Evaluation Report on the **ECIES Cryptosystem**

Jacques Stern

1 Introduction

This document is an evaluation of the **ECIES Cryptosystem**. Our work is based on the analysis of document [10], which provides both the specification and self-evaluation of the scheme, as well as on the research papers [1, 2] and the comments provided by Victor Shoup in [41], where additional security arguments can be found. This cryptosystem has been given alternative names: **DHAES** (*Diffie-Hellman Augmented Encryption Scheme*), **DHIES** (*Diffie-Hellman Integrated Encryption Scheme*) and **ECAES** (*Elliptic Curve Augmented Encryption Scheme*), while **ECIES** stands for *Elliptic Curve Integrated Encryption Scheme.* The present report is organized as follows: firstly, we briefly review the cryptosystem; next we discuss the security level of the cryptographic primitive which underlies the scheme and analyze its relation to the difficulty of the discrete logarithm problem on elliptic curves; finally, we evaluate the security level of the scheme itself in the light of strong security notions such as semantic security and security against adaptive chosen-ciphertext attacks. This is as requested by IPA.

2 Brief description of the scheme

2.1 Specification review

ECIES is based on the hardness of the discrete logarithm problem over an elliptic curve. The cryptosystem uses elliptic curves E over some prime p, $|p| = m$, or elliptic curves over $\mathbb{F}_{2^m}$. In the first case, possible values for m are

$$\{112, 128, 160, 192, 224, 256, 384, 521\}$$

and, in the second case:

$$\{113, 131, 163, 193, 233, 283, 409, 571\}.$$

Once the curve E has been chosen, a base point G is chosen on E, which generates a subgroup of prime order n, such that, denoting by $\#(E)$ the number of points in the curve and defining the *cofactor* h by $h = \#(E)/n$, inequality $h \leq 4$ holds.

ECIES is a hybrid encryption scheme built on an adaptation of the El Gamal cryptosystem [15] to the context of elliptic curves. The basic function f on which ECIES is based is defined by

$$\begin{array}{rcl} f : \{0, \ldots, n-1\} & \longrightarrow & E \\ r & \longmapsto & r \cdot G \end{array}$$

where $r \cdot G$ is obtained, by means of the usual elliptic curve addition, as the sum of r times G. Inverting f is precisely the elliptic curve discrete logarithm problem (ECDLP). Clearly, f is one-to-one. The inverse function, denoted $\log_G$, is believed to be hard to compute. Another function used by the scheme is the Diffie-Hellman function:

$$\begin{array}{rcl} \mathsf{DH}_G : E^2 & \longrightarrow & E \\ X, Y & \longmapsto & \log_G(Y) \cdot X = \log_G(X) \cdot Y \end{array}$$

A variant of this function is the *cofactor* Diffie-Hellman function:

$$\begin{array}{rcl} \mathsf{DH}'_G : E^2 & \longrightarrow & E \\ X, Y & \longmapsto & h \cdot \log_G(Y) \cdot X = h \cdot \log_G(X) \cdot Y \end{array}$$

We will omit subscript G in the above when the context is clear.

Besides the curve parameters, the public key of ECIES consists of an element of the curve of order n, say Q. The private key d is the discrete logarithm of Q in base G.

Before going further, we introduce a more formal framework, that will be useful when we later perform the security analysis. A public-key encryption scheme on a message space $\mathcal{M}$ consists of three algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$:

- the key generation algorithm $\mathcal{K}(1^m)$ outputs a random pair of private-public keys $(\mathsf{sk}, \mathsf{pk})$, relatively to a security parameter m
- the encryption algorithm $\mathcal{E}_{\mathsf{pk}}(M; r)$ outputs a ciphertext C corresponding to the plaintext $M \in \mathcal{M}$, using random coins r
- the decryption algorithm $\mathcal{D}_{\mathsf{sk}}(C)$ outputs the plaintext M associated to the ciphertext C, or a Reject answer if the ciphertext is invalid.

Thus, the key generation algorithm $\mathcal{K}(1^m)$ of the ECIES Cryptosystem produces, on input m, a public key pk consisting of a curve E, a base point G, its order n and an element Q of order n. In the case of elliptic curves over $\mathbb{F}_p$, the equation of the curve is

$$y^2 = x^3 + ax + b$$

and the curve parameters form a tuple (p, a, b, G, n, h), where h is the cofactor, and the public key is given by this tuple, plus the public element Q. In the case of elliptic curves over $\mathbb{F}_{2^m}$, the equation of the curve is

$$y^2 + xy = x^3 + ax + b$$

and the curve parameters form a tuple (m, f, a, b, G, n, h), where f is an irreducible polynomial of degree m over $\mathbb{F}_2$, which defines the extension field $\mathbb{F}_{2^m}$, and h is the cofactor, and the public key is similarly given by this tuple plus the public element Q. The private key $\mathsf{sk} = d$ is the discrete logarithm of Q in base G.

2.1.1 Encryption and decryption

- **Encryption** $\mathcal{E}_{\mathsf{pk}}(M; r)$ first produces an ephemeral key pair (r, R), where r is randomly chosen in $\{1, n-1\}$ and $R = r \cdot G$. Next it computes a shared secret value k from $\mathsf{DH}(Q, R) = r \cdot Q$ (or —alternatively— $\mathsf{DH}'(Q, R) = h \cdot r \cdot Q$). By "shared", we mean that the recipient of the message is also able to obtain k, as will be seen in the sequel. More precisely, in the description provided by [10, 22], k is defined to be z, which denotes the first coordinate of the above curve element (the Diffie-Hellman secret). In this case the shared secret value belongs to the underlying field, hence the name *shared field element.*

  From the shared secret value k, key material Key is generated by means of a KDF (*Key Derivation Function*). The scheme uses a MAC scheme and a symmetric encryption scheme. Both are keyed. Thus, an encryption key EK and a MAC key MK, of appropriate length, are needed. They are extracted from Key. Finally, the symmetric encryption scheme, keyed by EK, is applied to the message M, thus producing an encrypted message EM. Finally, a tag D is appended, applying the MAC scheme, keyed by MK, to EM (optionally followed by another piece of shared information). The final cryptogram is $C = R \,\|\, EM \,\|\, D$.

- **Decryption** $\mathcal{D}_{\mathsf{sk}}(C)$ is based on recovering k. Granted the private key d, it is possible to obtain the shared secret value k from $\mathsf{DH}(Q, R) = d \cdot R$ (or —alternatively— from $\mathsf{DH}'(Q, R) = h \cdot d \cdot R$). Once k has been retrieved, one can compute the two keys EK and MK, check the correctness of the tag D and decrypt EM, thus recovering the plaintext.

At this point, we wish to note that we have not been too careful with notations in the above description. For example, writing $C = R \,\|\, EM \,\|\, D$ treats R as a bit string or a byte string, whereas it is actually a point on an elliptic curve. Document [10] provides the adequate conversion routines, using —or not— the so-called point compression. We believe that our approach is suitable for performing a high level security analysis. This is why we use simplified notations and ignore type conversions. However, the same approach should not extend to the definition of encodings into bit strings. Especially, it is crucial at various steps, that specific encodings be one-to-one. In the sequel, we will be extremely careful with such encodings.

2.1.2 The key derivation function

The key derivation function is formally defined in [10, 22]. For our purpose, it is enough to state that it uses a hash function H, which is unkeyed and applied to k, to a counter and —optionally— to publicly shared information $\mathsf{SharedInfo}$. Function H is repeatedly applied to obtain key material Key of appropriate length (ℓ bits):

$$\mathsf{KDF}(k, \ell, \mathsf{SharedInfo}) = \lfloor K_1 \,\|\, K_2 \,\|\, \ldots \,\|\, K_i \rfloor_\ell$$

where $K_j = H(k \,\|\, j \,\|\, \mathsf{SharedInfo})$, for $j = 1, \ldots, i$ and $i = \lceil \ell / \mathsf{HLen} \rceil$.

Remark that, if $\mathsf{Key}_1 = \mathsf{KDF}(k, \ell_1, \mathsf{SharedInfo})$ and $\mathsf{Key}_2 = \mathsf{KDF}(k, \ell_2, \mathsf{SharedInfo})$, with $\ell_1 \leq \ell_2$, then Key_1 is a prefix of Key_2.

With respect to the scheme presented in [1], there is a slight difference, since the latter includes R in the input to KDF. Furthermore, the key material Key is derived from the entire information on Diffie-hellman secret, i.e. using an one-to-one encoding, and not the first coordinate only: the first coordinate z of an elliptic curve point is not one-to-one since two opposite points have identical first coordinate.

2.1.3 Symmetric encryption

Several encryption schemes can obviously be used in conjunction with ECIES. However, starting from the early papers proposing the scheme, one-time pad (XORing the encryption key with the message), has always been included as a suitable candidate. In terms of security, this is related to the observation that the ECIES construction only requires a symmetric encryption scheme that is semantically secure against passive attacks. Use of the one-time pad is now referred as the *stream cipher mode*. In this mode, the encryption key EK is of the same length as the message to encrypt.

