Evaluation Report on the **ECAES Cryptosystem**

1 Introduction

This document is an evaluation of the **ECAES Cryptosystem**. Our work is based on the analysis of document [8], which provides both the specification and self-evaluation of the scheme, as well as on the research paper [1], where additional security arguments can be found. Note that document [8] uses the alternative name **ECIES** for *Elliptic Curve Integrated Encryption Scheme*, while the **A** of **ECAES** stands for *augmented*. 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

ECAES 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.

ECAES is a hybrid encryption scheme built on an adaptation of the El Gamal cryptosystem (see [13]) to the context of elliptic curves. The basic function f on which ECAES 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 ECAES consists of an element of the curve of order n, say Q. The secret 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 secret-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.

Thus, the key generation algorithm $\mathcal{K}(1^m)$ of the ECAES 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 secret key $\mathsf{sk} = d$ is the discrete logarithm of Q in base G.

We now turn to 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 field element z, the first coordinate of a curve element computed as $\mathsf{DH}(Q, R) = r \cdot Q$ (or - alternatively - as $\mathsf{DH}'(Q, R) = h \cdot r \cdot Q$). By "shared", we mean that the recipient of the message is also able to obtain z, as will be seen in the sequel.

From the shared field element z, key material is generated. 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 obtained by means of a hash function H, which is unkeyed and applied to z and - optionally - to publicly shared information. 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 $C = R||EM||D$.

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 [8] 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.

With respect to the scheme presented in [1], there is a slight difference, since the latter includes R in the input to H.

Decryption $\mathcal{D}_{\mathsf{sk}}(C)$ is based on recovering z. Granted the secret key d, it is possible to obtain the shared field element z as the first coordinate of $\mathsf{DH}(Q, R) = d \cdot R$ (or - alternatively - of $\mathsf{DH}'(Q, R) = h \cdot d \cdot R$). Once z 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.

2.2 Comments on the specification

Document [8] 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 [8] is rather sketchy and refers to [1]. The latter is 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, the assumption that is used, which we will later discuss, is certainly contrived. It has been severely criticized by other researchers. We quote [37]:

> 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.

Of course, the opinion comes from a competitor and the present report will aim at clarifying the issue.

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 [8, 1] 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 [27], 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 [26]). 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 secret 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 $\mathcal{A}$.

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 [17]. However, we do not see any argument that would give such guarantee. As mentioned in [1], it is possible to obtain such a bit-string by applying a randomly keyed universal hash function, following the method described in [27] and also used in [37]. Recall that, if H_k is a universal hash function, keyed by k, with ℓ-bit outputs, then, the leftover hash lemma of [20] implies that hashing a set of 2^λ bit-strings produces a distribution $(k, H_k(x))$, whose distance to the uniform distribution is $\leq \frac{1}{2^{(\lambda-\ell)/2}}$. 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 [30] which takes computing time equivalent to about $\sqrt{\pi n/2}$ group operations. In 1993, van Oorschot and Wiener in [39], 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 [16, 41]), 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 [7] led to the computation of a discrete logarithm in a group with 2^{109} elements (see [19]). 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 [6]) 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 [24], 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 ([33]), Rück ([31]), Smart ([38]) and Satoh-Araki ([32]) 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 [2] 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 [15] (see also [14]) 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

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

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 ECAES 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 [25]. Considering that the appropriate warnings are given, this does not constitute a strong objection to the proposal under review.

4 Security Analysis

Document [8] includes 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 stand for *indistinguishable*, if for any adversary $\mathcal{A} = (A_1, A_2)$ with running time bounded by t, the advantage

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

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* [12]. 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 [3] 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 [37], 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 $H(z)$ as the answer, where $H(z)$ is computed from (G, Q, R) as prescribed by the cryptosystem. Unfortunately, no such proof appears possible: the difficulty lies in the simulation of the decryption queries. Before we explain how documents [8, 1] solve the matter, we turn to the random oracle model, deriving the key material as the image of z by the random oracle H, as suggested in document [8], rather than $H(R, z)$, as in [1]. Again, it does not appear possible to write up a proof by lack of a correct simulator: when receiving a query R, the simulator should provide $H(z)$, 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 [28]).

