Evaluation Report on the **PSEC Cryptosystem**

1 Introduction

This document is an evaluation of the **PSEC Cryptosystem**. Our work is based on the analysis of documents [28, 29], which provide both the specification and self-evaluation of the scheme, as well as on various research paper such as [14, 15, 16, 25, 26], where additional security arguments can be found. 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

PSEC is based on the hardness of the discrete logarithm problem over an elliptic curve. The cryptosystem uses elliptic curves E over some field $\mathbb{F}_q$, where q is of the form q_0^n. Usual cases are those where q is prime or a power of two, but the cryptosystem does not exclude further cases. Once the curve E has been chosen, a base point P is chosen on E, which generates a subgroup of prime order p.

PSEC uses as a building block an adaptation of the El Gamal cryptosystem (see [11]) to the context of elliptic curves. The basic function f on which PSEC is based is defined by

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

where $r \cdot P$ is obtained, by means of the usual elliptic curve addition, as the sum of r times P. Inverting f is precisely the elliptic curve discrete logarithm problem

(ECDLP). Clearly, f is one-to-one. The inverse function, denoted $\log_P$, is believed to be hard to compute. Another function used by the scheme is the Diffie-Hellman function and denoted by DH_P, where the subscript P will be omitted when clear from the context:

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

Besides the curve parameters, the public key of PSEC consists of an element of the curve of order p, say W. The secret key s is the discrete logarithm of W in base P.

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 $\mathcal{K}(1^k)$ outputs a random pair of secret-public keys $(\mathsf{sk}, \mathsf{pk})$, relatively to a security parameter k
- 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 $\mathcal{K}(1^k)$ of the PSEC Cryptosystem produces, on input k, a public key pk consisting of a curve E a base point P, its order p, which is a prime number with k bits and an element W of order p. In the case of elliptic curves over a field $\mathbb{F}_q$ of characteristic $\neq 2, 3$, the equation of the curve can be taken as

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

and the curve parameters form a tuple $(\mathbb{F}_q, a, b, P, p)$. The public key is given by this tuple plus the public element W. In the case of elliptic curves over a field $\mathbb{F}_q$ of characteristic 2, the equation of the curve can be taken as

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

and the curve parameters form a tuple $(\mathbb{F}_q, a, b, P, p)$. The public key is similarly given by this tuple plus the public element W. For fields of characteristic 3, there is no such "reduced form" for the equation, but the submission seems to implicitly discard this case. When $\mathbb{F}_q$ is defined as an extension field of some smaller $\mathbb{F}_{q_0}$, where q_0 is prime, $\mathbb{F}_q$ is presented as a tuple (q_0, n, f, b), where f is an irreducible polynomial of degree n over $\mathbb{F}_{q_0}$, which defines the extension field $\mathbb{F}_{2^n}$, and b is a bit indicating whether a regular basis or a normal basis is used. In all cases, the secret key $\mathsf{sk} = s$ is the discrete logarithm of W in base P.

It should be added that there are additional items in the public key, consisting of hash function identifiers, which we ignore at this point. We will bring more precision on these elements at a later stage of our report. Similarly, there are further integers to indicate the respective bit-lengths of various data.

The encryption function $\mathcal{E}_{\mathsf{pk}}(m;r)$ of the *primitive encryption function* of PSEC first produces an ephemeral key pair (r, C_1), where r is randomly chosen in $\{0, 2^{\ell_r} - 1\}$ and $C_1 = r \cdot P$. Next it computes a field element z, the first coordinate of a curve element computed as $\mathsf{DH}(W, C_1) = r \cdot W$. Finally, it encrypts an ℓ_m-bit message m as (C_1, c_2), with $c_2 = m \oplus z$. This requires $\ell_m \leq |q|$.

At this point, we wish to note that we have not been too careful with notations in the above description. For example, writing $m \oplus z$ treats z as a bit string or a byte string, whereas it is actually a field element. Document[28] provides the adequate conversion routines. 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.

Decryption $\mathcal{D}_{\mathsf{sk}}(C_1, c_2)$ is based on recovering z. Granted the secret key s, it is possible to obtain the field element z as the first coordinate of $\mathsf{DH}(W, C_1) = s \cdot C_1$. Once z has been retrieved, one can compute m as $c_2 \oplus z$.

We now turn to the actual PSEC scheme. At this point, we should mention that PSEC actually includes three different schemes.

- PSEC-1 is a public key encryption scheme
- PSEC-2 and PSEC-3 are hybrid encryption schemes

We first describe PSEC-1. PSEC-1 uses a hash function H, whose inputs are $\ell_r + \ell_m$-bit long and whose outputs are ℓ_h-bit long. Therefore, the public key includes a reference to the identifier of the hash function. Document [28] requires $\ell_r + \ell_m \leq |q|$ and $\ell_h \leq |p| = k$. Encryption $\mathcal{E}_{\mathsf{pk}}(m;r)$ computes C_1 as $t \cdot P$, with $t = H(m||r)$. Next, it defines a field element z, the first coordinate of a curve element computed as $\mathsf{DH}(W, C_1) = t \cdot W$. Finally, it encrypts an ℓ_m-bit message m as (C_1, c_2), with $c_2 = (m||r) \oplus z$.

Decryption $\mathcal{D}_{\mathsf{sk}}(C_1, c_2)$ is based on recovering z. Granted the secret key s, it is possible to obtain the field element z as the first coordinate of $\mathsf{DH}(W, C_1) = s \cdot C_1$. Once z has been retrieved, one can compute $m||r$ by extracting the $\ell_r + \ell_m$ leading bits of $c_2 \oplus z$. Before outputting m, the decryption algorithm checks that C_1 has been properly computed from $H(m||r)$.

PSEC-2 uses two hash functions H and G and a symmetric encryption scheme E. Accordingly, the public key includes a reference to the identifiers for the hash functions and encryption scheme and an indication of the length of the various inputs and outputs. Document [28] requires $\ell_r \leq |q|$. Encryption $\mathcal{E}_{\mathsf{pk}}(m;r)$ computes C_1 as $t \cdot P$, with $t = H(m||r)$. Next, it defines a field element z, the first coordinate of a curve element computed as $\mathsf{DH}(W, C_1) = t \cdot W$ and computes $c_2 = r \oplus z$. From r, a key k for the symmetric encryption scheme E is derived as $\mathsf{k} = G(r)$. Using this key, a symmetric encryption step produces a cryptogram $c_3 = E_{\mathsf{k}}(m)$. The final ciphertext C is the triple (C_1, c_2, c_3).

