ECIES

Security Evaluation of the Encryption Scheme and Primitives

Evaluator: Prof. Jean-Jacques Quisquater, Math RiZK, consulting

Scientific Support: Dr. François Koeune, K2Crypt

Executive summary.

This document summarizes our security evaluation of ECIES. On the plus side we conclude that no seriously damaging attack has been found in the current state-of-the-art, although we strongly recommend to avoid the use of the XOR as a symmetric encryption primitive, except in very specific scenarios. On the minus side, it appears that the formal proofs of the security of DHIES – from which ECIES is derived – are not easily instantiable to ECIES, as is for example witnessed by an attack, due to Shoup, which contradicts one of the security proofs. Following this attack, a modification of ECIES has been recently proposed by Shoup, but there is no clear consensus yet in the scientific community as to whether this modified scheme should replace ECIES or not. Our advice would therefore be to let some time elapse before adopting ECIES or an updated version of it. This would be especially useful to avoid the risk of coexistence of several similar, but nonetheless incompatible "standards".

Contents

1 Preliminary concepts

1.1 Elliptic curve cryptography

The points of an elliptic curve E over a finite field K form an abelian group. The addition operation of this abelian group involves a pretty small number of arithmetic operations in the underlying field K, and is easy to implement, both in hardware and in software. Moreover, the discrete logarithm problem in this group is believed to be very difficult, in particular, harder than the discrete logarithm problem in finite fields of the same size as K. It was for this reason that elliptic curves were suggested for implementing public key cryptosystems [30].

We refer to [30, 9] for more information on elliptic curve cryptography.

1.2 Diffie-Hellman key agreement protocol

A key component of all ECIES-like schemes is the Diffie-Hellman key agreement primitive. This section will briefly describe this primitive.

Consider a finite cyclic group G, and an element $g \in G$ of prime order. We will use the additive notation for the group operation. To agree on a secret, shared value (denoted as the key), Alice chooses an integer v and computes $V = v.g$ (i.e., g added v times to itself), that she sends to Bob. Bob does the same, choosing an integer u and sending $U = u.g$ to Alice. Alice (resp. Bob) then obtains the shared value by computing $W = v.U$ (resp. $W = u.V$). It is believed[1] that recovering W based on the sole knowledge of U and V is a hard problem, which means that a third party having only access to the exchanged values cannot recover the shared secret.

This key agreement protocol can easily be turned into a public-key encryption scheme, known as the ElGamal scheme. In this scheme, Alice first computes $V = v.g$ and publishes V as her public key. When someone wants to send a message M to Alice, he chooses a random value u and transmits $C = (u.g, M + u.V)$. Knowing v, Alice can compute $v.u.g = u.V$ and recover M by subtracting it from $M + v.U$. Including $u.g$ in the ciphertext provides an "implicit" Diffie-Hellman key exchange.

[1] This assumption will be discussed in more details in section 3.5.

2 System description

ECIES is better viewed as a member of a family of schemes, all of which are based on the same construction. We will start by describing this general construction, and will then describe the primitives that ECIES uses as building blocks.

2.1 General construction

ECIES-like schemes involve a group G, an element $g \in G$ of prime order, a Diffie-Hellman-like key agreement protocol KA, a key derivation function KDF, a message authentication MAC, and a symmetric encryption primitive ENC. We will come back to these components below.

Figure 1: Global view of the encryption operation. The shaded rectangles comprise the ciphertext.

A global view of the scheme is depicted on figure 1. In the setup phase, Alice computes an ElGamal key pair (v, V) and publishes the public key V. To send her an encrypted message, Bob chooses a random u and computes an "ephemeral public key", $u.g$. Note that Alice and Bob (and presumably no one else) will

both be able to compute the "secret value" $u.v.g$. This value is passed to the key derivation function KDF and the result is parsed into two pieces: a MAC key, MacKey, and an encryption key, EncKey. The message to be sent is then symmetrically encrypted with the encryption key, and a MAC of the resulting ciphertext is computed using the MAC key. The ciphertext consists of the ephemeral public key, the symmetrically encrypted plaintext, and the authentication tag generated by the MAC.

2.2 ECIES

ECIES follows the above construction, with the following choices.

Remark: as this report does not have the ambition to serve as a replacement for [15], and in order not to overload this description with cumbersome details (octet to integer conversion, ...), we will focus here on a high-level description, and refer to [15] for technical details.

2.2.1 Group

The group in which the key exchange takes place is the group generated by a point g of prime order n on an elliptic curve over $\mathbb{F}_p$ or $\mathbb{F}_{2^m}$. The exact form of the elliptic curve can be found in [15].

2.2.2 Key agreement protocol

The key agreement protocol is the elliptic curve Diffie-Hellman primitive, which is nothing more than Diffie-Hellman as described in section 1.2, with $< g >$ as underlying group. The shared value is the x-coordinate of the resulting point.

A variant of the Diffie-Hellman primitive, known as the cofactor Diffie-Hellman primitive, is also proposed; this variant, as well as the rationale for it, will be described in section 4.1.

2.2.3 Key derivation function

The key derivation function takes as input a shared (presumably secret) value, and transforms (expands) it into key material of the appropriate size. Its goal is to hide any observable regularity in the representation of group elements.

The only key derivation function supported so far is ANSI-X9.63-KDF. This function basically consists in repeatedly hashing the shared value concatenated

with a counter[2], with the key material consisting in the successive outputs. The only hash function allowed so far is `SHA-1`, but [15] states that support for `SHA`'s successors will be added in the future, as they get standardized.

2.2.4 MAC scheme

A MAC scheme allows two users, supposedly sharing some secret value `MacKey`, to check that a message was transmitted correctly. Before sending the message, the sender uses the secret value to compute a *tag* of the message. When receiving the tag and message, the recipient, knowing the secret, is able to check that the tag corresponds to the message, and therefore that the message has not been tampered with.