4.2.1 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 2.1 and 3.1.1.

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

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

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 ECAES 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.

4.2.2 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
$$\mathcal{D}_0 = (G, Q, R, H(z))$$
generated by the cryptosystem
2. or else from the analogous distribution $\mathcal{D}_1$ where z is replaced by a random element of $\mathcal{X}$ (notations are those from section 3.1.2).

$\mathcal{A}$ is allowed to query a decryption oracle by submitting triples (G, Q, 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.

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 oracle calls. Then*

$$\begin{aligned} \frac{\varepsilon}{2} &\leq \mathsf{Succ}^{\mathsf{GDH}}(\delta, t', 2 \cdot q_H \cdot (q_D + 1)) + 2 \cdot (q_D + 1) \cdot q_H \cdot \delta \\ \textit{where } t' &\leq t + 2 \cdot (q_D + 1) \cdot q_H \cdot \tau \end{aligned}$$

and where τ denotes the running time of the δ-ECDDH oracle.

Proof: It will be convenient to allow implicit calls to the oracle H. This means submitting an elliptic curve point R and receiving $H(z)$, 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 secret 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 $(G, Q, R, H(z))$ 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 $\mathsf{DH}_G(Q, R)$, 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 **H-list**, a list of implicit calls together with their answers. For each fresh query 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 $(G, Q, \tilde{R}, P_i)$, where $\tilde{R}$ ranges over all queries currently in the implicit list. As soon as it finds an element such that the answer of the δ-ECDDH oracle is positive, $\mathcal{B}$ returns the corresponding value $H(u)$. If none is found, the oracle picks a random answer and adds it to the **H-list**. Fresh implicit calls at $\tilde{R}$ are handled similarly. For each string u appearing 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 $(G, Q, \tilde{R}, P_i)$. As soon as it finds an element such that the answer of the δ-ECDDH oracle is positive, $\mathcal{B}$ returns the corresponding value $H(u)$. 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 (G, Q, R') to the decryption oracle, $\mathcal{B}$ 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** and the implicit list, $\mathcal{B}$ returns 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 $2q_H \cdot (q_D + 1)$ times.

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 $\frac{1}{2}(\theta_1 + (1 - \theta_0))$, 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 $1/2 + \frac{\varepsilon}{2}$. We relate the probabilities of the same event in all games, repeatedly using the simple yet useful lemma from [37]

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.

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 x-coordinate z of 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 $\mathsf{Succ}^{\mathsf{GDH}}(\delta, t', 2 \cdot q_H \cdot (q_D + 1))$ that $\mathcal{B}$ has correctly computed $\mathsf{DH}(Q, R)$. 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 $2(q_D + 1)q_H$ times, the failure probability is bounded by $2(q_D + 1)q_H\delta$.

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

$$\mathsf{Succ}^{\mathsf{GDH}}(\delta, t', 2 \cdot q_H \cdot (q_D + 1)) + 2 \cdot (q_D + 1) \cdot q_H \cdot \delta$$

Note that game $\mathcal{G}_5$ runs within time bound $t' \leq t + 2(q_D + 1)q_H\tau$, 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 the x-coordinate z of 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.

4.3 The intractability hypothesis of [8]

Documents [8, 1] 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 $(G, Q, R, H(z))$, computed as prescribed by the cryptosystem, even calling an oracle which returns $H(z')$, when queried at $R' \neq R$. Thus, this assumption is **exactly** the statement that key encapsulation is secure.

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 [8], is secure provided the symmetric encryption scheme and the MAC scheme in use are secure. Thus, as suggested in [37], it seems actually that the security proof of [1] tells nothing significant on the public key part.