Decryption $\mathcal{D}_{\mathsf{sk}}(C)$ is based on recovering k. Granted the secret key s, it is possible to obtain the field element z as the first coordinate of $\mathsf{DH}(W, C_1) = s \cdot C_1$. Once z has been retrieved, one can compute r by extracting the ℓ_r leading bits of $c_2 \oplus z$. From r, k is computed and, once k has been retrieved, one can obtain the plaintext m by applying the decryption algorithm of the symmetric encryption scheme. Before outputting m, the decryption algorithm checks that C_1 has been properly computed from $H(m||r)$.

PSEC-3 is similar to PSEC-2 but produces a ciphertext C with four components $C = (C_1, c_2, c_3, c_4)$. Encryption uses an additional random element u from $\{0,1\}^{|q|}$ and picks r in $\mathbb{Z}_p^\star$. Algorithm $\mathcal{E}_{\mathsf{pk}}(M; r, u)$ computes C_1 as $r \cdot P$. Next, it defines a field element z, the first coordinate of a curve element computed as $\mathsf{DH}(W, C_1) = r \cdot W$ and computes $c_2 = u \oplus z$. From u, a key k for the symmetric encryption scheme E is derived as $\mathsf{k} = G(u)$. Using this key, a symmetric encryption step produces a cryptogram $c_3 = E_{\mathsf{k}}(m)$. Finally, a hash value $c_4 = H(C_1||c_2||c_3||u||m)$ is computed and the ciphertext C is the tuple (C_1, c_2, c_3, c_4).

Decryption $\mathcal{D}_{\mathsf{sk}}(C)$ is based on recovering k. Granted the secret key s, it is possible to obtain the field element z as the first coordinate of $\mathsf{DH}(W, C_1) = s \cdot C_1$. Once z has been retrieved, one can compute u as $c_2 \oplus z$. From u, k is computed and, once k has been retrieved, one can obtain the plaintext m by applying the decryption algorithm of the symmetric encryption scheme. Before outputting m, the decryption algorithm checks that the hash value c_4 is correct.

2.2 Comments on the specification

Document [28] needs further editing. There are ambiguities. For example, notation p is used in place of q_0 on page 3, lines 12 and 13; notation h appears out of context on page 4, line 1 from bottom; similarly for notation y on page 5, line 9 and for notation R on page 6, line 4 from bottom. Also, several parameters such as *rlen* do not appear in the public key on page 5, line 3. On the contrary, *rlen* appears in the public key of PSEC-3, on page 10, line 16, whereas it is not needed since r is chosen at random in $\mathbb{Z}_p^\star$.

The description of the key generation algorithm is vague: no indication is given on how the elliptic curve is generated, on how point counting is achieved, on what additional checks are done at curve generation etc. Document [28] simply refers to the IEEE standards (see [20]).

There is no reference to the literature regarding the version of the El Gamal scheme that is termed *primitive encryption function* in document [28]. This is especially worrying, as it is known that El Gamal encryption has subtle flaws, when the message space does not coincide with the subgroup in which the DH function is computed. We will return to this feature further on.

Since there are three versions of PSEC, users might be a bit puzzled. No indication is given of the respective situations where one scheme would be more appropriate than the other. What can be observed, and clearly appears in the appendix of [29], is that PSEC-1 and PSEC-2 include reencrypting as a part of decryption, which is a rather undesirable feature in terms of efficiency. This is not a drawback in terms of security unless there is no check whatsoever. In this case, it is possible to find s in a chosen-ciphertext attack. We describe the attack for PSEC-1 with curves over a prime fields, but the attack applies in other contexts. One simply submits a ciphertext (V, c) computed from a point V outside the curve and a random c. Such point lies on another curve E' with equation

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

and V can be chosen in such a way that the ECDL is easy on E'. Now, submitting (V, c) allows to get the leading ℓ_m bits of the first coordinate of sV. This is because the computation of $s \cdot V$ does not use the value of b. Similarly, submitting $(2V, c')$ allows to get an approximation of the first coordinate of $2sV$. Now we can write four polynomials coming from the equation of E' at sV, $2sV$, and from the doubling formulas. Eliminating the y coordinates, we are left with a polynomial relation between the x coordinates, together with an approximation of the two unknowns. If the approximation is good enough, methods of [8], allow to recover these unknowns and therefore sV. Using the discrete logarithm in E', one gets s.

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 [28, 29] relate the security of the scheme to 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

$$\mathbf{R_E} = \{(P, W, Q, R)\}$$

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

$$\mathbf{D_E} = \{(P, W, Q, R)\}$$

with $\log_P(Q) = \log_W(R)$. 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 (P, W, Q, R) such that $P \neq W$ 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 [24], 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 $xy \cdot P$ from P, $x \cdot P$ and $y \cdot P$, while the latter states that it is hard to compute x from P and $x \cdot P$. 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 [23]). In the present context, such attacks are not addressed by performing checks to verify e.g. that elements that are claimed to belong to the subgroup generated by P, actually lie within this subgroup. Nevertheless, PSEC-1 and PSEC-2 perform an implicit check by reencrypting the data while PSEC-3 performs an analogous check, by means of the final tag c_4. The security proof that will be provided in the present report, will help clarifying whether this is enough.

3.1.2 Security of the field element

In this section, we consider the simplified version of the primitive encryption function that simply establishes the 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. With the notations of section 2.1, z is the first coordinate of $R = \mathsf{DH}(C_1, W)$. 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, an element x of the underlying field is not necessarily the first coordinate of a point of order p 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*

$$(P, W, C_1, z)$$

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 data manipulations and is negligible.

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

$$\mathbf{R} = \{(P, W, Q, R)\}$$

with P, W, Q and R at random in the subgroup of order p of E

$$\mathbf{D} = \{(P, W, Q, R)\}$$

with $R = \mathsf{DH}(W, Q)$ We run the key generation algorithm and generate an elliptic curve E together with a random element of large prime order p. 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 W. This implicitly defines a secret key. Next we assume that a public key encryption occurs with an ephemeral key, whose public part is Q. Finally, we submit (P, W, Q, z) to $\mathcal{A}$, where z is the first coordinate of R. 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 manipulations needed to compute the data to be handled to $\mathcal{A}$.