There is still a slight difference between research papers [1, 2], and standardization documents [22, 10]. Both derive key material as

$$\mathsf{Key} = \mathsf{KDF}(k, \mathsf{MLen} + \mathsf{ELen}, \mathsf{SharedInfo}),$$

where k is an encoding of the shared secret value. However, while the former parse it as $\mathsf{Key} = \mathsf{MK} \,\|\, \mathsf{EK}$, the latter use a different parsing $\mathsf{Key} = \mathsf{EK} \,\|\, \mathsf{MK}$. In the first case, the MAC key depends on k only, whereas in the other case, the MAC key depends on k, and, additionally, on the bit-length of the plaintext. As will be seen later, this may have an important impact on security.

2.2 Comments on the specification

Document [10] is clearly written but mainly directed towards implementors. For example, one may wonder why it includes the cofactor Diffie-Hellman function: if X and Y are in the subgroup of prime order n generated by G, then the cofactor h, which is bounded by 4, has an inverse modulo n and $\mathsf{DH}(X,Y) = h^{-1} \cdot \mathsf{DH}'(X,Y)$. Thus, computing DH and DH' is strictly equivalent from a complexity theoretic viewpoint. Inclusion of the cofactor method eases the implementation. When an ephemeral key pair (r, R) is used in an encrypted message, there is no guarantee for the receiver that it has been correctly manufactured. Hence, in order to avoid subtle attacks, the receiver should check that R is a non unit element of the subgroup generated by G, which means

1. that R lies on the curve
2. that R is not $\mathcal{O}$
3. that R is of order n

Using DH' allows to discard the third check, since multiplication by h brings back anyway into the subgroup of order n.

In terms of security arguments, document [10] is rather sketchy and refers to [1, 2]. The latter are disappointing. By this, we do not mean that the security arguments that are offered are wrong. We will discuss them in more detail further on in the present report and, as will be seen, they are mathematically sound. However, they do not convey a totally convincing picture of the security of the actual scheme, for several reasons:

- the research papers [1, 2] are not too explicit on implementation details, such that encoding routines. This may seem natural since they have not been written for implementors. However, it turns out that some of the choices later adopted in the standardization proposals are clearly unfortunate.
- the assumption that the papers use is certainly contrived. It has been severely criticized by other researchers. We quote [42]:

  > The authors make intractability assumptions that are *interactive*; indeed, these intractability assumptions amount to little more than a restatement of the definition of security in terms of the particular implementation they propose.

  To be fair, it should be added that the opinion comes from a competitor. In any case, the present report will aim at clarifying both issues.

3 Security level of the cryptographic primitive

In this section, we investigate the security of the underlying cryptographic primitive, both in terms of complexity-theoretic reductions and with respect to the recommended parameters.

3.1 Complexity-theoretic arguments

Documents [10, 1, 2] relate the security of the scheme to a version of the discrete logarithm for elliptic curves. There are several basic primitives that can be considered.

3.1.1 The elliptic curve decisional and computational Diffie-Hellman hypotheses

We keep the notations of Section 2.1. Recall that the decisional Diffie-Hellman hypothesis on an elliptic curve E, with a large subgroup of prime order, asserts that it is hard to distinguish the distributions $\mathbf{D_E}$ and $\mathbf{R_E}$, where

$$\mathbf{R_E} = \{(G_1, G_2, U_1, U_2)\}$$

with all four elements taken at random in the large subgroup and

$$\mathbf{D_E} = \{(G_1, G_2, U_1, U_2)\}$$

with $\log_{G_1}(U_1) = \log_{G_2}(U_2)$. A quantitative version measures the maximum advantage $\mathsf{AdvDDH}(t)$ of a statistical test T that runs in time t. This means the maximum of the difference of the respective probabilities that T outputs 1, when probabilities are taken over $\mathbf{D_E}$ or $\mathbf{R_E}$.

As is well known, there is a standard self-reducibility argument: by randomization, it is possible to transform an arbitrary tuple (G_1, G_2, U_1, U_2) such that $G_1 \neq G_2$ into a random equivalent one, i.e. the output is in $\mathbf{D_E}$ (resp. $\mathbf{R_E}$), if and only if the input is. Thus, if $\mathsf{AdvDDH}(t)$ is significant, one can use a distinguisher to decide, with probability close to one, whether a tuple is in $\mathbf{D_E}$. This involves performing repeated tests with the distinguisher, and deciding whether the number of one outputs has a bias towards $\mathbf{D_E}$ or $\mathbf{R_E}$. Based on the law of large numbers, a decision with small constant error probability requires running $O(\mathsf{AdvDDH}^{-2})$ tests. One can decrease the error probability drastically by repeating the above computations an odd number of times and deciding based on the median of the averages. In [31], the authors claim that one can reach error probability 2^{-n} by repeating the test $O(p(n)).\mathsf{AdvDDH}^{-1}$, where p is a polynomial, but the proof is missing. In any case, the loss in the reduction is huge. Thus, despite its elegance, the self-reducibility argument is a bit misleading in terms of exact security.

Related to the above is the elliptic curve computational Diffie-Hellman assumption (ECCDH) and the elliptic curve discrete logarithm assumption. The former states that it is hard to compute $xy \cdot G$ from G, $x \cdot G$ and $y \cdot G$, while the latter states that it is hard to compute x from G and $x \cdot G$. It is obvious that DDH is a stronger assumption than CDH, which in turn, is stronger than the discrete logarithm assumption. However, no other relation is known and the only way to solve the hard problems underlying DDH or CDH is to compute discrete logarithms.

It should be mentioned that the DDH cannot hold in groups with a small subgroup. This is why cryptographic schemes usually work with a subgroup of an elliptic curve of large prime order and the current proposal is no exception. Even with this proviso, there are subtle protocol attacks using invalid keys, i.e. keys that do not belong to the prescribed large subgroup (see [30]). In the present context, such attacks are addressed either by performing consistent checks to verify e.g. that elements, that are claimed to belong to the subgroup generated by G, actually lie within this subgroup. Lightweight checks are possible if the cofactor Diffie-Hellman function is used.

3.1.2 Security of the shared field element

In this section, we consider the simplified version of the scheme that simply establishes the shared field element, denoted by z in the above. In this restricted context, It appears that the security of the scheme is closely connected to the decisional Diffie-Hellman assumption. We will only consider the setting where the Diffie-Hellman function is applied. With the notations of Section 2.1, z is the first coordinate of $P = \mathsf{DH}(Q, R)$. We would like to see that z looks like a random string to a passive adversary. However, this cannot hold in a simple-minded approach: given an elliptic curve E over $\mathbb{F}_p$ or $\mathbb{F}_{2^m}$, a field element x is not necessarily the first coordinate of a point of order n on the curve. We let $\mathcal{X}$ be the set of such x.

Theorem 1 *Based on the elliptic curve decisional Diffie-Hellman hypothesis ECDDH, it is hard to distinguish the distribution*

$$(G, Q, R, z)$$

generated by the cryptosystem, from the analogous distribution with z replaced by a random element of $\mathcal{X}$. More accurately, if there is an adversary $\mathcal{A}$ that distinguishes the above distributions within time bound t, with advantage ε, then there exists a machine $\mathcal{B}$ that solves the decisional Diffie-Hellman problem with advantage ε within time bound $t + \tau$, where τ accounts for a few extra elliptic curve operations and is bounded by $\mathcal{O}(m^3)$.

In the above, the advantage in distinguishing two distributions is the absolute value of the difference of the probabilities that the algorithm outputs 1, with inputs taken from each.

Proof. Let $\mathcal{A}$ be an adversary that distinguishes the two distributions defined in the theorem. We show how to attack the ECDDH by distinguishing the distributions $\mathbf{D}$ and $\mathbf{R}$, where

$$\mathbf{R} = \{(G, Q, R, z)\}$$

with G, Q, R at random in the subgroup of order n of E and z taken at random from $\mathcal{X}$

$$\mathbf{D} = \{(G, Q, R, z)\}$$

with z the first coordinate of $P = \mathsf{DH}(Q, R)$ We run the key generation algorithm and generate an elliptic curve E together with a random element of large prime order G. We next show how to use $\mathcal{A}$ to break the ECDDH: we take the base point of the cryptosystem to be the first element of the input to $\mathcal{A}$ and we complete the public key by the second element Q. This implicitly defines a private key. Next we assume that a public key encryption occurs with an ephemeral key, whose public part is R. Finally, we submit (G, Q, R, z) to $\mathcal{A}$. If the original input is from $\mathbf{D}$, the last element of the tuple is exactly as produced by the cryptosystem. On the other hand, if the input is from $\mathbf{R}$, the last coordinate is a random element of $\mathcal{X}$. Thus, we have obtained a distinguisher between $\mathbf{D}$ and $\mathbf{R}$, with exactly the same advantage as $\mathcal{A}$. Finally, the advantage of any algorithm $\mathcal{A}$ that runs in time t is bounded by $\mathsf{AdvDDH}(O(t))$, where $O(t) = t + \tau$ accounts for the few extra elliptic curve operations needed to compute the data to be handled to $\mathcal{A}$. $\square$

Remark. It would be desirable to ensure that one gets a bit-string indistinguishable from a random string with m bits, which would mean that the information z is semantically secure in the sense of the seminal paper [19]. However, we do not see any argument that would give such guarantee. As mentioned in [1, 2], it is possible to obtain such a bit-string by applying a randomly keyed universal hash function, following the method described in [31] and also used in [42]. Recall that, if H_k is a universal hash function, keyed by k, with ℓ-bit outputs, then, the leftover hash lemma of [23] implies that hashing a set of 2^λ bit-strings produces a distribution $(k, H_k(x))$, whose distance to the uniform distribution is less than $1/2^{(\lambda-\ell)/2}$. Here, λ is a few bits below the size of the security parameter m. Thus in order to get a bound at most $1/2^{128}$ and to obtain a 128 bit encryption key and a 128 bit MAC key, one would need $m \geq 512$. This would only be compatible with the largest suggested parameter for the scheme. In any case, this is not the path followed by the submission.

