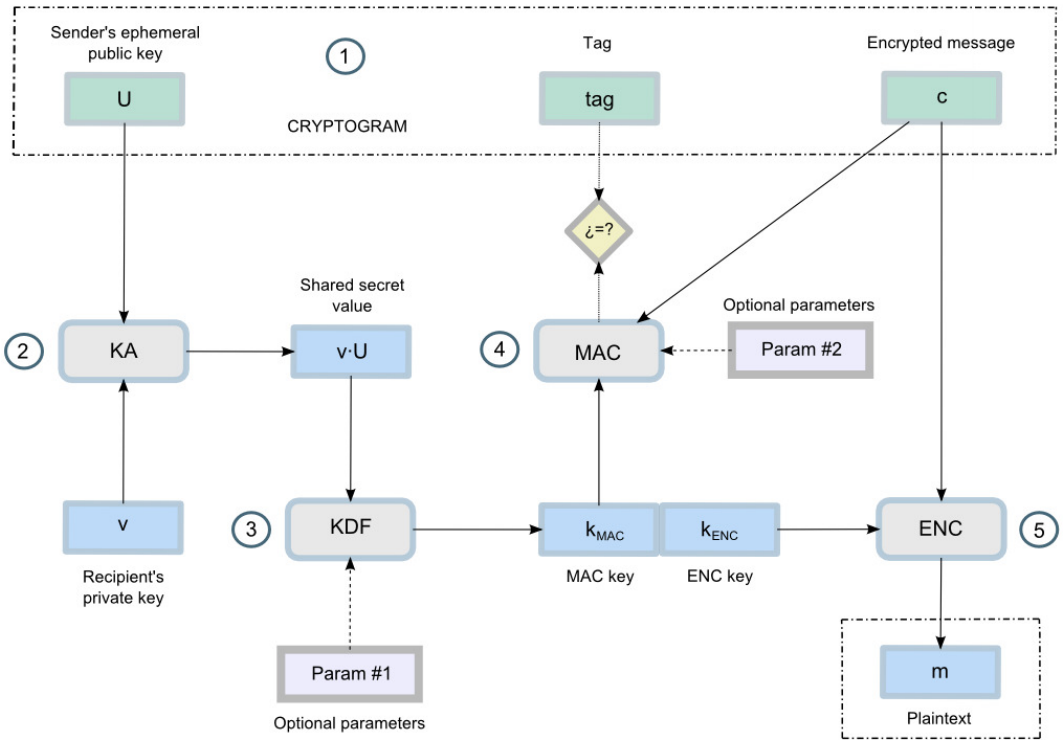

Fig. 3. ECIES decryption functional diagram.

TABLE 3
ECIES KDF FUNCTIONS PER STANDARD

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

In Table 4, the HASH functions used in ECIES are presented. SHA-1 is the well-known digest function included in [21]; SHA-2 represents the family composed by SHA-256, SHA-384, and SHA-512 [21]; SHA-2* is the SHA-2 family with the addition of the SHA-224 hash algorithm [21]; RIPEMD is the set of hash algorithms defined in [22]; and WHIRLPOOL is the function defined in [23].

TABLE 4
ECIES HASH FUNCTIONS PER STANDARD

ANSI X9.63	IEEE 1363a	ISO 18033-2	SECG SEC 1
SHA-1	SHA-1	SHA-1	SHA-1
	SHA-2	SHA-2	SHA-2*
	RIPEMD	RIPEMD	
		WHIRLPOOL	

The symmetric ciphers considered in ECIES are shown in Table 5, where TDES is the Triple DES algorithm in CBC mode [24]; AES represents the Advanced Encryption Standard family, i.e., AES-128, AES-192, and AES-256 [25]; and MISTY1, CAST-128, Camellia, and SEED are the algorithms specified in [26], [27], [28], and [29], respectively.

TABLE 5
ECIES ENC FUNCTIONS PER STANDARD

ANSI X9.63	IEEE 1363a	ISO 18033-2	SECG SEC 1
XOR	TDES	TDES	XOR
	AES	AES	AES
		MISTY1	
		CAST-128	
		Camellia	
		SEED	

In Table 6, the allowed MAC functions are shown. DEA is the MAC function specified in ANSI X9.19 [30]; X9.71 is the reference to another MAC standard developed by ANSI [31]; MAC1, HMAC-SHA-1, and HMAC-RIPEMD are defined in [32]; HMAC-SHA-2 represents the family of HMAC algorithms, i.e., HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512, described in [33]; HMAC-SHA-2* is the same as HMAC-SHA-2 with the addition of the HMAC-SHA-224 function; and CMAC-AES is the set of HMAC functions related to the AES symmetric algorithm, that is, CMAC-AES-128, CMAC-AES-192, and CMAC-AES-256, included in [34].