Remarks.

1. The above proof implicitly assumes that C_1 is uniformly distributed in the subgroup generated by P, when produced by the cryptosytem. Unfortunately, the formatting of data that is adopted in [28], potentially makes the assumption incorrect. Taking ℓ_r much larger than $|p| = k$ is a sufficient condition. Such condition does not appear in document [28] and we do not see how to otherwise solve the matter.

2. It would be desirable to ensure that one gets a bit-string indistinguishable from a random string with $|q|$ bits, which would mean that the information z is semantically secure in the sense of the seminal paper [18]. However, we do not see any argument that would give such guarantee. It is possible to obtain such a bit-string by applying a randomly keyed universal hash function, following the method described in [24] and also used in [33]. Recall that, if H_k is a universal hash function, keyed by k, with ℓ-bit outputs, then, the leftover hash lemma of [21] 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 exactly the security parameter k. Thus in order to get a bound at most $1/2^{128}$ and to obtain a 128 bit encryption key, one would need $|q| \geq 384$. In any case, this is not the path followed by the submission.

Building on the second remark, we see that the primitive encryption function is not semantically secure. Let m_0, m_1 be chosen at random. Let (C_1, c_2) be a ciphertext of m_b. Observe that, with probability close to $1/2$, $m_{1-b} \oplus c_2$ is not in $\mathcal{X}$. Since $m_b \oplus c_2$ is, we can obtain the proper value of b. When $m_{1-b} \oplus c_2$ is in $\mathcal{X}$, we simply guess at random. We have thus built a adversary that correctly guesses b with probability $3/4$. This contradicts semantic security.

3.1.3 Inverting the primitive encryption function

In this section, we wish to relate the ability to invert the primitive encryption function of PSEC to the ECCDH. We let $g(m, r)$ the result of the primitive encryption algorithm $\mathcal{E}_{\mathsf{pk}}(m; r)$. We prove the following:

Theorem 2 *If a machine $\mathcal{A}$ is able to compute x from $g(x, r)$, $x \in \{0,1\}^{\ell_m}$, $r \leq 2^{\ell_r}$, with probability ε, within time bound t, there exists a machine $\mathcal{B}$ that is able to compute the DH function with probability $\varepsilon' \geq \frac{\varepsilon}{2}\frac{2^{\ell_r}}{p}\frac{2^{\ell_m}}{2^{|q|}}$, within time bound $t' \leq t + \tau$, where the overhead τ accounts for performing computations over $\mathbb{F}_q$ and is bounded by $\mathcal{O}(|q|^3)$.*

Proof: Let us consider an adversary $\mathcal{A}$ able to compute x from $y = f(x, r)$ with probability ε, within time bound t:

$$\mathsf{Succ}^{\mathsf{ow}}(\mathcal{A}) = \Pr[x \overset{R}{\leftarrow} \{0,1\}^{\ell_m}, y \leftarrow f(x,r), z \leftarrow \mathcal{A}(y) : z = x] > \varepsilon$$

where probabilities include r as well as the internal random tape of the probabilistic machine $\mathcal{A}$.

let P be given and let (W, Q) an input to the DH function in base P. We consider P as a base point for the cryptosystem and W as a public key. This implicitly defines a secret key. We also consider Q as an ephemeral public key generated from a random string r and we pick a random $|q|$-bit string c at random. We handle (Q, c) to $\mathcal{A}$. Now, this is a valid ciphertext if:

- r is $< 2^{\ell_r}$
- $z \oplus c$ is at most ℓ_m-bit long

where z is the first coordinate of $\mathsf{DH}(W, Q)$. The first inequality holds with probability $\frac{2^{\ell_r}}{p}$ and the second with probability $\frac{2^{\ell_m}}{2^{\lceil q \rceil}}$.

When receiving a correct ciphertext, $\mathcal{A}$ returns the x coordinate of $\mathsf{DH}(W, Q)$, with probability ε. We let $\mathcal{B}$ return at random one of the two elliptic curve points with this coordinate. Altogether, the probability that $\mathcal{B}$ correctly computes the DH function is at least $\frac{\varepsilon}{2} \frac{2^{\ell_r}}{p} \frac{2^{\ell_m}}{2^{\lceil q \rceil}}$

From the above, we see that $\frac{p}{2^{\ell_r}}$ and $\frac{2^{\ell_m}}{2^{\lceil q \rceil}}$ should not be small. These requirements are not stated in [28], even if the results quoted in the self-evaluation report [29] duly consider cases where they hold.

3.2 Size of the parameters

As was just observed the security of the scheme appears closely related to the ECDDH for elliptic curves. 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 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 [27] which takes computing time equivalent to about $\sqrt{\pi n/2}$

group operations. In 1993, van Oorschot and Wiener in [35], 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 representation $X_i = a_i \cdot G + b_i \cdot Y$ 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 [17, 36]), it was shown how to improve the above by a multiplicative factor $\sqrt{2}$. This takes advantage of the fact that on 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 [22], one can infer that the minimum proposed parameters (160 bits) will presumably guarantee security for at least twenty or twenty-five years. Although this is quite reasonable, we slightly regret that no other figure is proposed.

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 $\mathbb{F}_{q^j}$ of $\mathbb{F}_q$, with small j.

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 ([32]), Rück ([30]), Smart ([34]) and Satoh-Araki ([31]) has shown how to solve the p-part of the DLP in polynomial time.

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 p, n is necessarily a divisor of $q^j - 1$. Recently, Balasubramanian and Koblitz have shown in [2] that this condition was sufficient to carry the MOV reduction.

Another reduction similar to the MOV reduction has appeared in the literature. It is due to Frey and Rück [13] (see also [12]) 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 $\mathbb{F}^\star_{q^j}$. In the case of elliptic curves considered in the current context, the above is applicable only if the order p of the base point is a divisor of $q^j - 1$.

The key generation algorithm does not specifically address the question of weak curves, which one may regret. However, one should note that, based on avoiding small values of j that make $q^j = 1 \bmod p$, there are simple ways to perform checks that eliminate such curves.