We find that the arguments in [8] are even somehow misleading in terms of security. The proof that was carried in the previous section involves a security loss $2(q_D+1)q_H\delta$. This can be decreased to $2q_H\delta$ if the random oracle is fed with (R, z) in place of just z. This would also decrease the computational cost of the reduction by a factor q_D. The

arguments in [8], claiming that the security analysis is the same, regardless of whether or not R is an input to the hash function, do not account for this security loss.

4.4 From key encapsulation to chosen ciphertext security

Following [36], 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 secret-public keys $(\mathsf{sk}, \mathsf{pk})$, relatively to a security parameter m
- the encapsulation algorithm $\mathcal{E}_{\mathsf{pk}}(r)$ outputs a ciphertext C using random coins r
- the decryption algorithm $\mathcal{D}_{\mathsf{sk}}(C)$ outputs the key material k associated to the ciphertext C (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, one can derive an hybrid encryption scheme exactly as in [1]. 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*

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

and 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.

Before going further in the proof, let us first define more formally the security notions for the symmetric encryption and the MAC schemes.

4.4.1 Symmetric Encryption

A symmetric encryption scheme with a key-length k, on messages of length ℓ, consists of two algorithms (E, D) which depend on a k-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* [17]: a symmetric encryption scheme is *semantically secure* if no polynomial-time 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

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

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.

4.4.2 Message Authentication Code

A Message Authentication Code (a.k.a. MAC) with key-length k consists of two algorithms $(\mathsf{MAC.Gen}, \mathsf{MAC.Ver})$, which depend on a k-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 secret key, even if it has access to MAC oracle 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

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

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 $\mathsf{MAC.Gen_k}(\cdot)$.

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 [11] meet the corresponding security criterion, without any cryptograhic assumption.

4.4.3 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 C and the key material, followed by the encryption EM of M_b and the MAC D of EM, using the derived key material.
3. run $A_2(C||EM||D, st)$ 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 C 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 C 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 C 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 $\mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t', q_D)$.

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 (C, EM, D) is generated using a single call to the 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 C, is returned together with the corresponding encrypted message EM'.

The simulator returns a MAC forgery (EM, D) with probability at most $\mathsf{Succ}^{\mathsf{mac}}(t', 1)$. Note that this exactly means that, in game $\mathcal{G}_3$, the κ-th query (C, EM', D') is a valid ciphertext. 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 The discrepancy between [1] and [8]

We are now in a position to further comment on the discrepancy between [1] and [8]. Recall that document [8] derives the key material as the image of z by the random oracle H, whereas [1] uses $H(R, z)$.

Combining theorems 2 and 3, one gets the following:

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

$$\begin{aligned} \varepsilon &\leq \mathsf{Succ}^{\mathsf{GDH}}(\delta, t', 2q_H(q_D+1)) + \mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t') + q_D \times \mathsf{Succ}^{\mathsf{mac}}(t', 1) \\ &\quad + 2q_H(q_D+1)\delta \\ \textit{where } t' &\leq t + 2q_H(q_D+1)\cdot\tau + \mathcal{O}(q_D) \end{aligned}$$

and where τ denotes the running time of the δ-ECDDH oracle.

The estimates in the previous security result would have been considerably better deriving the key material from (R, z). Indeed, the resulting modified key encapsulation scheme satisfies the following:

Theorem 5 *Let $\mathcal{A}$ be a CCA–adversary against the modified key encapsulation scheme with running time $\leq t$ and advantage ε, making q_D decryption queries and q_H oracle calls. Then*

$$\begin{aligned} \varepsilon &\leq \mathsf{Succ}^{\mathsf{GDH}}(\delta, t', 2 \cdot q_H) + 2q_H \cdot \delta \\ \textit{where } t' &\leq t + 2q_H \cdot \tau \end{aligned}$$

Proof: We use a similar adversary $\mathcal{B}$ as in the proof of theorem 2, but with a more efficient simulation. This simulation still maintains two lists, the **H-list** and the list of implicit queries, but the workload to ensure consistency decreases. This is because a query to H of the form $(\tilde{R}, u)$ can only conflict with an implicit query at $\tilde{R}$. To check consistency, one just needs to call the δ-ECDDH oracle at $(G, Q, \tilde{R}, P_i)$, where P_1 and P_2 are the two elliptic curve points P_1, P_2, whose first coordinate is u (if they exist). Altogether, this produces $2q_H$ queries to the δ-ECDDH oracle. Details are straightforward.