Two MAC schemes are currently supported: HMAC-SHA-1-160 and HMAC-1-80, both of which are based on the hash function `SHA-1`. These schemes are specified in IETF RFC 2104 [28] and ANSI X9.71 [6].

2.2.5 Symmetric encryption primitive

Two symmetric encryption schemes are currently supported: 3-key TDES in CBC mode and XOR encryption scheme.

3-key TDES is specified in ANSI X9.52 [5]; here the IV is fixed to the value 0000000000000000_{16}. The XOR encryption scheme is the simple encryption scheme in which encryption consists of XORing the key and the message, and decryption consists of XORing the key and the ciphertext to recover the message. The key material has to be of the same length as the message to be encrypted. Note that we strongly recommend to avoid the use of the XOR encryption scheme, except in very specific contexts. See section 4.3 for more details.

In the future, it is planned that the Advanced Encryption Standard (AES) will be added to this list.

[2] An additional input, `SharedInfo`, consisting in some information shared between the two parties, is also allowed. According to [15], "*Selection of appropriate information to include in SharedInfo will likely depend on the particular application, but common things to include are, for example, the identities of U and V, the public keys QU and QV , counter values, and an indication of the symmetric scheme for which the agreed keying data will be used.*". We will not extend further on SharedInfo parameters.

2.2.6 Parameters choice

The type (elliptic curve definition, underlying field $\mathbb{F}_{2^m}$ or $\mathbb{F}_q$,...) and size of parameters seem appropriate in view of the current state-of-the-art.

Regarding parameters type, it is interesting to note that SEC1's conservative policy has proven useful. As a matter of fact, in a review of a preliminary version of SEC1 (namely, version 0.4), Boneh [10] considered that "*the standard is conservative in its choice of fields $\mathbb{F}_{2^m}$. The list of nine fields all use a prime value for m. Currently, we do not know of a reason why $\mathbb{F}_{2^m}$ with m composite should be avoided. However, it is quite possible that the ECC discrete log on $\mathbb{F}_{2^m}$ with a composite m is only as hard as the ECC discrete log problem on the largest subfield of $\mathbb{F}_{2^m}$. [...] This is a sound design principle*". Since this paper was published, new attacks have been discovered [19, 20, 17, 38] that actually exploit the composite character of m to obtain a subexponential solution of the elliptic curve discrete log problem (note that these attacks are mentioned in the current version of SEC1).

SEC1 proposes various sizes for the underlying field, depending on the required security level and available computing power. Boneh [10] estimates that "*the smallest allowable [...] field size provides a level of security comparable to a 56-bit symmetric key, while the largest field size provides a level of security comparable to a 256-bit symmetric key. The lowest level of security is adequate when export restrictions are in place. The highest level of security is appropriate for use with the upcoming AES encryption standard*".

3 Security proofs

Important remark: the security proofs discussed in this section were in fact presented in the context of the DHIES scheme rather than ECIES itself. Although they were presented by ECIES's authors as a quality witness of ECIES as well (they were for example joined to the submission to Nessie), instantiating them to ECIES is far from trivial, as will for example be illustrated by Shoup's attack.

3.1 What is a security proof?

The security of cryptographic schemes is often witnessed by the *absence* of attacks: a scheme is considered to be secure if it has been around for a long period, has received widespread attention, and could nonetheless not be signif-

icantly weakened by any attack. Famous examples of this "negative" security argument are the Data Encryption Standard (DES) and RSA.

As opposed to this, provable security is often considered as a decisive advantage for a cryptographic scheme. In this context, it is *proved* (in a precise mathematical sense) that the scheme is secure, provided some assumptions are satisfied. These "basic" assumptions are typically the security of the primitives the scheme is based on (e.g. the Diffie-Hellman key agreement), or the difficulty of some well-known problem (e.g. factorization, discrete log computation, ...).

Typically, security is proved by exhibiting a reduction, that turns a breaking of the scheme into a solution of the underlying problem. By contraposition, this proves that, if the underlying problem is actually hard, then the scheme is secure.

3.2 Random oracle model and standard model

Since security proofs are pretty difficult to write, and the underlying schemes are often inefficient, an alternate proof model, called the *random oracle model* has been proposed ([8]). In this model, some functions are assumed to be perfectly random: they are modelled, in the security proof, by an oracle which answers every new request by a random value, chosen independently of previous answers. Therefore, the only way to know the value the function will take on some given input is to query the oracle, and no attack may exploit a "structure" in the function's behaviour.

Unfortunately, when moving to a practical scheme, these presumably perfect functions have to be replaced by actual functions, with the hope that these functions will not present any more structure than their ideal corresponding. Of course the theoretical proofs become mathematically unsound in these practical applications, but security proofs in the random oracle model are nonetheless usually considered as a good witness of security.

However, the assumptions underlying the random oracle model constitute a serious restriction on the generality of the proofs, and, when possible, proofs that get rid of this model are therefore highly desirable (see for example [14] for a more comprehensive discussion on the concerns raised about the random oracle model).

The security proofs concerning DHIES hold in the standard model, i.e. do not rely on random oracles.

3.3 Resistance level

An important point to settle in a security proof is a precise definition of which operations the adversary is allowed to perform, and which information he has access to. Of course, the more power we give to the adversary, the better the quality of the resisting scheme. In the same way, it is also necessary to precisely define what we will consider as a successful attack.