3.4 Conclusion

Document [28] does not provide a range of parameters for the cryptosystem. It simply mentions that k should be at least 160. Although this yields a quite reasonable "lifetime", further parameters might be suggested. The main concern with the cryptosystem is its nonstandard use of the first coordinate of a curve element as a mask. This appears to contradict the semantic security of the primitive encryption function on which the scheme is based.

4 Security Analysis of PSEC-1

Document [28] only includes security statements and points to the literature, notably [14] for details.

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* [10]. 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 PSEC-1

We turn to the security analysis. The self-evaluation report [29] claims that PSEC-1 is semantically secure against CCA adversaries. It does not include any proof but makes the following statement about the security results:

> They are easily obtained from results presented in [14, 15].

This is problematic: on one hand, the specification document [28] is a bit vague on the requirements that one should put on the bit-size of several data, and on the other hand, we have shown above that the primitive encryption function of [28] is not semantically secure, contrary to what could be expected. Thus, it appears necessary to perform a quick but complete review of the entire security analysis.

Document [28] claims that PSEC-1 is IND-CCA in the random oracle model, based on the ECDDH. This can be based on the following exact security result.

Theorem 3 *Let $\mathcal{A}$ be a CCA–adversary attacking $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, within time bound t, with advantage ε, making q_D and q_H queries to the decryption oracle and the hash function, respectively. Then there exists an adversary $\mathcal{B}$ attacking the semantic security of the primitive encryption function with advantage ε' and within time bound t', where*

$$\begin{aligned} \varepsilon' &\geq \varepsilon - \frac{2q_H}{2^{\ell_r}} - \frac{q_D}{p} \\ t' &\leq t + q_H \cdot \tau \end{aligned}$$

where ℓ_r is the bisize of r, and τ is the time needed to execute the encryption algorithm and is bounded by $\mathcal{O}(|q|^3)$

The scheme can actually be proven "plaintext–aware" [5, 3], which is claimed to imply chosen-ciphertext security. To prove the above, we turn $\mathcal{A}$ into an attacker for the primitive encryption function. This requires simulating the oracle and describing a plaintext-extractor.

4.2.1 Description of the reduction

Let $\mathcal{A} = (A_1, A_2)$ be an adversary against the semantic security of $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. The description of $\mathcal{B}$ is as follows:

1. $\mathcal{B}$ first runs $\mathcal{K}(1^k)$ to obtain the public key
2. next, $\mathcal{B}$ runs A_1 on the public data, and gets a pair of messages $\{m_0, m_1\}$ as well as a state information st. It chooses two random strings r_0, r_1 of the appropriate length, then flips a random bit b and then defines $C \leftarrow y$, to be a ciphertext of $(m_b || r_b)$
3. $\mathcal{B}$ runs $A_2(C, st)$ and returns the answer b' of $A_2(C, st)$.

Of course, during the entire simulation, $\mathcal{B}$ also has to simulate answers from the random oracle. This is done using the **H-list**, a dynamic data structure consisting of all queries to the random oracle H together with the respective answers. For a fresh query h to H, i.e. a query not on the **H-list**, the oracle picks a random answer. Finally, we define the plaintext-extractor as follows:

1. on a query C' to the decryption oracle, $\mathcal{B}$ looks at the **H-list**. If there is an element on this list which is parsed as $(m'||r', H(m'||r'))$, if it is of the appropriate bit-length and if $\mathcal{E}_{\mathsf{pk}}(m'; r') = C'$, then $\mathcal{B}$ outputs m' as the requested plaintext
2. Otherwise, $\mathcal{B}$ rejects the ciphertext

4.2.2 Probabilistic analysis

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
2. game $\mathcal{G}_2$, which is as $\mathcal{G}_1$ but aborts without calling the random oracle, whenever some $(m_i||r_i)$ appears as a fresh query. In this case the output b' is set at $b' = i$. The decryption queries are still answered by the actual decryption oracle.
3. game $\mathcal{G}_3$, where the random oracle is simulated
4. game $\mathcal{G}_4$, which is as $\mathcal{G}_3$ but where the decryption queries are answered by the plaintext-extractor

We observe that the probability that $b' = b$ in game $\mathcal{G}_1$ is exactly the advantage ε of $\mathcal{A}$. We relate the probabilities of the same event in all games repeatedly using the simple yet useful lemma from [33]

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.

Game $\mathcal{G}_2$ differs from $\mathcal{G}_1$ if it aborts. Observe that the input C is a ciphertext of $m_b||r_b$ but this leaves the choice of r_{1-b} random. Thus, r_{1-b} can appear in a query to

H only with probability $\frac{q_H}{2^{\ell_r}}$. If this does not happen, then a query of the form $m_i||r_i$ is such that $i = b$. Thus the advantage of $\mathcal{G}_2$ exceeds $\varepsilon - \frac{q_H}{2^{\ell_r}}$.

The simulation in game $\mathcal{G}_3$ is perfect unless it conflicts with the implicit constraint coming from the assumption that C is a ciphertext of $(m_b||r_b)$. In the find stage, r_b has not been chosen and this happens, for each oracle query, with probability $\leq \frac{1}{2^{\ell_r}}$. In the guess stage, this cannot happen since $\mathcal{G}_2$ has earlier aborted. Altogether, we discard a set of probability $\leq \frac{q_H}{2^{\ell_r}}$. Applying the lemma, we see that the probabilities in the resulting games differ by at most this value.

To go from $\mathcal{G}_3$ to $\mathcal{G}_4$, we have to bound the probability that the plaintext-extractor rejects a correct ciphertext C'. Let $m'||r'$ be obtained through the decryption algorithm by computing $R' = \mathsf{DH}(C'_1, W)$ and extracting the leading bits of $z' \oplus c'_2$, where z' is the first coordinate of R'. The situation that we wish to avoid happens if the value of H at $(m'||r')$ is later set at a value t' such that $C'_1 = t' \cdot P$. We get:

$$H(m'||r') = \log_P(C'_1) \bmod p$$

which defines a subset of probability $\simeq \frac{1}{p}$. Altogether, we get the bound $\frac{q_D}{p}$.