Thus, 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.

5 Conclusions

Based on our analysis, we believe that the cryptosystem ECAES is presumably secure, with the proposed parameters, for the foreseeable future. However, based on the submission, we have the following restrictions:

- The non-standard assumption on which ECAES 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.
- 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.
- Contrary to earlier versions, the submission 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.

References

[1] M. Abdalla, M. Bellare and P. Rogaway. DHAES: An encryption scheme based on the Diffie-Hellman problem, `http://www-cse.ucsd.edu/users/mihir`

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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. Eurocrypt'2000, Lecture Notes in Computer Science 1807,(2000), 1–18.

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

[8] Certicom, Standards for efficient cryptography, SEC1: elliptic curve cryptography, sept, 20, 2000.

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

[10] R. Cramer and V. Shoup, A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. Crypto'98, Lecture Notes in Computer Science 1462, (1998), 13–25.

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

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

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

[14] 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.

[15] 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.

[16] 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.

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

[18] Dan Gordon, Discrete Logarithms in $GF(p)$ using the Number Field Sieve, *SIAM J. Discrete Math.*, 6, (1993), 124-138.

[19] R. Harley, D. Doligez, D. de Rauglaudre, X. Leroy, Elliptic Curve Discrete Logarithms: ECC2K-108,
`http://cristal.inria.fr/~harley/ecdl7/`

[20] R. Impagliazzo and D. Zuckermann, How to rectcle random bits, *30th annual symposium on foundations of computer science*, (1989), 248–253.

[21] 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`

[22] 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`

[23] H. Krawczyk, LFSR-based hashing and authentication, Crypto'94, Lecture Notes in Computer Science 839, (1995), 129–139.

[24] A.K. Lenstra and E. Verheul, Selecting cryptographic key sizes, PKC'2000, Lecture Notes in Computer Science 1751, (2000), 446–465.

[25] R. Lercier and F. Morain, Counting the number of points on elliptic curves over finite fields: strategies and perormances, Eurocrypt' 95, Lecture Notes in Computer Science 921, (1995), 79–94.

[26] C.H Lim and P.J. Lee. A key recovery attack on discrete log based schemes using a prime order subgroup, Crypto '97, Lecture Notes in Computer Science 1294, (1997), 249–263.

[27] M. Naor, O. Reingold, Number-theoretic Constructions of Efficient Pseudo-random Functions, *38-th annual symposium on foundations of computer science*, (1997), 458–467.

[28] T. Okamoto and D. Pointcheval. The Gap-Problems: a New Class of Problems for the Security of Cryptographic Schemes. In *PKC '2001*, LNCS. Springer-Verlag, Berlin, 2001.

[29] 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.

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

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

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

[33] 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.

[34] Oliver Schirokauer, Discrete Logarithms and Local Units, *Phil. Trans. R. Soc. Lond.* A 345, (1993), 409–423.

[35] V. Shoup, A composition theorem for universal one-way hash functions, Eurocrypt'2000, Lecture Notes in Computer Science 1807, (2000), 275–288.

[36] V. Shoup, Using hash functions as a hedge against chosen ciphertext attack, Eurocrypt'2000, Lecture Notes in Computer Science 1807, (2000), 445–452.

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

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

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

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

[41] M. J. Wiener and R.J. Zuccherato. Fast attacks on elliptic curve cryptosystems, SAC'98, Lecture Notes in Computer Science 1556, (1999), 190–200.