3.2 Size of the parameters

As was just observed the security of a simplified version of the scheme appears closely related to the ECDDH for the class of elliptic curves generated by the cryptosystem, even if there is a minute security loss in terms of exact security. As will be seen in the sequel, the scheme is actually based on a formally different security assumption. In any case, the only method known to attack the decisional Diffie-Hellman problem on elliptic curves is to solve the underlying discrete logarithm problem (ECDLP). Therefore, in order to estimate whether the specific restrictions on the curve and the suggested parameters offer a wide security margin, it is useful to review the performances of the various algorithms known for the ECDLP. We will distinguish between exponential algorithms, whose running time depend on the size of the group and subexponential algorithms, which apply to specific classes of weak curves.

3.2.1 Exponential algorithms

The best algorithm known to date for solving the DLP in any given group G is the Pollard ρ-method from [34] which takes computing time equivalent to about $\sqrt{\pi n/2}$ group operations. In 1993, van Oorschot and Wiener in [44], showed how the Pollard ρ-method can be parallelized so that, if t processors are used, then the expected number of steps by each processor before a discrete logarithm is obtained is $\simeq \frac{\sqrt{\pi n/2}}{t}$. In order to compute the discrete logarithm of Y in base G, each processor computes a kind of random walk within elements of the form $a \cdot G + b \cdot Y$, selecting X_{i+1} through one of the three following rules

1. set $X_{i+1} = G + X_i$
2. set $X_{i+1} = 2 \cdot X_i$
3. set $X_{i=1} = Y + X_i$

Decisions on which rule to apply are made through a random-looking but deterministic computation, using e.g. hash values. "Distinguished" points X_i are stored together with their representations $X_i = a_i \cdot G + b_i \cdot Y$ in a list that is common to all processors. When a collision occurs in the list, the requested discrete logarithm becomes known.

In recent work (see [18, 46]), it was shown how to improve the above by a multiplicative factor $\sqrt{2}$. This takes advantage of the fact that one can simultaneously handle a point X and its opposite $-X$. Slightly better improvements can be obtained for specific curves with automorphisms.

The progress of such algorithms is well documented. In April 2000, the solution to the ECC2K-108 challenge from Certicom [9] led to the computation of a discrete logarithm in a group with 2^{109} elements (see [21]). This is one of the largest effort ever devoted to a public-key cryptography challenge. The amount of work required to solve the ECC2K-108 challenge was about 50 times that required to solve the 512-bit RSA cryptosystem (see [8]) and was thus close to 400000 mips-years.

It is expected that such figures will grow slowly, unless unexpected discoveries appear in the area. From the predictions in [28], one can infer that the proposed range of parameters (from 112 to 521 bits) will presumably allow for a choice that guarantees security for the foreseeable future, at least for the next 50 years.

3.3 Security against subexponential attacks

As is well known, there are two classes of elliptic curves for which non trivial attacks have been found. They are

1. the supersingular curves
2. the anomalous curves

Supersingular curves over a field $\mathbb{F}_q$, with q a power of p, are defined by the condition that the trace of the Frobenius map is zero modulo p. For such curves, Menezes, Okamoto and Vanstone (MOV) have shown how to reduce the discrete logarithm problem to the DLP in an extension field $\mathbb{F}_{q^k}$ of $\mathbb{F}_q$, with small k. Note that, for elliptic curves over a prime field $\mathbb{Z}_p$, those curves have exactly $p+1$ elements and are specifically excluded by the key generation algorithm which performs the following check

$$p^B \neq 1 \bmod n \qquad \text{for any } 1 \leq B \leq 20$$

Anomalous curves are those which contain a p-torsion point other than $\mathcal{O}$, or, equivalently, those whose Frobenius map has trace congruent to one modulo p. For such curves, work of Semaev ([37]), Rück ([35]), Smart ([43]) and Satoh-Araki ([36]) has shown how to solve the p-part of the DLP in polynomial time. Note that, for elliptic curves over a prime field $\mathbb{Z}_p$, those curves have exactly p elements and are specifically excluded by the key generation algorithm.

The MOV reduction constructs an embedding from the curve into the multiplicative group of a suitable extension field of $\mathbb{F}_q$ and can be applied in a more general setting than originally envisioned by the authors. However, if the base point is an element of order n, n is necessarily a divisor of $q^k - 1$. Recently, Balasubramanian and Koblitz have shown in [4] that this condition was sufficient to carry the MOV reduction. The key generation algorithm specifically addresses this question. In the case of curves over $\mathbb{F}_p$, one gets that $p^k = 1 \bmod n$. From this, it follows that k is at > 20, which is large enough to turn down subexponential algorithms in the extension field. In the case of curves over $\mathbb{F}_{2^m}$, there is an analogous test

$$2^{mB} \neq 1 \bmod n \qquad \text{for any } 1 \leq B \leq 20$$

with the same consequences.

Another reduction similar to the MOV reduction has appeared in the literature. It is due to Frey and Rück [17] (see also [16]) and can be stated in the more general context of Jacobians on which the Tate pairing exists. Let m be an integer relatively prime to q, and let $\mu_m(\mathbb{F}_q)$ be the group of roots of unity in $\mathbb{F}_q$ whose order divides m. Assume that the Jacobian $J(\mathbb{F}_q)$ contains a point of order m. Then there is a surjective pairing

$$\varphi_m : J_m(\mathbb{F}_q) \times J(\mathbb{F}_q)/mJ(\mathbb{F}_q) \rightarrow \mu_m(\mathbb{F}_q)$$

which is computable in $\mathcal{O}(\log q)$, where $J_m(\mathbb{F}_q)$ is the group of m-torsion points. This pairing, the so-called Tate pairing, can be used to relate the discrete logarithm in the group $J_m(\mathbb{F}_q)$ to the discrete logarithm in some extension $\mathbb{F}^\star_{q^k}$. In the case of elliptic curves, considered in the current context, the above is applicable only if the order n of the base point is a divisor of $q^k - 1$. As a consequence, the curves produced by the key generation algorithm are protected against the FR reduction, exactly due to the same argument used for MOV reduction.

3.3.1 Conclusion

Based on current estimates, it appears that the range of proposed parameters for ECIES allows choices that should remain secure for at least fifty years. However, even if it offers a guarantee that the MOV and FR reductions do not apply, the key generation algorithm leaves open the possibility to choose curves with complex multiplication or even Koblitz curves (curves over $\mathbb{F}_{2^m}$ with a and b in the two-element field). This is somehow contrary to the current trend, which would recommend having the curve generated at random and ensuring that there is a point of large prime order by counting the number of elements of the curve by means of the SEA algorithm [29]. Considering that the appropriate warnings are given, this does not constitute a strong objection to the proposal under review.

4 Security Analysis

Documents [1, 2] include a detailed yet intricate proof that the scheme is secure, based on a non-standard assumption that will be described later, and on security hypotheses on the encryption scheme and on the MAC scheme. In the present report, we have tried to include a more standard argument.

4.1 Formal framework

An asymmetric encryption scheme is *semantically secure* if no polynomial-time attacker can learn any bit of information about the plaintext from the ciphertext, except its length. More formally, an asymmetric encryption scheme is (t, ε)-IND where IND stands for *indistinguishable*, if for any adversary $\mathcal{A} = (A_1, A_2)$ with running time bounded by t, the advantage

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = \Pr_{\substack{b \xleftarrow{R} \{0,1\} \\ r \xleftarrow{R} \Omega}} \left[\begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{K}(1^m), (M_0, M_1, st) \leftarrow A_1(\mathsf{pk}) \\ c \leftarrow \mathcal{E}_{\mathsf{pk}}(M_b; r) : A_2(c, st) \overset{?}{=} b \end{array} \right] - 1/2$$

is $< \varepsilon$, where the probability space includes the internal random coins of the adversary, and M_0, M_1 are two equal length plaintexts chosen by the adversary in the message-space $\mathcal{M}$.

Another security notion has been defined in the literature, the so-called *non-malleability* [14]. Informally is states that it is impossible to derive, from a given ciphertext, a new ciphertext such that the plaintexts are meaningfully related. We won't discuss this notion any further since it has been proven equivalent to semantic security in an extended attack model.