Finally, in game $\mathcal{G}_4$, the value of $H(m_b||r_b)$ is left random. Thus $\mathcal{G}_4$ can be viewed as a distinguisher for the primitive encryption scheme. Its advantage is at least

$$\varepsilon - \frac{q_H}{2^{\ell_r}} - \frac{q_H}{2^{\ell_r}} - \frac{q_D}{p}$$

Now, the heavy load in the simulation is reencryption of all candidates from the **H-list**. This means q_H such computations. This completes the proof.

Remark. The bound $\frac{q_D}{p}$ increases to $\frac{q_D}{2^{\ell_h}}$, when the outputs of the hash function are of length $\ell_h < |p|$.

4.3 Summary of the security analysis

If it were possible to piece together the result in theorem 3 and the semantic security of the primitive encryption function, based on the ECDDH, one would obtain the security of PSEC-1 against CCA adversaries. Unfortunately, the primitive encryption function is not semantically secure. There is a further objection: even if the primitive encryption function was modified to meet semantic security, it would be extremely difficult to draw any conclusion from the estimates in theorem 3. This comes from the term $\frac{2q_H}{2^{\ell_r}}$, which requires ℓ_r to be quite large. Allowing $q_H \sim 2^{60}$, as is often envisioned in such setting, and aiming at a security level 2^{-80}, means $\ell_r \geq 140$, which would leave space for only 20 bits of message. In any case, the suggested size $\ell_r = 32$ from [28] is clearly too low to back the security on theorem 3. It is easily seen that there is a adversary against semantic security, with advantage $1/2$, in time 2^{ℓ_r} (just encrypt m_0 with all possible choices of r). Thus, the relevance of the conversion method from [14] to elliptic curve cryptosystems is questionable.

5 Security Analysis of PSEC-2

As for PSEC-1, document [29] includes only security statements and points to the literature, notably [15] for proofs. The self-evaluation report [29] states that PSEC-2 is IND-CCA in the random oracle model, based on the elliptic curve computational Diffie-Hellman hypothesis ECCDH.

5.1 Semantic 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 [33], but which we use here with a slightly improper meaning, due to the fact that, in the present context, the session key is message dependent. This analyzes the semantic security of the symmetric session key itself in the context of CCA-adversaries: the attacker is allowed to query a decryption oracle by submitting a pair (C_1, c_2), where $C_1 \in E$ and c_2 is a $|q|$-bit string, and receiving the key session key $\mathsf{k} = G(r)$ as the answer, where C_1 and c_2 are built from r and m as prescribed by the cryptosystem. His aim is to distinguish the distribution consisting of a ciphertext (C_1, c_2) and the corresponding key material k from the analogous distribution where the key material is replaced by a random string. 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.

We prove the following:

Theorem 4 *Let $\mathcal{A}$ be a CCA–adversary attacking the key encapsulation scheme of PSEC-2 within time bound t, with advantage ε, making q_D, q_G and q_H queries to the decryption oracle and the hash functions G and H, respectively. There exists an adversary $\mathcal{B}$ inverting the primitive encryption function g with advantage $\frac{\varepsilon'}{q_H+q_G}$ and within time bound t', where*

$$\begin{aligned} \varepsilon' &\geq \varepsilon/2 - \frac{q_G + q_H}{2^{\ell_r}} - \frac{q_D}{p} \\ t' &\leq t + q_H \cdot \tau \end{aligned}$$

ℓ_R is the bit-size of R, and τ is the time needed to execute the encryption algorithm and is bounded by $\mathcal{O}(|q|^3)$

Proof: From $\mathcal{A}$, we build a machine $\mathcal{B}$ which attempts to invert the primitive encryption function g.

1. $\mathcal{B}$ receives its input, a ciphertext $C = (C_1, c_2)$.

2. $\mathcal{B}$ tosses a random coin b. If $b = 0$ it generates a random session key k and submits (C, k) to the distinguisher $\mathcal{A}$; if $b = 1$, it manufactures a correct pair (C', k') as prescribed by the scheme. In both cases, it handles the pair to $\mathcal{A}$

3. when the execution of the distinguisher has ended, $\mathcal{B}$ outputs a tentative the decryption of C, from the queries asked to G and H (see below)

Of course, during the entire simulation, $\mathcal{B}$ also has to simulate answers from the random oracle. This is done using a **H-list** and a **G-list**. For a fresh query to H, i.e. a query not on the **H-list**, the oracle picks a random answer and similarly for G. We also define a plaintext-extractor as follows:

1. on a query C' to the decryption oracle, $\mathcal{B}$ looks at the **H-list**. If there is an element on this list which is parsed as $(m'||r', H(m'||r'))$, with r' of the appropriate length, and if $g(r', H(m'||r')) = C'$, then $\mathcal{B}$ outputs $G(r')$ as the requested session key, including it on the **G-list**, if needed.

2. Otherwise, $\mathcal{B}$ rejects the ciphertext

Finally, we explain what is the final answer of $\mathcal{B}$ is, besides the b' bit computed by $\mathcal{A}$. When execution has ended, $\mathcal{B}$ looks at the **G-list** and the **H-list** and outputs a randomly chosen element, which appears as a question r in the **G-list** or such that $(m||r)$ appears as a question in the **H-list**.

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

2. game $\mathcal{G}_2$, where the decryption queries are answered by the plaintext-extractor

3. game $\mathcal{G}_3$, where the oracle is simulated

We let ε' the probability that r appears in the **G-list** or the **H-list** upon completion.

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 pairs (C, k), with random session key k and θ_1 the probability that it outputs 1 when they are taken from pairs constructed according to the rules of key-encapsulation. This is $1/2 + \frac{\varepsilon}{2}$.

To go from $\mathcal{G}_1$ to $\mathcal{G}_2$, we have to bound the probability that the plaintext-extractor rejects a correct ciphertext C'. Let r' be the decryption of C' under the primitive encryption scheme. The situation that we wish to avoid happens if the value of H at $(m'||r')$ is later set at a value t' such that $C'_1 = t' \cdot P$. We get:

$$H(m'||r') = \log_P(C'_1) \bmod p$$

which defines a subset of probability $\simeq \frac{1}{p}$. Altogether, we get the bound $\frac{q_D}{p}$ (or - if the output bit-length ℓ_H of H is $< p$ - the bound $\frac{q_D}{2^{\ell_h}}$).