TABLE 6
ECIES MAC FUNCTIONS PER STANDARD

ANSI X9.63	IEEE 1363a	ISO 18033-2	SECG SEC 1
DEA	MAC1	H-SHA-1	H-SHA-1
ANSI X9.71		H-SHA-2	H-SHA-2*
		H-RIPEMD	CMAC-AES

6 ECIES ADDITIONAL OPTIONS

Due to the significant number of functions implied in the operation of ECIES, there are several options that must be fixed in order to allow the recipient to correctly interpret the cryptogram and successfully decrypt it.

In this section we present the most interesting additional options that must be taken into account by both ECIES developers and users.

6.1 Point compression usage

When converting an elliptic curve point into a binary string, sender and recipient must agree on one of the following two formats:

- *Uncompressed*: Both coordinates are taken into account. A header byte `0x04` indicates that this is the format in use, so the byte string corresponding to the elliptic curve point $P=(Px,Py)$ would be `0x04`$||\ Px\ ||\ Py$, where Px and Py are the binary representations of the coordinates (considered as integer numbers), and $||$ is the concatenation operator.
- *Compressed*: Only the first coordinate is used, which is signalled by using the header byte `0x02` or `0x03`. The proper value of the header is decided based on some computations performed involving both coordinates, so for any elliptic curve point only one compressed binary representation, either `0x02`$||Px$ or `0x03`$||Px$, is valid.

6.2 Shared secret value generation

Independently of which of the KA functions is used (DH or DHC), users face a variety of options regarding the information that will be taken as input in the KDF function:

- Firstly, users must decide whether to use the whole point $P=(Px,Py)$, obtained as the output of the KA function, or just the first coordinate of that point, Px.
- Secondly, they must decide whether to use the element selected given the previous decision, or the hash output of that element, as it is described in [19].

6.3 Keying material interpretation

Before obtaining the MAC and ENC keys from the output of the KDF function, users must define the interpretation order of that output. The two options available are:

- First, the MAC key; then, the ENC key ($k_{MAC}||k_{ENC}$).
- First, the ENC key; then, the MAC key ($k_{ENC}||k_{MAC}$).

CONCLUSIONS

ECIES is the best known encryption scheme in the scope of ECC, which is one of the most interesting current cryptographic trends. Even though ECIES provides some valuable advantages over other cryptosystems as RSA, the number of slightly different versions of ECIES included in the standards may obstruct the adoption of ECIES.

After analyzing the ECIES descriptions contained in ANSI X9.63, IEEE 1363a, ISO/IEC 18033-2, and SECG SEC 1, it can be stated that it is not possible to implement a software version compatible with all those standards, regarding both the specific operations and the list of allowed functions and algorithms. In addition to this, implementations may face another important problem, which is the limitation in the functions available to the developer in the application programming interface of the target device (PCs, smart cards, mobile phones, etc.).

Taking into account both the interoperability and security aspects, even though the newer versions (ISO/IEC 18033-2 and SECG 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.), and include recommendations to avoid the latest criptographic attacks, so those standards should be considered as the starting point for any ECIES implementation.

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.E. Hellman, "New directions in cryptography", *IEEE Transactions in Information Theory*, vol. 22, pp. 644-654, 1976.

[2] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", *Communications of the ACM*, vol. 26, pp. 96-99, 1983.

[3] T. ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms", *IEEE Transactions on Information Theory*, vol. 31, pp. 469—472, 1985.

[4] V.S. Miller, "Use of elliptic curves in cryptography", *Lecture Notes in Computer Science*, vol. 218, pp. 417-426, 1986.

[5] N. Koblitz, "Elliptic curve cryptosystems", *Mathematics of Computation*, vol. 48, pp. 203-209, 1987.

[6] Bundesamt für Sicherheit in der Informationstechnik (BSI), *Elliptic Curve Cryptography*, TR 03111, 2009. http://www.bsi.de/literat/tr/tr03111/BSI-TR-03111.pdf

[7] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. New York: Springer-Verlag, 2003.