The above definition of semantic security covers passive adversaries. It is a *chosen–plaintext* or CPA attack since the attacker can only encrypt plaintext. In the extended model, the *adaptive chosen–ciphertext* or CCA attack, the adversary is given access to a decryption oracle and can ask the oracle to decrypt any ciphertext, with the only restriction that it should be different from the challenge ciphertext. It has been proven in [5] that, under CCA, semantic security and non-malleability are equivalent. This is the strongest security notion currently considered.

4.2 Chosen ciphertext security of key encapsulation

Before we turn to the actual security analysis, we would like to focus on what can be called *key encapsulation*, a term introduced in [42], but which we use here with a slightly different meaning. This requires extending Theorem 1 to the context of CCA–adversaries: the attacker is allowed to query a decryption oracle by submitting an element R of the elliptic curve and receiving the key Key as the answer, where Key is computed from (G, Q, R) as prescribed by the cryptosystem. Note that Key is computed by the key derivation function: $\mathsf{Key} = \mathsf{KDF}(k, \ell, \mathsf{SharedInfo})$. To avoid the dependency on ℓ, we put an upper bound L on ℓ, and we assume that the adversary receives a key of maximal length $\mathsf{H}(k) = \mathsf{KDF}(k, L, \mathsf{SharedInfo})$. Function H will be later modeled as a random oracle. Still, two difficulties arise. The first is related to the case where the key material is derived from the shared field element $k = z$ only, and may open a security breach. The second lies in the simulation of the decryption queries, and is solved, in documents [1, 2], by basing the scheme on a non-standard security assumption.

4.2.1 Weakness of using the shared field element

We first show that defining the key material from the shared field element $k = z$ only (i.e from the first coordinate of the Diffie-Hellman secret) is incorrect. Note that this is the standard method in [10]. Indeed, by simply querying the key encapsulation scheme at $-R$, one immediately obtains the value $\mathsf{H}(z)$ corresponding to R.

We thus insist on the fact that the key material Key has to be derived from a one-to-one encoding k of $\mathsf{DH}(Q, R)$, or better of both R and $\mathsf{DH}(Q, R)$ (as done in the original version [1]), or simply R and z (which is equivalent from the information theoretic point of view). In the following we assume that the shared secret value is $k = (R, z)$, which is an option in the IEEE standard [22]. It can be noted that k is no longer a field element, but since it is used as an input to a hash function, this is not a problem.

4.2.2 The random oracle model

Even if one derives the key material as the image of R and z by a random oracle H, it still appears impossible to write up a proof by lack of a correct simulator: when receiving a query R, the simulator should provide $\mathsf{H}(R, z)$, as prescribed by the cryptosystem. A natural option is to search in the H-List, a dynamic data structure consisting of all queries to the random oracle H, together with the respective answers, hoping to find the appropriate value of z. Although the approach is sound, the new difficulty is to spot the correct query, which amounts to solve an instance of the ECDDH problem. Thus, we are naturally led to consider the so-called gap problems (see [32]).

4.2.3 Gap–Diffie-Hellman Problem

In the following, we review the version of the so-called gap problems that is needed in the current context. Besides the original definition, we consider a weaker assumption. We first define oracles, that an adversary can call. Notations are straightforward and come from Section 2.1 and Section 3.1.1.

- *an ECDDH oracle*: on input (G, Q, R, P), it perfectly answers whether $P = \mathsf{DH}(Q, R)$ or not.
- *a δ-ECDDH oracle*: on any input (G, Q, R, P), it answers whether $P = \mathsf{DH}(Q, R)$ or not, with some error probability δ. More precisely, the advantage of this oracle is greater then $1 - \delta$:

$$\left| \begin{array}{l} \Pr[\mathsf{ECDDH}(G, Q, R, P) = 1 \,|\, (G, Q, R, P) \in \mathbf{D_E}] \\ - \Pr[\mathsf{ECDDH}(G, Q, R, P) = 1 \,|\, (G, Q, R, P) \in \mathbf{R_E}] \end{array} \right| \geq 1 - \delta.$$

The reason why we introduce the second oracle is that, as already noted in Section 3.1.1, if δ is significantly smaller than 1, one can use this δ-oracle ($\mathcal{O}((1-\delta)^{-2})$ times) to decide, with an error probability as small as one may want, whether an element is in $\mathbf{D_E}$ or not.

We now define the elliptic curve Gap Diffie-Hellman problem, which consists in solving the ECCDH problem having access to an ECDDH oracle. A weaker assumption is the intractability of the δ-Gap Diffie-Hellman problem, which consists in solving the ECCDH problem, having an access to a δ-ECDDH oracle.

Although this does not appear in the submission, we will show that ECIES is secure against CCA adversaries, provided the Gap Diffie-Hellman problem is intractable and this can be extended to the formally weaker version of the hypothesis. It can be observed that both versions of the gap problem are polynomially equivalent (at least under a non-uniform reduction). Indeed, let $\mathcal{A}$ be an adversary that computes the Diffie-Hellman function after κ queries to an ECDDH oracle, with probability ε (where κ and $1/\varepsilon$ are polynomially bounded). Then, from any δ-ECDDH oracle, one can build a δ'-ECDDH oracle, achieving $\delta' < \varepsilon/2\kappa$. Therefore, if one simulates the perfect oracle, called by $\mathcal{A}$, using this δ'-ECDDH simulator, then $\mathcal{A}$ succeeds in solving the computational Diffie-Hellman problem with probability $\varepsilon - \kappa \times \delta' \geq \varepsilon/2$. Still, even if both problems are polynomially equivalent, the computational cost of the above reduction may be huge, depending on the original value of δ. In the following, we denote by $\mathsf{Succ}^{\mathsf{GDH}}(\delta, t, \kappa)$ the maximal success probability of any adversary in computing the DH function, within time t after less than κ queries to a δ-ECDDH oracle.

4.2.4 Security Analysis

We turn to the semantic security of key encapsulation against CCA adversaries. Let $\mathcal{A}$ be an attacker receiving inputs taken from two distributions as follows:

1. either, from the distribution
$$\mathcal{D}_0 = (G, Q, R, \mathsf{H}(R, z))$$
generated by the cryptosystem
2. or else from the analogous distribution $\mathcal{D}_1$ where $\mathsf{H}(R, z)$ is replaced by a random element ρ of the same bit length.

$\mathcal{A}$ is allowed to query a decryption oracle by submitting R', with $R' \neq R$ and receiving the corresponding key material. In the random oracle model, we establish the following exact security result, where the advantage of $\mathcal{A}$ is the difference of the probabilities that $\mathcal{A}$ outputs 1.

Note again that without including R, or a one-to-one encoding of $\mathsf{DH}(Q, R)$, as part of the input to H, one could query $(G, Q, -R)$ to the oracle ($-R \neq R$, but the first coordinates are identical), and easily distinguish the distributions.

Theorem 2 *Let $\mathcal{A}$ be a CCA-adversary as above, with running time $\leq t$ and advantage ε, making q_D decryption queries and q_H H-oracle calls. Then*
$$\begin{aligned} \frac{\varepsilon}{2} &\leq \mathsf{Succ}^{\mathsf{GDH}}(\delta, t', 2 \cdot q_H) + 2 \cdot q_H \cdot \delta \\ t' &\leq t + 2 \cdot q_H \cdot \tau, \end{aligned}$$
where τ denotes the running time of the δ-ECDDH oracle.

In order to prove this theorem, we will repeatedly use the following simple yet useful lemma from [42].

Lemma 1 *Let E, F, and E', F' be events of two probability spaces such that both*

$$\Pr[E|\neg F] = \Pr[E'|\neg F'] \text{ and } \Pr[F] = \Pr[F'] \leq \varepsilon.$$

Then,

$$|\Pr[E] - \Pr[E']| \leq \varepsilon$$

Proof. We write

$$\Pr[E] = \Pr[E|\neg F]\Pr[\neg F] + \Pr[E|F]\Pr[F]$$
$$\Pr[E'] = \Pr[E'|\neg F']\Pr[\neg F'] + \Pr[E|F']\Pr[F']$$

Hence

$$\Pr[E] - \Pr[E'] = \Pr[E|\neg F](\Pr[\neg F] - \Pr[\neg F']) + (\Pr[E|F]\Pr[F] - \Pr[E|F']\Pr[F']).$$

The right hand side becomes $\Pr[E|F]\Pr[F] - \Pr[E|F']\Pr[F']$, which is bounded by ε. □

Proof. (of Theorem 2). It will be convenient to allow implicit calls to the oracle H. This means submitting an elliptic curve point R and receiving $\mathsf{H}(R, z)$, where z is the first coordinate of $\mathsf{DH}(Q, R)$. Of course, implicit calls and explicit calls should not contradict each other.

From $\mathcal{A}$, we build a machine $\mathcal{B}$, as in the proof of Theorem 1.

1. $\mathcal{B}$ receives its input (G, Q, R), a triple of elements from E; it takes the base point of the cryptosystem to be the first element G of the input. Next, it completes the public key by the second element Q. This implicitly defines a private key. It then assumes that a public key encryption occurs with an ephemeral key, whose public part is R.