The most commonly used classification of adversaries and adversary's goals is summarized to [7]. As far as adversary's goals are concerned, the authors consider two different goals, indistinguishability (IND) and non-malleability (NM). "*Indistinguishability formalizes an adversary's inability to learn any information about the plaintext x underlying a challenge ciphertext y, capturing a strong notion of privacy. Non-malleability formalizes an adversary's inability, given a challenge ciphertext y, to output a different ciphertext y' such that the plaintexts x, x' underlying these two ciphertexts are "meaningfully related". (For example, $x' = x + 1$.) It captures a sense in which ciphertexts can be tamper-proof.*"

Regarding the adversary's power, three different attacks are considered: "*In order of increasing strength these are chosen-plaintext attack (CPA), non-adaptive chosen-ciphertext attack (CCA1), and adaptive chosen-ciphertext attack (CCA2). Under CPA the adversary can obtain ciphertexts of plaintexts of her choice. In the public-key setting, giving the adversary the public key suffices to capture these attacks. Under CCA1, [...] the adversary gets, in addition to the public key, access to an oracle for the decryption function. The adversary may use this decryption function only for the period of time preceding her being given the challenge ciphertext y. (The term non-adaptive refers to the fact that queries to the decryption oracle cannot depend on the challenge y. Colloquially this attack has also been called a "lunchtime", "lunch-break", or "midnight" attack.) Under CCA2, [...] the adversary again gets (in addition to the public key) access to an oracle for the decryption function, but this time she may use this decryption function even on ciphertexts chosen after obtaining the challenge ciphertext y, the only restriction being that the adversary may not ask for the decryption of y itself. (The attack is called adaptive because queries to the decryption oracle can depend on the challenge y.)*"

One can "mix-and-match" the goals {IND, NM} and attacks {CPA, CCA1, CCA2} in any combination, giving rise to six notions of security:

IND-CPA, IND-CCA1, IND-CCA2, NM-CPA, NM-CCA1, NM-CCA2.

The implications between these six notions have been proved in [7], and are

Figure 2: Implications and non-implications between security notions. Note that implications may of course be combined.

summarized on figure 2. The notions we will be interested in in this document are mainly IND-CPA and IND-CCA2. As can be seen from the figure, IND-CCA2 is equivalent to NM-CCA2, and can be considered as the strongest of the above security notions, whereas IND-CPA is a much weaker notion. Basically, DHIES's security proofs state that DHIES is secure in the strong sense, provided the underlying primitives are secure in a much weaker (and therefore easier to achieve) sense.

Remark: indistinguishability is often presented as a "find-guess" game. In this game, the adversary is first allowed (with or without access to the decryption oracle, depending on the criterion we are considering) to build two messages x_1, x_2. One of these messages is then encrypted into y and sent back to the adversary, who must discover which message it corresponds to. In the CCA2 scenario, the adversary has also access to the oracle in this phase, but with the restriction that he cannot submit his challenge y.

3.4 Tightness of the reduction

Another non-negligible question in security proofs is the tightness of the reduction. Intuitively, we would like to know how much easier it is to break the scheme than to solve the underlying hard problem. One possible characterization is the following.

We informally define Adv as the advantage the adversary has in achieving his goal, compared to an adversary that would produce his guess randomly[3].

[3]The formal definition of the advantage depends on the underlying problem. For example, in the case of the decisional Diffie-Hellman problem, the advantage is defined as

We also have to quantify the resources the adversary has access to. Typical resources are running time and number of queries to a given oracle. We will adopt the notation

$$\mathsf{Adv}_X^Y(r_1, \ldots, r_n)$$

to denote the maximum advantage of an adversary trying to break scheme X in the Y sense (e.g. IND-CCA2), and allowed r_i accesses to resource i (e.g. running time less than r_1).

By comparing the success probability and amount of resources needed by a breaking of the scheme and that needed by a breaking of the underlying primitive, we can obtain a measurement of the tightness of the reduction.

3.5 Assumptions on underlying schemes

As was said before, a security proof reduces the security of the scheme to the difficulty of solving some well-known problem, or to the security of some underlying primitive. This section will review the basic problems that span DHIES's security[4].

3.5.1 Computational Diffie-Hellman assumption (CDH)

We refer to the "standard" Diffie-Hellman assumption as the computational Diffie-Hellman assumption, CDH. It states that given $u.g$ and $v.g$, where u, v were drawn at random from $\{1, \ldots, |G|\}$ it is hard to compute $u.v.g$. Under the computational Diffie-Hellman assumption it might well be possible for the adversary to compute something interesting about $u.v.g$ given $u.g$ and $v.g$; for example, the adversary might be able to compute the most significant bit, or even half of the bits. This makes the assumption too weak to directly use in typical applications.

the difference between the probability to conclude that (U, V, W) is a Diffie-Hellman triple when it actually is, and the probability to conclude that (U, V, W) is a Diffie-Hellman triple when W was in fact chosen at random. We refer to [4] for an exhaustive list of the formal advantages.

[4] A more complete discussion of these assumptions, as well as additional variants of them, can be found in [4], of which this section is largely inspired; we will here stick to problems of direct interest to the security proof.

3.5.2 Decisional Diffie-Hellman assumption (DDH)

A stronger assumption that has been gaining popularity is the decisional Diffie-Hellman assumption, DDH. It states, roughly, that the distributions $(u.g, v.g, u.v.g)$ and $(u.g, v.g, w.g)$ are computationally indistinguishable when (u, v, w) are drawn at random from $\{1, \ldots, |G|\}$. This assumption can only hold in a group G whose order does not contain small prime factors.

3.5.3 Hash Diffie-Hellman assumption (HDH)

The assumption that will be used to prove security for `DHIES` under chosen-plaintext attack is weaker than DDH but stronger than CDH.

Basically, we need to be able to get some number of "hard-core" bits from the Diffie-Hellman key, namely key derived bits that cannot be distinguished from random bits. Our assumption is that applying a suitable key derivation function `KDF` to $u.v.g$ will yield such bits. More precisely, it states that the distributions $(u.g, v.g, \texttt{KDF}(u.v.g))$ and $(u.g, v.g, W)$ are computationally indistinguishable when (u, v) are drawn at random from $\{1, \ldots, |G|\}$, and W is a random string of the same length as the output of `KDF`.