The simulation in game $\mathcal{G}_3$ is perfect unless it conflicts with the implicit constraint on H coming from the assumption that C is a ciphertext corresponding to an unknown (but fixed) r. In the find stage, r has not been chosen and is involved in a G-query or a H-query with probability $\leq \frac{1}{2^{\ell_r}}$. In the guess stage, this can only happen when r does appear as a possible choice for the final output of $\mathcal{B}$. Altogether, we discard a set of probability $\leq \frac{q_G+q_H}{2^{\ell_r}} + \varepsilon'$. Applying the lemma, we see that the probabilities in the resulting games differ by at most this value. Note that, once the exceptional runs have been discarded, we are left with executions where r is not asked to G. We can freely interpret $G(r) = \mathsf{k}$ and are left with a distribution of inputs independent from b. The probability that $b' = b$ in this case is therefore $1/2$.

Finally, we have bounded the probability that $\mathcal{B}$ inverts the primitive encryption function of PSEC by $\frac{\varepsilon'}{q_H+q_G}$, where ε' is such that

$$\varepsilon/2 \leq \varepsilon' + \frac{q_D}{p} + \frac{q_G + q_H}{2^{\ell_r}}$$

Now, the heavy load in the simulation is the reencryption of all candidates from the **H-list**. This means q_H such computations.

5.2 From key encapsulation to chosen ciphertext security

We prove the following.

Theorem 5 *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*

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

and $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t')$ denotes the security level of the symmetric encryption scheme, as defined below.

Before going further with the proof, let us define more formally the relevant security notion for the symmetric encryption.

5.2.1 Symmetric Encryption

A symmetric encryption scheme with key-length k, operating 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* [18]: 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

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

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.

5.2.2 Security of PSEC-2

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

1. run the key generation algorithm for the 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_1, c_2) and the session key k and encrypt m_b as c_3, using the session key
3. run $A_2((C_1, c_2, c_3), st)$ and get an answer b'. Finally, output bit $b = b'$.

As in the proof of theorem 4, 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 session key to encrypt the test message m_b is replaced by a random string and where queries whose initial part matches with (C_1, c_2) 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_1, c_2) are rejected

Note that the running time of all games is $t' = t + \mathcal{O}(q_D)$, where the second term accounts for the symmetric decryptions. Also, observe that the probability that $\mathcal{G}_1$ outputs 1 (which means $b' = b$) is exactly $1/2 + \varepsilon$. Indeed, game $\mathcal{G}_1$ provides the adversary operating within the real-life setting. As in the proof of theorem 4, 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 session key corresponding to (C_1, c_2) 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 of m_b and the corresponding session key (in game $\mathcal{G}_1$) and the analogous distribution where the key is replaced by a random string (in game $\mathcal{G}_2$). This bounds the difference by $\mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t', q_D)$.

Going from game $\mathcal{G}_2$ to game $\mathcal{G}_3$, we note that a difference only occurs when one rejects a valid ciphertext (C_1, c_2, c_3'). Let m' be the corresponding plaintext. Since the queried ciphertext is different from the challenge ciphertext, m' is different from m_b. However, since the first component C_1 is similar, we have

$$H(m_b||r) \cdot P = H(m'||r) \cdot P$$

The above yields the collision

$$H(m_b||r) = H(m'||r) \bmod p$$

Since H is a random oracle, such collision appears with probability $\frac{1}{p}$. As usual, one should replace $\frac{1}{p}$ by $\frac{1}{2^{\ell_h}}$, when the bit-length of the H-outputs is $< k$.

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 5.2.1. Therefore, the advantage of the adversary in this latter game $\mathcal{G}_3$ is bounded by $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t')$.

Remark. There is a slight ambiguity under the notation $\mathsf{Adv}^{\mathcal{K},\mathcal{E},\mathcal{D}}(t', q_D)$ in the above theorem: it actually refers to the advantage of a distinguisher for the key encapsulation scheme, which first runs A_1 and next uses as message space the two test messages output by A_1. It is easily seen that theorem 4 can be applied to such distinguishers. Thus, despite the ambiguity, we have chosen to split the security analysis, for the sake of clarity.

5.3 Summary of the security analysis

Piecing together the results of theorems 4, 5 and 2, we see that PSEC-2 is semantically secure against CCA adversaries, based on the ECCDH. The sequence of reductions under this result is not extremely good, due to the multiplicative loss $\frac{1}{q_G+q_H}$ in theorem 4 and it is unclear whether it could lead to meaningful estimates.

It can be noted that the hypotheses of the proof of theorem 2 requires that ℓ_h is close to k. A few bits can be discarded with a multiplicative loss $1/2$ in the probabilities, for each bit. However, the submission only requires that $\ell_h \leq k$ and this leaves open the possibility that $\ell_h << k-1$, in which case the security proof collapses. The self-evaluation document [29], sets $\ell_h = k-1$.

Similarly, ℓ_r should be close to $|q|$. However, the submission only requires that $\ell_r \leq |q|$ and this leaves open the possibility that $\ell_r << |q|$, in which case the security proof collapses. The self-evaluation document [29], sets $\ell_h = |q| - 1$.

Thus, the submission does not make completely clear the role of the various parameters: the analysis shows that

1. the bit-length ℓ_h of the hash function should not be much smaller than k (the submission only requires $\ell_h \leq k$).
2. the bit-length ℓ_r of the random string should be close enough to $|q|$ (the submission only requires $\ell_r \leq |q|$).

However, no part of the specification specifically points out these requirements, even if the results quoted in the self-evaluation report [29] duly consider cases where they hold.

6 Security Analysis of PSEC-3

The self-evaluation report [29] includes security statements and an appendix, which is a preliminary version of [25, 26]. PSEC-3 is a hybrid cryptosystem which addresses the drawback that reencryption was needed at the decryption phase of PSEC-2 (as it was in PSEC-1). This is done by applying the conversion method from [25]. This conversion does not include the first hash function H at key encapsulation but uses it to as a tag once the symmetric encryption has been performed. This is along the same lines as [1, 33]. However, the resulting scheme does not rely any more on the Diffie-Hellman hypothesis for elliptic curves but on a version of the so-called gap-problems, which now discuss.