2. $\mathcal{B}$ tosses a random coin b. If $b = 0$, $\mathcal{B}$ computes a correct tuple $(G, Q, R, \mathsf{H}(R, z))$ as prescribed (note that such computation requires one implicit call to the oracle); if $b = 1$, $\mathcal{B}$ picks a random value ρ and forms (G, Q, R, ρ). In both cases, $\mathcal{B}$ handles the tuple to $\mathcal{A}$, which returns a bit b';

3. when the execution of the distinguisher has ended, $\mathcal{B}$ outputs a tentative value for $\mathsf{DH}(Q, R)$, from the queries asked to H (see below).

Of course, during the entire simulation, $\mathcal{B}$ has to simulate answers from the random oracle, which means ensuring the consistency between implicit and explicit calls. This is done by maintaining, besides the usual H-List which stores explicit query-answer $((R, z), \rho)$ from H, a list of implicit calls together with their answers, that is (R, ρ) which means that $\mathsf{H}(R, z) = \rho$, where z is the first coordinate of $\mathsf{DH}(Q, R)$.

- For each fresh query (R, u) to H (i.e. not on the H-List), $\mathcal{B}$ computes the two elliptic curve points P_1, P_2, whose first coordinate is u (if they exist) and queries the δ-ECDDH oracle at (G, Q, R, P_i). If the oracle answers positively for one of them, $\mathcal{B}$ looks for (R, ρ) in the implicit list and returns the corresponding value $\mathsf{H}(R, u) = \rho$, if it is present. Otherwise, the oracle picks a random answer and adds it to the H-List.

- Fresh implicit calls at R are handled similarly: For each string u such that (R, u) appears as a question in the H-List, $\mathcal{B}$ computes the two elliptic curve points P_1, P_2, whose first coordinate is u (if they exist) and queries the δ-ECDDH oracle at (G, Q, R, P_i). As soon as it finds an element such that the answer of the δ-ECDDH oracle is positive, $\mathcal{B}$ returns the corresponding value $\mathsf{H}(R, u)$. If none is found, the oracle picks a random answer and adds it to the implicit list.

We also define a plaintext-extractor as follows:

- On a query R' to the decryption oracle, where G and Q are assumed to be the base point and the public key respectively, $\mathcal{B}$ makes an implicit query at R' and returns the answer. Note that, if an answer is computed from the H-List and the implicit list, as explained above, the extractor returns this value, whereas it is otherwise random.

Finally, we explain what is the final answer of $\mathcal{B}$ is, besides the bit computed by $\mathcal{A}$. When execution has ended, $\mathcal{B}$ makes an implicit query at R. If an answer is computed from the H-List, $\mathcal{B}$ returns the elliptic curve point P used for computing this answer as a solution to the ECCDH instance.

Altogether, the simulator makes $q_D + 1$ implicit calls to the oracle, one for each decryption query and one for the final step. Proper bookkeeping allows to perform the consistency checks by calling the δ-ECDDH oracle at most $2q_H$ times. Indeed, one can add a flag to each H-query (R, u), telling whether u corresponds to the first coordinate of $\mathsf{DH}(Q, R)$ or not.

We wish to compare the behavior of several games

1. game $\mathcal{G}_1$, where the decryption queries are answered by an actual decryption oracle, and the random oracle is perfect (note that there is no implicit call here)

2. game $\mathcal{G}_2$, where the decryption queries are answered by the plaintext-extractor, using a perfect ECDDH oracle. Calls to the random oracle, both implicit and explicit, are still assumed perfect

3. game $\mathcal{G}_3$, which is as $\mathcal{G}_2$ but where the oracle is simulated until $\mathcal{B}$ is able to output an answer. Further oracle queries are answered by a "true" oracle.

4. game $\mathcal{G}_4$, which is as $\mathcal{G}_3$ but stops (without calling the oracle), as soon as $\mathcal{B}$ is able to return a potential solution to the ECCDH instance

5. game $\mathcal{G}_5$, where a δ-ECDDH oracle is used.

We observe that the probability that $b' = b$ in game $\mathcal{G}_1$ is $(\theta_1 + (1 - \theta_0))/2$, where θ_0 be the probability that algorithm $\mathcal{A}$ outputs 1 on inputs taken from $\mathcal{D}_0$, while θ_1 is the probability that it outputs 1 when they are taken from $\mathcal{D}_1$. This is $1/2 + \varepsilon/2$. We relate the probabilities of the same event in all games, repeatedly using Lemma 1.

- To go from $\mathcal{G}_1$ to $\mathcal{G}_2$, we note that the plaintext-extractor perfectly simulates decryption if implicit and explicit calls are perfect.
- The simulation in game $\mathcal{G}_3$ is perfect since it cannot conflict with the implicit constraint coming from the challenge. Indeed, R cannot be implicitly queried since implicit queries are only asked in relation with decryption queries. The pair (R, z), where z is the x-coordinate of $\mathsf{DH}(Q, R)$, cannot be explicitly queried either, because this would mean that $\mathcal{B}$ has earlier found the value of $\mathsf{DH}(Q, R)$.
- To go from $\mathcal{G}_3$ to $\mathcal{G}_4$, we just have to exclude the event of probability that $\mathcal{B}$ has correctly computed $\mathsf{DH}(Q, R)$: $\mathsf{Succ}^{\mathsf{GDH}}(\delta, t', 2 \cdot q_H)$. This bounds the difference of the probabilities that each game outputs one.
- To go from $\mathcal{G}_4$ to $\mathcal{G}_5$, one just needs to estimate the probability that the δ-ECDDH oracle returns a wrong answer. Since it is queried $2q_H$ times, the failure probability is bounded by $2q_H\delta$.

At this point, we have bounded the difference of probabilities for $\mathcal{G}_1$ and $\mathcal{G}_5$ by:

$$\mathsf{Succ}^{\mathsf{GDH}}(\delta, t', 2 \cdot q_H) + 2 \cdot q_H \cdot \delta.$$

Note that game $\mathcal{G}_5$ runs within time bound $t' \leq t + 2q_H\tau$, where τ denotes the running time of the δ-ECDDH oracle. To conclude, consider the probability of that $\mathcal{G}_5$ outputs a correct guess $b' = b$. In $\mathcal{G}_5$, the value of H at (R, z), where z is the x-coordinate of $\mathsf{DH}(Q, R)$, is left random. Thus, bit b' is independent of b and the probability of having $b' = b$ is exactly $1/2$. This finishes the proof of the theorem. $\square$

4.3 The intractability hypothesis

Documents [1, 2] claim to avoid the random oracle. However, they use a contrived and non-standard assumption. The assumption states that it is difficult to distinguish tuples $(G, Q, R, \mathsf{H}(R, z))$, computed as prescribed by the cryptosystem, even calling an oracle which returns $\mathsf{H}(R', z')$, when queried at $R' \neq R$. Thus, this assumption is **exactly** the statement that key encapsulation is secure. Note that the variant of the assumption that would replace $(G, Q, R, \mathsf{H}(R, z))$ by $(G, Q, R, \mathsf{H}(z))$, and $\mathsf{H}(R', z')$ by $\mathsf{H}(z')$ is not correct. Thus, the security of key encapsulation does not hold for the version of ECIES appearing in [10].

In the sequel, we will show that any public-key encryption scheme, based on a secure key encapsulation protocol and built along the lines of [10], is secure provided the symmetric encryption scheme and the MAC scheme in use are secure. Thus, as suggested in [42], it actually seems that the security proofs of [1, 2] tell nothing significant on the public key part.

We find that the arguments in [10, 2] are even somehow misleading in terms of security. In [10], the encoding of group elements is not one-to-one, since it simply uses $k = z$, and accordingly the above proof is no more valid. Furthermore, the attack presented above can be used to break the semantic security of the key encapsulation scheme.

In [2], the encoding is one-to-one. However, R is not included in the input to H. In other words, $k = \mathsf{DH}(Q, R)$. As a result, there is a security loss of $2q_D q_H \delta$. This also increases the computational cost of the reduction by a factor q_D.

4.4 From key encapsulation to chosen ciphertext security

Following [40], we give a formal definition of a key encapsulation scheme. It consists of three algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$:

- the key generation algorithm $\mathcal{K}(1^m)$ outputs a random pair of private-public keys $(\mathsf{sk}, \mathsf{pk})$, relatively to a security parameter m
- the encapsulation algorithm $\mathcal{E}_{\mathsf{pk}}(r)$ outputs a ciphertext R using random coins r
- the decryption algorithm $\mathcal{D}_{\mathsf{sk}}(R)$ outputs the key material Key associated to the ciphertext R (and thus depending on the above random coins r.)

The security of such a scheme has been considered in Section 4.2. It states that an attacker cannot distinguish the distribution consisting of a ciphertext and the corresponding key material from the analogous distribution where the key material is replaced by a random string, even if he can query ciphertexts (other than the challenge ciphertext). For such a scheme, we denote by $\mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t, q_D)$ the maximal advantage for any adversary in distinguishing both distributions, within time bound t, after q_D queries to the decryption oracle.

For any such scheme, provided it produces enough key material and the MAC key is uniquely defined from r only, one can derive an hybrid encryption scheme exactly as in [1] or [10]. We prove the following.