3.5.4 Oracle Diffie-Hellman assumption (ODH)

Suppose we provide an adversary A with $v.g$ and an oracle $\mathcal{H}_v$, which computes the function $\mathcal{H}_v(X) = v.X$. Think of $v \in \{1, \ldots, |G|\}$ as having been chosen at random. Now if we give the adversary $u.g$ (where $u \in \{1, \ldots, |G|\}$ is chosen at random) then the oracle will certainly enable the adversary to compute $u.v.g$: the adversary need only ask the query $u.g$ and he gets back $\mathcal{H}_v(u.g) = u.v.g$. Even if we forbid the adversary from asking $u.g$, still he can exploit the self-reducibility of the discrete log to find the value of $u.v.g$. For example, the adversary could compute $\mathcal{H}_v(u.g + g) = u.v.g + v.g$ and subtract $\mathcal{H}_v(g) = v.g$ from the result. But what if instead we give the adversary an oracle $\mathcal{H}_v$ which computes $\mathcal{H}_v(X) = \texttt{KDF}(v.X)$, for an appropriate key derivation function (basically, a hash function)? Suppose the adversary's goal is to compute $\texttt{KDF}(u.v.g)$, where $u.g$ and $v.g$ are provided to the adversary. Now, as long as the oracle $\mathcal{H}_v$ can not be queried at $u.g$, the oracle would seem to be useless. This is the idea of the oracle Diffie-Hellman assumption.

More formally, it states that the distributions $(u.g, v.g, \texttt{KDF}(u.v.g))$ and $(u.g, v.g, W)$ are computationally indistinguishable when (u, v) are drawn at random from $\{1, \ldots, |G|\}$, and W is a random string of the same length as the

output of KDF, even if A is given access to the oracle $\mathcal{H}_v$ with the sole restriction that it cannot be queried at $u.g$.

3.5.5 MAC's security assumption

Our security assumption about MAC is that it resists chosen-messages attacks: we consider an adversary who is allowed to submit messages to an oracle, obtaining valid tags for these messages. His goal is to produce a pair (M, τ) such that τ is a valid tag for M and (M, τ) was not previously obtained via a query to the oracle.
Remark: This definition is stronger than the usual one. Usually, one asks that the adversary not be able to produce MACs of new messages. Here we require additionally that the adversary not be able to generate new MACs of old messages. However, if the MAC generation function is deterministic and verification is done by simply re-computing the MAC (this is typically true) then there is no difference.

3.6 DHIES security proofs

This section will discuss the two security proofs provided in [4].

The first result shows that DHIES is resistant against chosen-plaintext attacks.

Theorem 3.1 *Let G be a represented group, let* SYM *be a symmetric encryption scheme, let* MAC *be a message authentication scheme, and let* KDF *be a function. Let* DHIES *be the asymmetric key encryption scheme associated to these primitives. Then, for any numbers t and c,*

$$Adv^{\textit{ind-cpa-fg}}_{\texttt{DHIES}}(t, c) \leq 2Adv^{\textit{hdh}}_{G,\texttt{KDF}}(t) + Adv^{\textit{ind-cpa-fg}}_{\texttt{SYM}}(t, 0, 0).$$

Here t denotes the running time, and c stands for the maximum length of the challenge ciphertext.

Intuitively, the theorem states that, if no adversary with maximum running time t has a significant chance to solve the hash Diffie-Hellman problem, and no adversary with maximum running time t has a significant chance to win a chosen-plaintext find-guess game against SYM, then no adversary with maximum running time t can have a significant chance to win a chosen-plaintext find-guess game against DHIES. Note that this reduction is very tight.

Remark: The notation $\mathrm{Adv}^{\text{ind-cpa-fg}}_{\mathsf{SYM}}(t,0,0)$ deserves some additional comment. The last two zeroes correspond to the maximum number of queries to the decryption oracle, and to the maximum total length of these queries (clearly, as the first is equal to 0, so is the second). Zero allowed queries means that this adversary must be able to break SYM without any access to the oracle (in this sense, the term "chosen-plaintext attack" is a bit misleading in this case). The requirements for a symmetric primitive to be eligible as a component of this scheme are therefore pretty weak.

The above proof's main shortcoming is that it only considers chosen-plaintext adversaries. A security proof against more powerful adversaries would be useful. This is the purpose of the following theorem:

Theorem 3.2 *Let G be a represented group, let* SYM *be a symmetric encryption scheme, let* MAC *be a message authentication scheme, and let* KDF *be a function. Let* DHIES *be the asymmetric encryption scheme associated to these primitives. Then for any numbers t, q, μ, and c,*

$$\begin{aligned}\mathrm{Adv}^{\text{ind-cca-fg}}_{\mathtt{DHIES}}(t,q,\mu,c) \leq \quad & \mathrm{Adv}^{\text{ind-cpa-fg}}_{\mathtt{SYM}}(t,0,0) + 2.\mathrm{Adv}^{\text{odh}}_{G,\mathtt{KDF}}(t,q) \\ & +2.\mathrm{Adv}^{\text{suf-cma}}_{\mathtt{MAC}}(t,1,c,q,\mu).\end{aligned}$$

Intuitively, this theorem means that:

- if SYM is resistant, in the IND sense, to adversaries with maximum running time t and allowed no use of the encryption oracle;
- if KDF is resistant, in the ODH sense, to adversaries with maximum running time t and allowed q queries to the ODH oracle;
- if MAC is resistant to adversaries with maximum running time t, allowed one query to the tag-generation oracle (with maximum length c), and q queries to the tag-verification oracle (with maximum length μ);
- then DHIES is resistant, in the IND sense, to *adaptative* adversaries with maximum running time t, allowed q queries to the decryption oracle (with maximum length μ) and with a challenge ciphertext of length at most c.

Note that this is a pretty strong security reduction, in the sense that, on one hand, it is very tight, and, on the other hand, it proves that DHIES is IND-CCA2, while SYM, for example, is only required to be IND-CPA.

3.7 Limitations of security proofs

Although security proofs constitute an invaluable improvement compared to the heuristic character of the "absence of attacks" criterion, one must guard himself against the risk of considering them as an absolute, definitive guarantee of full-proof security.