[8] National Institute of Standards and Technology (NIST), *Recommendation for key management – Part 1: General*, SP 800-57, 2007.

[9] V. Gayoso Martínez, L. Hernández Encinas, and C. Sánchez Ávila, "A Comparison of the Standardized Versions of ECIES", *Proceedings of the Sixth International Conference on Information Assurance and Security – IAS 2010*, Atlanta, 2010.

[10] J. Silverman, *The Arithmetic of Elliptic Curves*. New York: Springer-Verlag, 1986.

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

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

[13] National Institute of Standards and Technology (NIST), *Recommendation for Pair-wise Key Establishment Schemes Using Discrete Logarithm Cryptography*, SP 800-56A, 2005.

[14] Standards for Efficient Cryptography Group (SECG), *Elliptic Curve Cryptography*, SEC 1, version 2, 2009. http://www.secg. org/download/aid-780/sec1-v2.pdf

[15] National Institute of Standards and Technology (NIST), *Digital Signature Standard (DSS)*, FIPS 186-2, 2000.

[16] American National Standards Institute (ANSI), *Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, X9.62, 1998.

[17] M. Abdalla, M. Bellare, and P. Rogaway, "DHAES: An encryption scheme based on the Diffie-Hellman problem", submission to IEEE P1363a, 1998. http://grouper.ieee.org/groups/1363/P1363a/contributions/dhaes.pdf

[18] M. Abdalla, M. Bellare, and P. Rogaway, *DHIES: An encryption scheme based on the Diffie-Hellman problem*, unpublished, 2001. http://www.cs.ucdavis.edu/~rogaway/papers/dhies.pdf

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

[20] International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), *Information Technology – Security Techniques – Encryption Algorithms – Part 2: Asymmetric Ciphers*, 18033-2, 2006.

[21] National Institute of Standards and Technology (NIST), *Secure Hash Standard*, FIPS 180-2, 2002.

[22] H. Dobbertin, A. Bosselaers, and B. Preneel, "RIPEMD-160: A Strengthened Version of RIPEMD", *Lecture Notes in Computer Science*, vol. 1039, pp. 71-82, 1996.

[23] International Organization for Standardization/International Electrotechnical Commission (ISO/IEC), *Information Technology -- Security Techniques -- Hash-functions -- Part 3: Dedicated Hash-functions*, 10118-3, 2004.

[24] American National Standards Institute (ANSI), *Triple Data Encryption: Modes of Operation*, X9.52, 1998.

[25] National Institute of Standards and Technology (NIST), *Advanced Encryption Standard*, FIPS 197, 2001.

[26] M. Matsui, *Specification of MISTY1 - A 64-bit Block Cipher*, submission to NESSIE, 2000. https://www.cosic.esat.kuleuven.be/nessie/workshop/submissions/misty1.zip

[27] C. Adams, *The CAST-128 Encryption Algorithm*, RFC 2144, 1997. http://www.ietf.org/rfc/rfc2144.txt

[28] K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, and T. Tokita, "Camellia: A 128-Bit Block Cipher Suitable for Multiple Platforms - Design and Analysis", *Lecture Notes in Computer Science*, vol. 2012, pp. 39-56, 2001.

[29] H.J. Lee, S.J. Lee, J.H. Yoon, D.H. Cheon, and J.I.Lee, *The SEED Encryption Algorithm*, RFC 4269, 2005. http://www.ietf.org/rfc/rfc4269.txt

[30] American National Standards Institute (ANSI), *Financial Institution Retail Message Authentication*, X9.19, 1996.

[31] American National Standards Institute (ANSI), *Keyed Hash Message Authentication Code*, X9.71, 2001.

[32] H. Krawczyk, M. Bellare, and R. Canetti, *HMAC: Keyed Hashing for Message Authentication*, RFC 2104, 1997. http://www.ietf.org/rfc/rfc2104.txt

[33] National Institute of Standards and Technology (NIST), *The Keyed-Hash Message Authentication Code (HMAC)*, FIPS 198, 2002.

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

INFORMATION ABOUT AUTHOR(S):

Víctor Gayoso Martínez obtained his Master Degree in Telecommunication Engineering from the Polytechnic University of Madrid in 2002. Since then, he has been working in topics related to smart cards, Java technology 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.