Theorem 3 *Let $\mathcal{A}$ be a CCA–adversary attacking the hybrid cryptosystem, within time bound t, with advantage ε, making q_D queries to the decryption oracle. Then*

$$\begin{aligned} \varepsilon &\le \mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t', q_D) + \mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t') + q_D \times \mathsf{Succ}^{\mathsf{mac}}(t', 1) \\ t' &\le t + \mathcal{O}(q_D), \end{aligned}$$

where $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t')$, $\mathsf{Succ}^{\mathsf{mac}}(t', 1)$, respectively denote the security level of the symmetric encryption scheme and the MAC scheme, as defined below.

4.4.1 Symmetric encryption

A symmetric encryption scheme with a key-length ELen, on messages of length ℓ, consists of two algorithms (E, D) which depend on a ELen-bit string k, called the secret key:

- the encryption algorithm $\mathsf{E}_\mathsf{k}(m)$ outputs a ciphertext c corresponding to the plaintext $m \in \{0,1\}^\ell$, in a deterministic way;
- the decryption algorithm $\mathsf{D}_\mathsf{k}(c)$ recovers the plaintext m associated to the ciphertext c.

A security notion similar to those used for asymmetric encryption is considered, known as *semantic security* [19]: a symmetric encryption scheme is *semantically secure* if no attacker can learn any bit of information about the plaintext from the ciphertext, besides its length. More formally, a symmetric encryption scheme is (t, ε)-IND if, for any adversary $\mathcal{A} = (A_1, A_2)$ with running time bounded by t, $\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) < \varepsilon$, where

$$\mathsf{Adv}^{\mathsf{ind}}(\mathcal{A}) = \Pr_{\substack{\mathsf{k} \overset{R}{\leftarrow} \{0,1\}^{\mathsf{ELen}} \\ b \overset{R}{\leftarrow} \{0,1\}}} [(m_0, m_1, s) \leftarrow A_1(\mathsf{ELen}), c \leftarrow \mathsf{E}_\mathsf{k}(m_b) : A_2(c, s) \overset{?}{=} b] - 1/2$$

In the above, probabilities include the random coins of the adversary, and m_0, m_1 are two identical-length plaintexts in the message-space $\{0,1\}^\ell$.

We denote by $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t)$ the maximal advantage of any adversary, against the semantic security of the scheme (E, D), within time bound t. Note that, in the particular case of the the one-time pad, any adversary, whatever its computational power is, has advantage exactly zero.

4.4.2 Message authentication code

A message authentication code (a.k.a. MAC) with key-length MLen consists of two algorithms $(\mathsf{MAC.Gen}, \mathsf{MAC.Ver})$, which depend on a MLen-bit string k, called the secret key:

- the MAC generation algorithm $\mathsf{MAC.Gen}_{\mathsf{k}}(M)$, outputs a tag τ on an input message M;
- the MAC verification algorithm $\mathsf{MAC.Ver}_{\mathsf{k}}(M, \tau)$ outputs 1 or 0. A 1 answer means that τ is a valid tag of M.

It is required that for any key k and any message M,

$$\mathsf{MAC.Ver}_{\mathsf{k}}(M, \mathsf{MAC.Gen}_{\mathsf{k}}(M)) = 1$$

On the other hand, it should be hard for an adversary to build a valid message/tag pair without the secrete key, even if it has access to MAC oracles returning the value of $\mathsf{MAC.Gen}_{\mathsf{k}}(M)$ or of $\mathsf{MAC.Ver}_{\mathsf{k}}(M, \tau)$. As usual, the adversary is not allowed to query the oracle at the message for which it produces a forgery.

More formally, a MAC is (t, ε, q)-unforgeable if, for any adversary $\mathcal{A}$ with running time bounded by t, allowed to ask at most q queries to the oracles, $\mathsf{Succ}^{\mathsf{mac}}(\mathcal{A}) < \varepsilon$, where

$$\mathsf{Succ}^{\mathsf{mac}}(\mathcal{A}) = \Pr_{\mathsf{k} \overset{R}{\leftarrow} \{0,1\}^{\mathsf{MLen}}} [(M, \tau) \leftarrow \mathcal{A}^{\mathsf{MAC.Gen}_{\mathsf{k}}(\cdot), \mathsf{MAC.Ver}_{\mathsf{k}}(\cdot,\cdot)} : \mathsf{MAC.Ver}_{\mathsf{k}}(M, \tau) = 1]$$

In the above, probabilities also include the random coins of the adversary, and the forgery (M, τ) involves a tag τ which has not been obtained from $\mathsf{MAC.Gen}_{\mathsf{k}}(\cdot)$.

As for symmetric encryption scheme, we let $\mathsf{Succ}^{\mathsf{mac}}(t, q)$ denote the maximal success probability of any adversary, after at most q queries to the oracles, within time t.

In the following, we will specifically consider a scenario where the adversary asks a single query. In this setting, it is known that MACs built from universal hash functions [13] meet the corresponding security criterion, without any cryptographic assumption.

4.4.3 Implementation pitfalls

Before turning to the proof of theorem 3, we first discuss, in this section and the next one, the pitfalls that implementations may meet.

Firstly, the weakness spotted in section 4.2.1 directly contradicts the CCA security of the variant of the ECIES cryptosystem, where the key material Key is derived from the shared field element z only. Indeed, the challenge ciphertext $C = (R||EM||D)$ can be decrypted by querying $C' = (-R||EM||D)$.

Secondly, as noted in section 2.1.3, standardization efforts related to ECIES [22, 10], derive the key material Key from the shared secret value k and parse the result as $\mathsf{Key} = \mathsf{EK} \,||\, \mathsf{MK}$, where EK and MK are the encryption key and the MAC key respectively, of bit-length ELen and MLen. In stream cipher mode, this leads to the unpleasant feature that MK depends not only on k but also on the length of the plaintext. The next section shows that this also leads to contradicting CCA security.

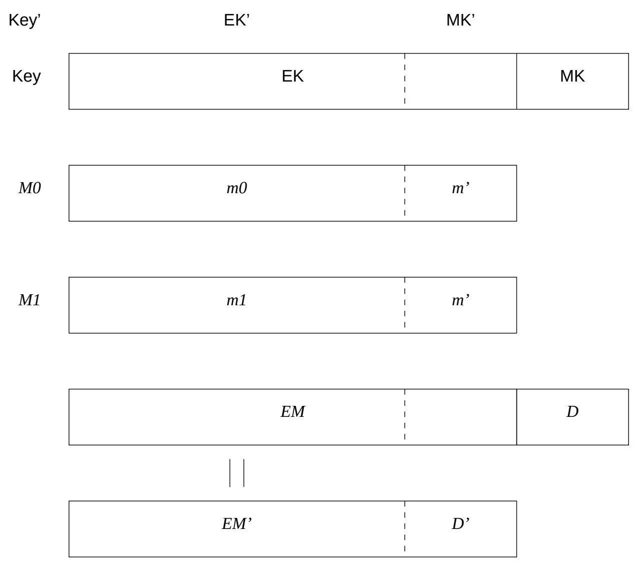

Figure 1: Attack against the stream cipher mode

4.4.4 Stream cipher mode

The KDF proposed in ECIES and described in Section 2.1.2, has the following property: if $\mathsf{Key}_1 = \mathsf{KDF}(k, \ell_1)$ and $\mathsf{Key}_2 = \mathsf{KDF}(k, \ell_2)$, with $\ell_1 \leq \ell_2$, then Key_1 is a prefix of Key_2, which is also a prefix of $\mathsf{Key} = \mathsf{KDF}(k, L)$, where L is an upper-bound to the bit length of the key material. This leads to a simple chosen-ciphertext attack against the hybrid scheme, that we now describe.

Semantic security under chosen-ciphertext attacks. The attack is depicted on Figure 1. The adversary chooses two different messages M_0 and M_1 of the same length, with their MLen trailing bits are identical

$$M_0 = m_0 \,||\, m' \text{ and } M_1 = m_1 \,||\, m'.$$

The challenge ciphertext $R \,||\, EM \,||\, D$ that it receives uses the key material $\mathsf{Key} = \mathsf{EK} \,||\, \mathsf{MK}$. Parsing EK into $\mathsf{EK} = \mathsf{EK}' \,||\, \mathsf{MK}'$, one gets:

$$EM = M_b \oplus \mathsf{EK} = (m_b \oplus \mathsf{EK}') \oplus (m' \oplus \mathsf{MK}') = EM' \oplus (m' \oplus \mathsf{MK}').$$

From EM, the attacker computes $EM' = m_b \oplus \mathsf{EK}'$, and $m' \oplus \mathsf{MK}'$, which provides MK', since m' is known. Then, it computes the MAC D' of EM' using the MAC key MK'. At this point, the adversary has obtained a valid encryption of m_b, as $R \,||\, EM' \,||\, D'$. Simply querying this new ciphertext to the decryption oracle, the attacker gets m_b, and thus obtains the value of bit b.

The flaw is a consequence of the fact that R does not uniquely define the MAC key MK, since the latter depends on the length of the plaintext as well. If one parses the key material Key another way, $\mathsf{Key} = \mathsf{MK} \,\|\, \mathsf{EK}$, as suggested in the research papers [1, 2], the attack collapses and the scheme can be proven secure.