In particular, it is fundamental to understand that these mathematical proofs fit in a very specific context, requiring an unambiguous definition of which operations the adversary may perform, what will be considered as a successful attack, etc. This specification of the model involves a large amount of assumptions, many of which are implicit, and the security proof will only make sense provided the reality does correspond to these assumptions. The literature provides a pretty large amount of provably secure systems getting attacked based on an insufficiently comprehensive model or incorrect implicit assumptions. In a very recent paper, to appear at Crypto 2002, Stern et al. [39] present examples of such attacks against the well-known ESIGN and ECDSA (the latter of which is part of SEC1), and point out that "*the fact that proofs also need to be validated through public discussion was somehow overlooked. [...] By flaws, we do not mean plain mathematical errors but rather ambiguities or misconceptions in the security model.*"

A good example of these limitations is given by Shoup's attack. As will be shown in section 4.2, there exists a malleability attack against ECIES. If the above proofs applied to ECIES, this would be in contradiction with the IND-CCA2 proof. It is interesting to look at this apparent contradiction in more detail, as it may help provide a better understanding of the implicit assumptions made in security proofs. In DHIES's security proof, it is implicitly assumed that no two elements of the group have the same representation. This is important, because the KDF function is not a group operation, as it acts on the representation (bit strings) rather than on the elements. Unfortunately, what was true in DHIES is not true anymore for ECIES: due to the encoding chosen (x-coordinate, as discussed in section 2.2.2), several group elements may have the same representation. As a consequence, the oracle Diffie-Hellman assumption does not hold anymore.

3.8 A proof in the generic group model

Besides the random oracle model, another model aimed at simplifying security proofs has been proposed. This model, known as the generic group model, models the group G as ideal, in the sense that there is no observable correlation

between the representations of the different group elements; for example, one cannot deduce the representation of the element $a + b$ from the representations of a and b: this representation must be obtained through a request to an oracle. For more information on the generic group model, see e.g. [21, 12, 13].

The restrictions mentioned about the random oracle model's limitations (namely that proofs vanish when we return to the standard model) apply to the generic group model as well. On the other hand, this model allowed Smart [37] to get rid of the "unnatural" ODH assumption. According to Smart, "*Although the proof of security of* ECIES *is in the standard model of computation, it relies on a non-standard intractability assumption. Namely, an interactive hash-based version of the Decision Diffie-Hellman problem. This has led some authors to criticize the security proof*".

Besides the fact that is does not rely on ODH, Smart's result is very similar to the previous one: he proved that,

Theorem 3.3 *In the generic group model,*

$$\begin{aligned} \mathit{Adv}^{\text{ind-cca-fg}}_{\texttt{DHIES}}(t, v, w, \mu, m) \leq \quad & \mathit{Adv}^{\text{ind-cpa-fg}}_{\texttt{SYM}}(t_1, 0, m, m') \\ & + 2.v.\mathit{Adv}^{\text{suf-cma}}_{\texttt{MAC}}(t_3, v-1) + \tfrac{2(v+w)^2}{q}. \end{aligned}$$

Here t denotes the running time, v the number of queries to the decryption oracle, w the number of queries to the generic group oracle, and μ, m, m' the size of corresponding requests. We refer to [37] for the exact expression of the relationship between t, t_1 and t_3.

Although this proof may appear more general than the one given by theorem 3.2, one must not forget the idealized character introduced by the generic group model. In fact, it is very well possible that the generic group model and the interactive hash-Diffie-Hellman assumption are in fact two similar artifacts hiding the same difficulty. As a matter of fact, they both get stuck on the same problem: the key derivation function.

On one hand, Smart notes that, although KDF can, in the generic group model, be simply defined as the bit representation of the underlying group element, this is certainly not the case when this group is replaced by an actual one, in which case KDF should be instantiated using a strong hash function.

On the other hand, Shoup's attack shows a difficulty directly related to ECIES's encoding function, which is intrinsic part of KDF's definition. As was shown in section 3.7, the disagreement between the security proof and the attack is due to the fact that the ODH assumption does not hold with this instantiation

of KDF. One may therefore consider that the ODH assumption has also hidden the problem.

In our opinion, there is therefore no clear advantage in Smart's proof compared to the one relying on ODH.

3.9 Do the building blocks fit their requirements?

It is not useless to repeat that the aforementioned security proofs reduce the security of DHIES to that of its building blocks. In the current state-of-the-art, it is *believed* that the Diffie-Hellman problem is intractable, that TDES resists chosen-plaintext attacks, that ANSI-X9.63-KDF is hardcore for the Diffie-Hellman problem, ... One must however always keep in mind that none of these facts has been proved, and that there is still the possibility for a solution to one of these problems to appear in the future. If this were the case, the security of the full scheme would of course have to be reconsidered.

3.10 Comparison with other schemes

ECIES was one of the candidates submitted to the Nessie project. The other candidates in the asymmetric encryption category were ACE Encrypt, EPOC-2, PSEC-2, RSA-OAEP, EPOC-1, EPOC-3, PSEC-1 and PSEC-3. The last four candidates were not selected for the second round of Nessie.

The Nessie team compared the security of the candidates, and concluded that ([31] "ECIES *and* PSEC-3 *are the schemes with the most efficient security reduction submitted in this category. Whilst the asymmetric component of* ECIES *is at least as secure as* PSEC-3*, it has stronger requirements for its symmetric components*" .

4 Overview of currently known attacks

4.1 Small subgroup attack

A potential attack against the Diffie-Hellman primitive is known as the *small subgroup attack* [22, 29], in which an adversary substitutes Bob's public key with a point of small order in an attempt to coerce Alice to calculate a predictable field element. The consequences of these attacks can be severe - in a key agreement

scheme, for example, the result can be compromise of a session key, or even compromise of Alices static secret key.