6.1 Gap Diffie-Hellman Problem

In the following, we review the version of the so-called gap problems (see [26]) 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 3.1.1.

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

is greater then $1 - \delta$:

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

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 wanted, 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 access to a δ-ECDDH oracle.

The submission states that PSEC-3 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 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.

6.2 Semantic security of key encapsulation

As we did for PSEC-2, we would like to first focus on key encapsulation. This is perfectly possible, and involves building a simulator: when receiving a query (C_1, c_2), the simulator should provide $G(u)$, where u is the decryption of (C_1, c_2) under the primitive encryption function of PSEC. A natural option is to search in the **G-list**, hoping to find the appropriate value of u, and to perform a check by calling the ECDDH oracle. Although the approach is sound, it involves at least $q_D \times q_G$ oracle calls. The full reduction yields a better count and this is why we have chosen to directly proceed to the full security proof.

6.3 Semantic security of PSEC-3

Theorem 6 *Let $\mathcal{A}$ be a CCA–adversary attacking the semantic security of PSEC-3 within time bound t, with advantage ε, making q_D, q_G and q_H queries to the decryption oracle and the hash functions G, H, respectively. There exists an adversary $\mathcal{B}$ inverting the primitive encryption function of PSEC with the help of a δ-ECDDH oracle with advantage ε' and within time bound t', where*

$$\begin{aligned} \varepsilon &\leq \varepsilon' + \mathsf{Adv}^{\mathsf{E,D}}(t') + \frac{q_D}{2^\ell} + (4q_H + 2q_G) \cdot \delta \\ \text{where } t' &\leq t + \mathcal{O}((4q_H + 2q_G)\tau) \end{aligned}$$

τ accounts for operations in the underlying field, whose complexity is $\mathcal{O}(|q|^3)$ and $\mathsf{Adv}^{\mathsf{E,D}}(t')$ denotes the security level of the symmetric encryption scheme.

Proof: We start from an attacker $\mathcal{A} = (A_1, A_2)$ and use this adversary as always.

1. run the key generation algorithm for the 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 public key encryption scheme to get (C_1, c_2) and the session key k, encrypt m_b as c_3 under the session key and finally compute $c_3 = H(C_1||c_2||c_3||u||m)$
3. run $A_2(C_1||c_2||c_3||c_4, st)$ and get the output bit b'. Finally, output bit $b = b'$.

As usual, we will need to simulate answers from the random oracle. This is done using a **H-list** and a **G-list**. For a fresh query to H, i.e. a query not on the **H-list**, the oracle picks a random answer and similarly for G. We also define a plaintext-extractor as follows:

1. on a query $C' = (C'_1||c'_2||c'_3||c'_4)$ to the decryption oracle, $\mathcal{B}$ first looks at the **H-list**. For any u' of the proper bit-length $|q|$ appearing in a question to H of the form $C'_1||c'_2||c'_3||u'||m'$ in the list, and such that the corresponding answer is c'_4, the extractor computes $z' = c'_2 \oplus u'$ and the two elliptic curve points R'_1, R'_2, whose first coordinate is z' (if they exist). Next, it queries the δ-ECDDH oracle at (P, W, C'_1, R'_i).
2. If the answer of the oracle is positive, the extractor calls G at u' (extending the **G-list** if needed) and obtains the corresponding session key k. Once this is done, it checks whether m' is the decryption of c'_3
3. As soon as it has found a pair (u', m'), as described above the extractor returns m' as the requested plaintext. If none has been found, the query is rejected.

Note that proper bookkeeping allows to query the δ-ECDDH oracle at most twice for each H-query.

Finally, we explain why the simulation is related to inverting the primitive encryption function of PSEC. We define a machine $\mathcal{B}$ which is as above but receives an input (C_1, c_2) and a session key to encrypt m_b. This machine performs an additional step when execution has ended. Parsing the questions in the **G-list** and the **H-list**, $\mathcal{B}$ looks for a string u of the appropriate length $|q|$, appearing in either list. For each such string, $\mathcal{B}$ computes $z = c_2 \oplus u$ and the two elliptic curve points R_1, R_2, whose first coordinate is z (if they exist). Next, it queries the δ-ECDDH oracle at (P, W, C_1, R_i). As soon as it finds an element such that the answer of the oracle is positive, $\mathcal{B}$ returns the corresponding value u. Note that the running time t' of $\mathcal{B}$, when it is executed with the help of the plaintext extractor is the running time t of $\mathcal{A}$ plus an additional term $\mathcal{O}(4q_H + 2q_G)\tau$ to process the data on the lists and call the oracles.

We 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

2. game $\mathcal{G}_2$, where the decryption queries are answered by the plaintext-extractor using a perfect ECDDH oracle until $\mathcal{B}$ is able to output an answer u. Oracles are still assumed perfect and further decryption queries are answered by an actual decryption oracle.

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 u.

4. game $\mathcal{G}_4$, which is played as $\mathcal{G}_3$ but stops (before asking the G or H oracle) as soon as $\mathcal{B}$ is able to output an answer u.

5. game $\mathcal{G}_5$, where the session key is not derived as prescribed by the cryptosystem but is randomly chosen

6. game $\mathcal{G}_6$, where a δ-ECDDH oracle is used

We observe that the probability that $b' = b$ in game $\mathcal{G}_1$ is exactly $1/2 + \varepsilon$. Indeed, game $\mathcal{G}_1$ provides the adversary with the real-life setting. As usual, we bound the difference of probabilities between $\mathcal{G}_1$ and $\mathcal{G}_6$.

To go from $\mathcal{G}_1$ to $\mathcal{G}_2$, we have to bound the probability that there is a mismatch between the plaintext-extractor and the decryption oracle. First notice that the decryption algorithm should definitely check that the first component of the ciphertext duly belongs to the curve and is an element of order p. Otherwise, there is a risk that the extractor rejects a ciphertext, while it is correctly decrypted. Granted this, we only have to consider the probability that the extractor rejects a correct ciphertext

$C' = (C'_1, c'_2, c'_3, c'_4)$. There are two cases, depending on whether (C'_1, c'_2) and (C_1, c_2) encrypt the same value or not. We first cover the second case. Let u' be the decryption of C'_1 under the primitive PSEC scheme. The situation that we wish to avoid can be described as follows

1. the value of G at u' is set at k'
2. let $m' = \mathsf{D}_{\mathsf{k}'}(c'_3)$; the value of H at $(C'_1||c'_2||c'_3||u'||m')$ is later set at c'_4.