Non-malleability under chosen-plaintext attacks. Observe that the attack can be seen as breaking non-malleability, even in a chosen-plaintext scenario. From an encryption $R \,\|\, EM \,\|\, D$ of a message $M_0 = m_0 \,\|\, m'$, whose trailing bits are m', the adversary can compute an encryption of m_0, by the method described above.

Conclusion. In order to avoid the attack, it is essential that the MAC key depends on R only. Thus, the version of ECIES in [10] cannot be supported by a security proof in the CCA scenario. There are various ways to modify the scheme and restore CCA security. The simplest is parsing Key as $\mathsf{MK} \,\|\, \mathsf{EK}$.

4.4.5 Proof of theorem 3

We now turn to the the proof of theorem 3.

We consider an attacker $\mathcal{A} = (A_1, A_2)$ and use this adversary as usual.

1. run the key generation algorithm for the key encapsulation scheme
2. next run A_1 on the public data to get a pair of messages $\{M_0, M_1\}$ as well as a state information st. Choose a random bit b, run the key encapsulation scheme to get R and the key material Key, and thus EK and MK, followed by the encryption EM of M_b and the MAC D of EM.
3. run $A_2(R \,\|\, EM \,\|\, D, st)$ and finally get an answer b'. Eventually, output bit $b = b'$.

As in the proof of Theorem 2, we will envision several games:

- game $\mathcal{G}_1$, where the decryption queries are answered by an actual decryption oracle
- game $\mathcal{G}_2$, where the key material to encrypt/MAC the test message M_b is replaced by a random string and where queries whose initial part matches with R are decrypted using the same random string
- game $\mathcal{G}_3$, which is as $\mathcal{G}_2$ but where all queries whose initial part matches with R are rejected

We observe that the probability that $\mathcal{G}_1$ outputs 1 (and thus $b' = b$) is exactly $1/2 + \varepsilon$. Indeed, game $\mathcal{G}_1$ provides the adversary with the real-life setting. As in the proof of Theorem 2, we bound the difference of probabilities between $\mathcal{G}_1$ and $\mathcal{G}_3$.

- In order to bound the difference between $\mathcal{G}_1$ and $\mathcal{G}_2$, observe that both can be played by calling the decryption oracle for key encapsulation rather than the actual decryption oracle. Game $\mathcal{G}_1$ needs as an additional input the key material corresponding to R and, similarly, game $\mathcal{G}_2$ will need the random key material. Thus, we have obtained a distinguisher between the distribution consisting of a ciphertext and the corresponding key material (in game $\mathcal{G}_1$) and the analogous distribution where the key material is replaced by a random string (in game $\mathcal{G}_2$). This bounds the difference by $\mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t', q_D)$.

  In the stream cipher mode, the key material Key has to be long enough to encrypt large messages. When playing by calling the decryption oracle, one sets the length to its maximal value.

- To go from game $\mathcal{G}_2$ to game $\mathcal{G}_3$, we have to bound the probability that $\mathcal{G}_3$ rejects a valid ciphertext. We relate this event to the ability to forge a MAC, by defining a further simulation. This simulation is similar to $\mathcal{G}_3$ with the following differences

  1. the tag D of the challenge ciphertext (R, EM, D) is generated using a single call to the $\mathsf{MAC.Gen}$ oracle
  2. a random index $\kappa \in \{1, \cdots, q_D\}$ is chosen and the tag D' included in the κ-th decryption query, whose initial part matches with R, is returned together with the corresponding encrypted message EM'.

  The simulator returns a MAC forgery (EM', D') with probability upper-bounded by $\mathsf{Succ}^{\mathsf{mac}}(t', 1)$. Note that this exactly means that, in game $\mathcal{G}_3$, the κ-th query (R, EM', D') is a valid ciphertext. Here, we use in a crucial way the fact that the MAC key MK only depends on R, which is our current assumption: since the challenge ciphertext (R, EM, D) and the query (R, EM', D') have the same leading part R, D and D' are computed from EM and EM' respectively, using the same key MK. This allows to bound the difference between the probabilities of games $\mathcal{G}_2$ and $\mathcal{G}_3$ by $q_D \times \mathsf{Succ}^{\mathsf{mac}}(t', 1)$.

  In the more general setting, where the MAC adversary can ask many queries to the oracles, one can replace this bound by $\mathsf{Succ}^{\mathsf{mac}}(t', q_D)$, since one can spot the forgery by oracle calls.

To conclude, consider the probability of that $\mathcal{G}_3$ outputs one. In this game, a random string is drawn as a session key and used to encrypt a randomly chosen test message M_b under this key. The adversary outputs one if he has correctly guessed bit b. This is exactly the situation of a semantic distinguisher as defined in Section 4.4.1. Therefore, the advantage of the adversary in this latter game $\mathcal{G}_3$ is bounded by $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t')$. This concludes the proof.

4.5 Discrepancies between [1], [2], [10], [22]

We are now in a position to further comment on the discrepancies between [1], [2], [10] and [22]. Recall that document [10] derives the key material Key as the image of z by the random oracle H, whereas [2] uses $\mathsf{H}(\mathsf{DH}(Q, R))$, and [1] uses $\mathsf{H}(R, \mathsf{DH}(Q, R))$. Note that the latter is equivalent to our analysis which uses $\mathsf{H}(R, z)$, but introduces a multiplicative factor 2 in the security estimates. The IEEE standard [22] is similar to [10], but allows the option to include R in the key derivation function, by processing it as part of the key derivation parameters. This is for "consistency with [1]", as stated in [22], section **11.3.2**, Note 2. A related Note (section **D5.3**, Note 1), acknowledges that

> leaving out the ephemeral public key also appears to weaken the provable security of the scheme.

Combining Theorems 2 and 3, one gets the following security result which holds for the proposal from [1]):

Theorem 4 *Let $\mathcal{A}$ be a CCA–adversary against the ECIES encryption scheme, with running time bounded by t and advantage ε, making q_D decryption queries and q_H H-oracle calls. Then*

$$\varepsilon \leq \mathsf{Succ}^{\mathsf{GDH}}(\delta, t', 2q_H) + \mathsf{Adv}^{\mathsf{E,D}}(t') + q_D \times \mathsf{Succ}^{\mathsf{mac}}(t', 1) + 2q_H\delta,$$

with $t' \leq t + 2q_H \cdot \tau + \mathcal{O}(q_D)$, where τ denotes the running time of the δ-ECDDH oracle.

The more recent research paper [2] proposes a one-to-one encoding of $\mathsf{DH}(Q, R)$ as the shared secret, and therefore also provides a provably secure scheme. However, the security estimates are not as good, since R is not included in the input to the hash function used for building the key material. The corresponding result reads as follows.

Theorem 5 *Let $\mathcal{A}$ be a CCA–adversary against the ECIES encryption scheme, with running time bounded by t and advantage ε, making q_D decryption queries and q_H H-oracle calls. Then*

$$\varepsilon \leq \mathsf{Succ}^{\mathsf{GDH}}(\delta, t', q_H(q_D + 1)) + \mathsf{Adv}^{\mathsf{E,D}}(t') + q_D \times \mathsf{Succ}^{\mathsf{mac}}(t', 1) + q_H(q_D + 1)\delta,$$

with $t' \leq t + q_H(q_D + 1) \cdot \tau + \mathcal{O}(q_D)$, where τ denotes the running time of the δ-ECDDH oracle.

Proof. We use a similar adversary $\mathcal{B}$ as in the proof of Theorem 2. Since R is not part of the input to the random oracle H, the simulation is less efficient: the simulator has to submit to the ECDDH oracle all pairs (R, P), where P is in the H-List, and R

appears either in the challenge or in the implicit list. This means $(q_D + 1)q_H$ queries altogether. Details are straightforward. □

To conclude, we note that there is a security loss when discarding R from the input to the hash function. We regret that no part of the submission accounts for this security loss.

We furthermore note that the version of ECIES described in [10], and also allowed in [22], is not provably secure in the CCA scenario, since it derives the key from the shared field element only. This is notwithstanding the specific issue of the stream cipher mode.

5 Conclusions

Based on our analysis, we believe that the cryptosystem ECIES is presumably secure, with the proposed parameters, for the foreseeable future. However, we found the security arguments disappointing and, based on the submission, we have the following restrictions:

1. The non-standard assumption on which ECIES relies is exactly the statement that the session key is semantically secure against chosen ciphertext attacks, and therefore does not tell anything significant on the public key part of the scheme. As a consequence, the security of this part can only be termed heuristic.

2. Although we have shown that a more standard security proof is possible, it would have to use random oracles and to rely on the so-called gap problems.

3. Contrary to an earlier version [1], paper [2], has reduced the amount of data used as an input to the hash function that derives the session key. Our analysis seems to indicate that the change duly entails a security loss. However, no part of the submission accounts for this security loss. The submission [10] further reduces this amount of data, simply using the shared field element z to derive the key. As a result, the security proof of IND-CCA collapses, and a chosen-ciphertext attack can be mounted.