One way to prevent this attack is to check that Bob's public key has order n before accepting to enter the key agreement protocol (this check is performed in the "public key validation" procedure described in [15]).

As an alternative, this attack can also be countered by using a variant of Diffie-Hellman, known as the *cofactor Diffie-Hellman primitive*. In this primitive, Alice and Bob compute and transmit the same values as in classical Diffie-Hellman (i.e. (v, V) for Alice, (u, U) for Bob), but the shared value is computed as $W = h.v.U = h.u.V$, where h is the order of the elliptic curve divided by the order of G. The candidate shared value is rejected if it is equal to the point at infinity $\mathcal{O}$. This multiplication by h has the effect that all calculations with a point in a small subgroup will yield the point at infinity, and hence conduct to rejection of the key.

Since it counters small subgroup attacks, this procedure allows a faster validation of the public key (denoted as "partial validation of the public key" in [15]), which can be useful in some contexts.

4.2 Benign malleability attack

As was noted in section 2.2.2, only the x-coordinate of the elliptic curve point is used as input to the key derivation function. Shoup [36] points out that this allows a malleability attack against ECIES as follows: consider a ciphertext $C = (u.g, c, \mathtt{MAC}_k(c))$. The first element of this triple, the ephemeral key, is an elliptic curve point. So, if we replace $u.g$ by $-u.g$ and apply the Diffie-Hellman protocol, we will obtain $-u.v.g$ rather than $u.v.g$ as shared value. But the addition rule on the elliptic curve (see [15, section 2.2]) is such that the opposite of the point (x, y) is the point $-(x, y) = (x, -y)$. Since only the x coordinate enters the key derivation function, the final shared key will be unchanged.

This shows that, given a valid ciphertext $C = (u.g, c, \mathtt{MAC}_k(c))$, one can easily forge another valid ciphertext $C' = (-u.g, c, \mathtt{MAC}_k(c))$.

A similar attack is possible when the cofactor Diffie-Hellman is used. In this case, one could add to $u.g$ a point whose order divides h. This point will "vanish" in the multiplication by h and we obtain once again a different valid ciphertext.

Of course, this does not represent a very damaging attack (Shoup [36] defines this as *benign malleability*). Nonetheless, it is in contradiction with the definition of non-malleability. To counter the attack, Shoup suggests to make

mandatory the addition of $u.g$ to the input to KDF (which is made possible, but not mandatory, by SEC1).

4.3 Weakness of the XOR encryption

In [36], Shoup also points out a significantly more damaging attack, showing that the scheme is malleable (in a stronger sense than the one mentioned in previous section) when XOR is used as the SYM primitive and messages are of variable length.

Suppose the input to the encryption algorithm is a message M of length l, and a label L.[5] After the key agreement and key derivation functions have been applied, we obtain a string $k||k'$, where k has length l and k' has length MAC.KeyLen (i.e., the key length required by MAC). The ciphertext is

$$C = (u.g, c, \mathtt{MAC}_{k'}(c||L)),$$

where $c = M \oplus k$. Now, consider a message M such that

- $M = M_1||M_2$, where $|M_1| = l'$ and $|M_2| =$ MAC.KeyLen, for some $l' > 0$,
- $c = c_1||c_2$, where $|c_1| = l'$ and $|c_2| =$ MAC.KeyLen
- M_2 is known to the attacker.

Then for any byte string Δ of length l', and any label $\tilde{L}$, the ciphertext

$$\tilde{C} = (u.g, \tilde{c}, \mathtt{MAC}_{\tilde{k}}(\tilde{c}||\tilde{L}),$$

where

$$\tilde{c} = c_1 \oplus \Delta \text{ and } \tilde{k} = c_2 \oplus M_2,$$

is a valid encryption of $M_1 \oplus \Delta$ with label $\tilde{L}$.

Thus, the scheme is trivially malleable, in a very strong way: we can transform the encryption of $M_1||M_2$ with label L into an encryption of $M_1 \oplus \Delta$ with label $\tilde{L}$, for any Δ and any $\tilde{L}$.

This attack relies heavily on the fact that the two messages are not of the same length, and could therefore not be applied if messages were forced to some fixed length. In any other scenario, the use of XOR as a symmetric encryption primitive should be prohibited. Making this restriction explicit in the norm would probably be desirable.

[5] We mention the label L only for the sake of completeness; it can basically be ignored in this discussion. The reader interested in the exact role of this label may for example find a nice description in section 2.1.4 of [36].

4.4 Additional modifications suggestions

Without exhibiting specific attacks, Shoup also issues some additional criticisms regarding various details of ECIES's specification. For example, he strongly recommends that all system parameters, including the choice of KDF, ENC and MAC, be fixed at key generation time, whereas IEEE P1363a[6] allows – although recommends to avoid – these to vary dynamically over the lifetime of the public key. Shoup even states that, in his opinion, "*allowing such flexibility is entirely unacceptable: all hope of a meaningful security analysis vanishes if one allows for this, and there may indeed be real harm that could come from "unintended interactions". This proposal recommends* with the strongest possible urgency *that the ISO standard should require all such options to be fixed at key generation time*".

This recommendation, as well as some others, is however subject to polemic and there is no clear consensus yet on whether they should be accepted or not.

4.5 Importance of careful implementation

A large number of cryptographic attacks target implementation-related issues (running time, power consumption, random generation,...). Being implementation specific, these attacks are outside the scope of SEC1 and this evaluation report. There are however some points that may be mentioned even at this level of generality.

4.5.1 Protection against side-channel attacks

A very powerful class of implementation-related attacks are the side-channel attacks (timing attack [25], simple and differential power analysis (SPA, DPA) [27, 26], and single and differential electromagnetic analysis (SEMA, DEMA) [18, 34, 35]).