The conditional probability of the second item, once the first happens, is bounded by $\frac{1}{2^{\ell_h}}$.

We turn to the first case, where the decryption of (C'_1, c'_2) under the primitive PSEC scheme equals the decryption u of (C_1, c_2). Here, the value of $G(u)$ is implicitly constrained by the assumption that C is a ciphertext of the test message m_b. Fortunately, $\mathcal{B}$ has already found the decryption of (C_1, c_2) in this case and, therefore, the plaintext-extractor is not called.

Altogether, we have bounded the probability that the plaintext-extractor is wrong by $\frac{q_D}{2^{\ell_h}}$. Discarding the corresponding executions of games $\mathcal{G}_1$ and $\mathcal{G}_2$ leads to perfect simulation. Thus the probabilities that $\mathcal{G}_1$ and $\mathcal{G}_2$ output one differ by at most $\frac{q_D}{2^{\ell_h}}$.

The simulation in game $\mathcal{G}_3$ is perfect since it cannot conflict with the implicit constraint coming from the assumption that C is a ciphertext of the test message m_b. Indeed, u cannot be part of a question to the simulated versions of G or H, where u is, as defined above, the decryption of (C_1, c_2) under the primitive PSEC scheme. This is because $\mathcal{B}$ has earlier found the decryption of (C_1, c_2).

To go from $\mathcal{G}_3$ to $\mathcal{G}_4$, we just have to exclude the event of probability $\varepsilon' = \mathsf{Succ}^{\mathsf{ow}}(\mathcal{B})$ that $\mathcal{B}$ has correctly decrypted (C_1, c_2) before it stops. This bounds the difference of the probabilities that each game outputs one.

The simulation in game $\mathcal{G}_5$ is perfect unless it conflicts with the implicit constraint on G coming from the assumption that C is a ciphertext of a test message m_b. Let u be the decryption of C_1 under the primitive PSEC scheme. The conflict can only occur if u is queried to G. Since $\mathcal{G}_3$ has aborted beforehand, this cannot happen.

To go from $\mathcal{G}_5$ to $\mathcal{G}_6$, one just needs to estimate the probability that the δ-ECDDH-oracle returns a wrong answer. Since it is queried $(4q_H + 2q_G)$ times, this is bounded by $(4q_H + 2q_G) \cdot \delta$

Finally, we have bounded the difference between the respective probabilities that $\mathcal{G}_1$ and $\mathcal{G}_6$ output one by:

$$\varepsilon' + \frac{q_D}{2^{\ell_h}} + (4q_H + 2q_G) \cdot \delta$$

To conclude, consider the probability that $\mathcal{G}_6$ 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 5.2.1. Therefore, the advantage of the adversary in this latter game $\mathcal{G}_6$ is bounded by $\mathsf{Adv}^{\mathsf{E},\mathsf{D}}(t')$.

6.4 Summary of the security analysis

Piecing together the results of theorems 6 and 2, we see that PSEC-3 is semantically secure against CCA adversaries, based on the hardness of the Gap Diffie Hellman problem for elliptic curves. The sequence of reductions under this result can be made tight by an appropriate choice of the parameters

Again, the submission does not make completely clear the role of the various parameters: contrary to the case of PSEC-1 and PSEC-2, there is no bit-size indication for u and r: u is randomly chosen in $\{0,1\}^{|q|}$ and r in $\mathbb{Z}_p^\star$. However, the reason for such difference is not explained and implementors might make mistakes.

Also, the submission does not clearly state whether the decryption algorithm checks that the first component of a ciphertext is an element of the curve of order p. We believe that this check is needed.

7 Conclusions

Based on our analysis, we think that the cryptosystem PSEC-1 is not semantically secure, as proposed. Thus, we would not recommend the scheme as it is. We believe that PSEC-2, PSEC-3 are secure, with the proposed parameters. However, based on the submission, we have the following restrictions:

- The specification [28] contains notational ambiguities and omits additional requirements on the parameters that appear necessary to complete the various security proofs. These requirements are correctly instantiated in the self-evaluation report [29] but the correct choices are not further documented.
- Although there is a proof that PSEC-2 and PSEC-3 enjoy semantic security against adaptive chosen-ciphertext attacks (with proper requirements on the bit-size of parameters), it is questionable whether this proof gives conclusive evidence for PSEC-2, when interpreted with the proposed parameters.
- The submission does not clearly state whether the decryption algorithm should check that the first component of a ciphertext is an element of order p of the elliptic curve. This seems needed for PSEC-3.
- The Gap Diffie Hellman assumption for elliptic curves, on which PSEC-3 is based, appears to have been introduced too recently to form the basis of a cryptosytem.

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] D. Coppersmith, Finding a Small Root of a Bivariate Integer Equation; Factoring with High Bits Known, Eurocrypt'96, Lecture Notes in Computer Science 1070, (1996), 178–189.

[9] 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.

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

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

[12] 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.

[13] 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.

[14] E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. In *PKC '99*, LNCS 1560, pages 53–68. Springer-Verlag, Berlin, 1999.

[15] E. Fujisaki and T. Okamoto. Secure Integration of Asymmetric and Symmetric Encryption Schemes. In *Crypto '99*, LNCS 1666, pages 537–554. Springer-Verlag, Berlin, 1999.

[16] E. Fujisaki and T. Okamoto. How to Enhance the Security of Public-Key Encryption at Minimum Cost. *IEICE Transaction of Fundamentals of Electronic Communications and Computer Science*, E83-A(1):24–32, January 2000.

[17] 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.

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

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

[20] IEEE standard specifications for public-key cryptography. IEEE Computer Society (2000).

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

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

[23] 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.

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

[25] T. Okamoto and D. Pointcheval. REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform. In *RSA '2001*, LNCS. Springer-Verlag, Berlin, 2001.

[26] 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.

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

[28] Specification of PSEC: provably secure elliptic curve encryption scheme.

[29] Self evaluation of PSEC: provably secure elliptic curve encryption scheme.

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

[31] 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.

[32] 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.

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

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

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

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