4. Contrary to the research papers [1, 2], the submission [10] implements stream cipher mode by computing the MAC key as the trailing bits of the key material. Thus, the MAC key depends on the length of the message. As a result, the security proof of IND-CCA collapses, and a chosen-ciphertext attack can be mounted.

Of course, it can be argued that the flaws spotted by our analysis are benign (see [24]). For example, item 4 above can be adequately answered by allowing stream cipher mode only when messages of constant length are processed. As for item 3, one may note that the version of ECIES from [10] has provable security in a slightly weaker model, recently termed *generalized chosen-ciphertext security* [3], or *benign malleability* [41]. In this model, the adversary has restricted access to the decryption oracle: not only it cannot query the challenge ciphertext, but it cannot either query any ciphertext derived from the challenge by a specific simple rule. In ECIES, this would simply forbid replacing R by $-R$ in the challenge. One can easily adapt Theorem 2 to this context.

Still, we only recommend the scheme with additional provisions.

- The MAC key should be defined from the key encapsulation part R of the ciphertext only. In cases where it may not hold, as allowed by [10] or [22], usage guidelines should prevent using stream cipher mode, when processing data of variable length. Similarly, it should not be possible to dynamically change the key derivation function, since it may entail the same security flaw.
- The key material should be derived from the entire information in the Diffie-Hellman shared key $\mathsf{DH}(Q, R)$. In other words the encoding method producing k from $\mathsf{DH}(Q, R)$ should be one-to-one. In cases where it may not hold, as allowed by [10] or [22], a warning should be included, explaining that the scheme does not meet the strongest possible CCA security model, but a weaker model, sufficient in practice.

References

[1] M. Abdalla, M. Bellare and P. Rogaway. DHAES: An encryption scheme based on the Diffie-Hellman problem. Submission to IEEE P1363, September 1998. `http://grouper.ieee.org/groups/1363/P1363a`

[2] M. Abdalla, M. Bellare and P. Rogaway. The oracle Diffie-Hellman assumption and an analysis of DHIES. In *CT-RSA '01*, LNCS 2020, pages 143–158. Springer-Verlag, Berlin, 2001. See also the full paper on the authors' web page. `http://www.cs.ucsd.edu/users/mihir/papers/dhies.html`

[3] J. H. An, Y. Dodis and T. Rabin. On the security of joint signature and encryption. In *Eurocrypt '02*, LNCS 2332, pages 83–107. Springer-Verlag, Berlin, 2002.

[4] R. Balasubramanian and N. Koblitz. The improbability than an elliptic curve has subexponential discrete log problem under the Menezes-Okamoto-Vanstone algorithm, *J. Cryptology*, 111, (1998), 141–145.

[5] M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among notions of security for public-key encryption schemes. In *Crypto '98*, LNCS 1462, pages 26–45. Springer-Verlag, Berlin, 1998.

[6] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. of the 1st CCS*, pages 62–73. ACM Press, New York, 1993.

[7] M. Bellare and P. Rogaway. Optimal asymmetric encryption – How to encrypt with RSA. In *Eurocrypt '94*, LNCS 950, pages 92–111. Springer-Verlag, Berlin, 1995.

[8] S. Cavallar, B. Dodson, A. K. Lenstra, W. Lioen, P. L. Montgomery, B. Murphy, H. te Riele, K. Aardal, J. Gilchrist, G. Guillerm, P. C. Leyland, J. Marchand, F. Morain, A. Muffett, C. Putnam, C. Putnam, P. Zimmermann. Factorization of a 512-Bit RSA modulus. In *Eurocrypt '00*, LNCS 1807, pages 1–18. Springer-Verlag, Berlin, 2000.

[9] Certicom. Information on the Certicom ECC challenge.
`http://www.certicom.com/research/ecc_challenge.html`

[10] Certicom. Standards for efficient cryptography, SEC1: elliptic curve cryptography. September 2000.

[11] D. Coppersmith, A.M. Odlyzko and R.Schroeppel. Discrete logarithms in $GF(p)$. *Algorithmica 1* (1986), 1–15

[12] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In *Crypto '98*, LNCS 1462, pages 13–25. Springer-Verlag, Berlin, 1998.

[13] J.L. Carter and M.N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18, (1979), 143–154.

[14] D. Dolev, C. Dwork, and M. Naor. Non-malleable cryptography. In *Proc. of the 23rd STOC.* ACM Press, New York, 1991.

[15] T. El Gamal. A public key crtyptosystem and signature scheme based on discrete logarithms. *IEEE Trans. on Inform. theory*, 31 (1985), 469–472.

[16] G. Frey, M. Müller, and H. G. Rück. The Tate-pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Transactions on Information Theory*, 45:1717–1719, 1999.

[17] G. Frey and H. G. Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of Computation*, 62:865–874, 1994.

[18] R. Gallant, R. Lambert and S.A. Vanstone. Improving the parallelized Pollard lambda search on binary anomalous elliptic curves. *Mathematics of Computation*, 69, (2000), 1699–1705.

[19] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science* 28, (1984), 270–299.

[20] Dan Gordon. Discrete logarithms in $GF(p)$ using the number field sieve. *SIAM J. Discrete Math.*, 6, (1993), 124-138.

[21] R. Harley, D. Doligez, D. de Rauglaudre, X. Leroy. Elliptic curve discrete logarithms: ECC2K-108.
`http://cristal.inria.fr/~harley/ecdl7/`

[22] IEEE P1363a/D9 – Standard specifications for public key cryptography: additional techniques, June 2001. `http://grouper.ieee.org/groups/1363/P1363a`

[23] R. Impagliazzo and D. Zuckermann. How to rectcle random bits. In *Proc. of the 30th FOCS*, (1989), 248–253.

[24] D. Johnson. Certicom analysis comparing NESSIE submision od ECIES and Victor Shoup's ECIES-KEM proposal od Sept. 17, 2001.

[25] A. Joux and R. Lercier. Computing a discrete logarithm in $GF(p)$, p a 90 digit prime.
`http://www.medicis.polytechnique.fr/~lercier/english/dlog.html`

[26] A. Joux and R. Lercier. Computing a discrete logarithm in $GF(p)$, p a 100 digit prime.
`http://www.medicis.polytechnique.fr/~lercier/english/dlog.html`

[27] H. Krawczyk. LFSR-based hashing and authentication. In *Crypto '94*, LNCS 839, pages 129–139. Springer-Verlag, Berlin, 1995.

[28] A.K. Lenstra and E. Verheul. Selecting cryptographic key sizes. In *PKC '00*, LNCS 1751, pages 446–465. Springer-Verlag, Berlin, 2000.

[29] R. Lercier and F. Morain. Counting the number of points on elliptic curves over finite fields: strategies and perormances. In *Eurocrypt '95*, LNCS 921, pages 79–94. Springer-Verlag, Berlin, 2000.

[30] C.H Lim and P.J. Lee. A key recovery attack on discrete log based schemes using a prime order subgroup. In *Crypt '97*, LNCS 1294, pages 249–263. Springer-Verlag, Berlin, 1997.

[31] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *Proc. of the 38th FOCS*, (1997), 458–467.

[32] T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *PKC '01*, LNCS 1992, pages 104–118. Springer-Verlag, Berlin, 2001.

[33] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance. *IEEE Transactions on Information Theory*, 24, (1978),106–110.

[34] J. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32, (1978), 918–924.

[35] H.G. Rück. On the discrete logarithm in the divisor class group of curves. Preprint (1997).

[36] T. Satoh and K. Araki. Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves. (1997), to appear in Commentarii Math. Univ. St Pauli.

[37] I.A. Semaev. Evaluation of discrete logarithms in a group of p-torsion points of an elliptic curve of characteristic p. *Math. Comp.*, 67 (1998), 353–356.

[38] O. Schirokauer. Discrete logarithms and local units. *Phil. Trans. R. Soc. Lond.* A 345, (1993), 409–423.

[39] V. Shoup. A composition theorem for universal one-way hash functions. In *Eurocrypt '00*, LNCS 1807, pages 275–288. Springer-Verlag, Berlin, 2000.

[40] V. Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *Eurocrypt '00*, LNCS 1807, pages 445–452. Springer-Verlag, Berlin, 2000.

[41] V. Shoup. A proposal for an ISO standard for public key encryption. Version 2.1, December 2001. `http://www.shoup.net/papers`

[42] V. Shoup and T. Schweinberger. ACE Encrypt: The Advanced Cryptographic Engine' public key encryption scheme. Manuscript, March 2000. Revised, August 14, 2000.

[43] N.P. Smart. The discrete logarithm problem on elliptic curves of trace one. *J. Cryptology*, 12, (1999), 141–151.

[44] P.C. van Oorschot and M. J. Wiener. Parallel collision search with cryptanalytic applications. *J. Cryptology*, 12, (1999), 1–28.

[45] D. Weber, T. F. Denny and J. Zayer.
`http://felix.unife.it/Root/d-Mathematics/d-Number-theory/t-Weber-discrete-logarithm-record-960925`

[46] M. J. Wiener and R.J. Zuccherato. Fast attacks on elliptic curve cryptosystems. In *SAC '98*, LNCS 1556, pages 190–200. Springer-Verlag, Berlin, 1999.