We will not enter into details about these attacks here, but it is worth pointing out that elliptic curve cryptography allows some countermeasures against them, based on various representations of points, and on different addition formulas [16, 24, 11][7]. This topic is not discussed in SEC1, and the addition formulas they

[6] The situation is not so clear regarding the SEC1 document: a strict interpretation of the specifications suggests the parameters must be fixed at key generation time, and the additional note from the P1363a document is not present here; anyway, a clarification of this issue would be useful.

[7] Note that most of these techniques are patented

present does not resist side-channel attacks. One might however argue that this is not the scope of SEC1: many other "practical" issues are also left aside by the document, and the addition formulas they propose are probably too inefficient to be used in a practical application anyway, so it might be considered that they are presented only for informational purpose.

However, the issue of whether an SPA-resistant implementation could be deemed as "compatible" with SEC1 would deserve some clarification.

4.5.2 Random generation

A lot of parameters (key pair, ephemeral key, ...) have to be generated randomly. This step must not be underestimated, since generating cryptographically secure random (or pseudorandom) numbers is one of the most difficult tasks of a cryptographer; many examples exist of applications whose security has been defeated by exploiting weaknesses in their random generator. One may regret the absence of a more detailed description of random generation in SEC1, or at least some references to other standards addressing this issue.

4.5.3 Verification of parameters' validity

Many cryptographic attacks are also based on the adversary presenting improperly formed parameters to Alice, in order to recover information about her secret parameters in her answer. A typical example in ECIES's context is the aforementioned small subgroup attack, based on the use of an invalid public key. It is therefore very important to check that the parameters received from a third party are correctly formed.

SEC1 describes a large numbers of such tests to be performed during the key generation, encryption, and so on – which is definitely a good point for a standard. We cannot insist too much on the fact that these tests are not facultative.

5 Conclusion

In view of the current state-of-the-art, our security analysis did not show any serious weakness in the ECIES scheme, except that we strongly recommend to avoid using the XOR encryption as a symmetric encryption primitive.

However, our analysis of the security proofs shows that they do not apply as easily as expected to ECIES. In particular, some of the implementation choices done in [15] do not fulfill the security level they are supposed to have.

Moreover, these weaknesses are demonstrated by some attacks that, although minor (in the sense that their do not seem practically exploitable by an attacker) show nonetheless that the security properties claimed by the security proofs are not fully satisfied.

In view of these attacks, a modification of ECIES, namely ECIES-KEM has been recently proposed by Shoup. However, this proposal is quite new, and its acceptance is not clear yet.

One of the criticisms issued against ECIES-KEM is that it is too recent, and still contains too much unspecified details for the moment. For example, Johnson [23] argues that "*The ECIES-KEM proposal is derivative from ECIES and has just begun the ISO consensus process; as such, at this time it is unclear what the final output will be*".

Besides that, Certicom has recently issued a document in reaction to Shoup's modification proposals, this document considers every change proposal, and refutes many of them, mainly for flexibility or concordance with practical needs reasons. This document has been sent to Nessie [23]. The Nessie team has announced [32] that they would consider both the original submission and Shoup's ECIES-KEM proposal.

As can be seen, there is no clear consensus yet in the scientific community as to whether this modified scheme should replace ECIES or not. Our advice would therefore be to let some time elapse before adopting ECIES or an updated version of it. This would be especially useful to avoid the risk of coexistence of several similar, but nonetheless incompatible "standards".

6 Acknowledgements

The authors are grateful to Mathieu Ciet for sharing his expertise of elliptic curve cryptography.

References

[1] *Nessie (New European Schemes for Signatures, Integrity, and Encryption)*, Project funded by the European Community, see `http://www.cryptonessie.org/`.

[2] M. Abdalla, M. Bellare, and P. Rogaway, *DHAES: an encryption scheme based on the Diffie-Hellman problem, submission to IEEE P1363a*, Available at `http://grouper.ieee.org/groups/1363/P1363a/Encryption.html#ABR`, September 1998.

[3] _____, *The oracle Diffie-Hellman assumptions and an analysis of DHIES*, Topics in Cryptology - CT-RSA 2001 (D. Naccache, ed.), Lecture Notes in Computer Science, vol. 2020, Springer, April 2001, pp. 143–158.

[4] _____, *DHIES, an encryption scheme based on the Diffie-Hellman problem*, Available at `http://www.cse.ucsd.edu/users/mihir/crypto-research-papers.html`, September 18, 2001.

[5] American National Standards Institute, *ANSI X9.52-1998: Triple Data Encryption Algorithm Modes of Operation*, 1998.

[6] _____, *ANSI X9.71-199x: keyed hash message authentication code*, Working Draft, March 1998.

[7] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway, *Relations among notions of security for public-key encryption schemes*, Advances in Cryptology - Crypto '98 (Berlin) (H. Krawczyk, ed.), Springer-Verlag, 1998, Lecture Notes in Computer Science Volume 1462, pp. 26–45.

[8] M. Bellare and P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*, Proc. of First Annual Conference on Computer and Commmunications Security, 1993, ACM.

[9] I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society, vol. 265, Cambridge University Press, 2000.

[10] D. Boneh, *Review of SEC1: Elliptic curve cryptography*, Available at `http://www.secg.org`.

[11] E. Brier and M. Joye, *Weierstrass elliptic curves and side-channel attacks*, Proc. of PKC '2002 (David Naccache and Pascal Paillier, eds.), Lecture Notes in Computer Science, vol. 2274, Springer, 2002, pp. 335–345.

[12] D.R.L. Brown, *Concrete lower bounds on the security of ECDSA in the generic group model*, Preprint, 2001.

[13] D.R.L. Brown and D.B. Johnson, *Formal security proofs for a signature scheme with partial message recovery*, Topics in Cryptology - CT-RSA 2001 (D. Naccache, ed.), Lecture Notes in Computer Science, vol. 2020, Springer, April 2001, pp. 126–142.

[14] R. Canetti, O. Goldreich, and S. Halevi, *The random oracle methodology, revisited*, Proc. of STOC '98, ACM, 1998.

[15] Certicom Research, *Standards for efficient cryptography (SECG)*, Available from `http://www.secg.org/secg_docs.htm`, September 2000.

[16] J.-S. Coron, *Resistance against differential power analysis for elliptic curves cryptosystems*, Cryptographic Hardware and Embedded Systems - CHES '99 (Çetin K. Koç and Christof Paar, eds.), Lectures Notes in Computer Science (LNCS), vol. 1717, Springer-Verlag, August 1999.

[17] S. Galbraith, F. Hess, and N.P. Smart, *Extending the GHS Weil Descent Attack*, Advances in Cryptography - Proceedings of EUROCRYPT 2002 (L. R. Knudsen, ed.), Lecture Notes in Computer Science, vol. 2332, Springer, 2002, pp. 29–44.

[18] K. Gandolfi, C. Mourtel, and F. Olivier, *Electromagnetic analysis: Concrete results*, Proc. of Cryptographic Hardware and Embedded Systems (CHES 2001) (Çetin Kaya Koç, David Naccache, and Christof Paar, eds.), Lecture Notes in Computer Science, vol. 2162, Springer, 2001, pp. 251–261.

[19] P. Gaudry, *An Algorithm for Solving the Discrete Logarithm Problem on Hyperelliptic Curves*, Advances in Cryptography - EUROCRYPT 2000 (B. Preneel, ed.), Lecture Notes in Computer Science, vol. 1807, Springer, 2000, pp. 19–34.

[20] P. Gaudry, F. Hess, and N.P. Smart, *Constructive and Destructive Facets of Weil Descent on Elliptic Curves*, Journal of Cryptology **15** (2002), no. 1, 19–46.

[21] M. Jakobsson and C.P. Schnorr, *Security of signed ElGamal encryption*, Proc. of Advances in Cryptology - ASIACRYPT '00 (T. Okamoto, ed.), LNCS, vol. 1976, Springer, 2000, pp. 73–89.

[22] D. Johnson, *Diffie-Hellman key agreement small subgroup attack*, Contribution to X9F1 by Certicom, July 1996.

[23] _____, *Certicom analysis comparing Nessie submission of ECIES and Victor Shoups ECIES-KEM ISO proposal of sept. 17, 2001*, Available at `http://www.cryptonessie.org/tweaks.html`, November 2001.

[24] M. Joye and C. Tymen, *Protections against differential analysis for elliptic curve cryptography*, Cryptographic Hardware and Embedded Systems - CHES 2001 (Çetin K. Koç, David Naccache, and Christof Paar, eds.), Lectures Notes in Computer Science (LNCS), vol. 2162, Springer-Verlag, August 2001, pp. 377–390.

[25] P. Kocher, *Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems*, Advances in Cryptology - CRYPTO '96, Santa Barbara, California (N. Koblitz, ed.), LNCS, vol. 1109, Springer, 1996, pp. 104–113.

[26] P. Kocher, Jaffe J., and B. Jub, *Differential power analysis*, Proc. of Advances in Cryptology – CRYPTO '99 (M. Wiener, ed.), LNCS, vol. 1666, Springer-Verlag, 1999, pp. 388–397.

[27] P. Kocher, J. Jaffe, and B. Jun, *Introduction to differential power analysis and related attacks*, `http://www.cryptography.com/dpa/`, 1998.

[28] H. Krawcyzk, M. Bellare, and R. Canetti, *HMAC: keyed hashing for message authentication*, Internet Engineering Task Force, Internet RFC 2104. Available from `http://www.ietf.org`, 1997.

[29] C. H. Lim and P. J. Lee, *A key recovery attack on discrete log-based schemes using a prime order subgroup*, Advances in Cryptology - Crypto '97 (Berlin) (Burt Kaliski, ed.), Springer-Verlag, 1997, Lecture Notes in Computer Science Volume 1294, pp. 249–263.

[30] Alfred J. Menezes, *Elliptic curve public key cryptosystems*, second ed., Kluwer Academic Publishers, 1995.

[31] B. Preneel, B. Van Rompay, L. Granboulan, G. Martinet, S. Murphy, R. Shipsey, J. White, M. Dichtl, P. Serf, M. Schafheutle, E. Biham, O. Dunkelman, V. Furman, M. Ciet, J.-J. Quisquater, F. Sica, L. Knudsen, and H. Raddum, *Nessie phase i: selection of primitives*, Available at `http://www.cryptonessie.org/`, September 2001.

[32] _____, *Update on the selection of algorithms for further investigation during the second round*, Nessie deliverable D18, March 2002.

[33] _____, *Security evaluation of NESSIE first phase*, Nessie deliverable D13, September 2001.

[34] Jean-Jacques Quisquater and David Samyde, *A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions: the SEMA and DEMA methods*, Eurocrypt rump session, 2000.

[35] _____, *Electromagnetic analysis (EMA): measures and countermeasures for smart cards*, Smart cards programming and security (e-Smart 2001), Lectures Notes in Computer Science (LNCS), vol. 2140, Springer, 2001, pp. 200–210.

[36] V. Shoup, *A proposal for an ISO standard for public key encryption*, Available at `http://shoup.net/papers/`, December 2001.

[37] N. Smart, *The exact security of ECIES in the generic group model*, Proc. of 8th IMA International Conference on Cryptography and Coding, December 2001, pp. 73–84.

[38] N. P. Smart, *How Secure are Elliptic Curves over Composite Extension Fields?*, Advances in Cryptology - Proceedings of EUROCRYPT 2001 (Berlin) (B. Pfitzmann, ed.), Lecture Notes in Computer Science, vol. 2045, Springer, 2001, pp. 30–39.

[39] J. Stern, D. Pointcheval, J. Malone-Lee, and N. P. Smart, *Flaws in applying proof methodologies to signature schemes*, Advances in Cryptology - Crypto 2002 (Berlin) (M. Yung, ed.), Springer-Verlag, 2002, Lecture Notes in Computer Science, to appear.