Evaluation Report on the **ECDHS Cryptosystem**

1 Introduction

This document is an evaluation of the **ECDHS Cryptosystem**. Our work is based on the analysis of document [9], which provides both the specification and self-evaluation of the scheme. The present report is organized as follows: firstly, we briefly review the cryptosystem; next we discuss the security level of the cryptographic primitive which underlies the scheme and analyze its relation to the difficulty of the discrete logarithm problem on elliptic curves; finally, we evaluate the security level of the scheme itself in the light of strong security notions similar to 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

ECDHS 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 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.

The basic function f on which ECDHS 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:

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

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

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

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

ECDHS is a key-agreement scheme: entities U that enter the scheme use a predefined curve E, as described above, and a base point G. For each such entity U, a key pair (d_U, Q_U) is established, consisting of an element Q_U of E of order n together with its discrete logarithm $d_U = \log_G Q_U$. The group element Q_U is the public key and d_U is the private key.

Key agreement is achieved by means of one of the following methods:

1. two entities U and V, with respective public keys Q_U and Q_V can derive a shared key by applying a key derivation function to $DH(Q_U, Q_V)$.

2. or else they can use $DH'(Q_U, Q_V)$ in place of $DH(Q_U, Q_V)$.

At this point, it is useful to introduce a more formal framework, that will be useful when we later perform the security analysis. A key-agreement 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 key agreement algorithm $\mathcal{E}$ is an interactive algorithm between two entities U, V, each endowed with a pair $(\mathsf{sk}_U, \mathsf{pk}_U)$ (and $(\mathsf{sk}_V, \mathsf{pk}_V)$ resp.), and each using random coins $r_U, r_V \in \Omega$

- the key derivation algorithm $\mathcal{D}_{\mathsf{sk}}(D; r)$ outputs the common key from the data Δ exchanged during the key agreement algorithm and the random coins. It should be the case that $\mathcal{D}_{\mathsf{sk}_U}(\Delta; r_U) = \mathcal{D}_{\mathsf{sk}_V}(\Delta; r_V)$

The key generation algorithm $\mathcal{K}(1^m)$ of the ECDHS Cryptosystem produces, on input m, a curve E, a base point G and its 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). 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. In both cases, a randomly chosen element Q_U of order n is generated for each participant and the public key pk_U consists of the above tuple together with Q_U. The secret key $\mathsf{sk}_U = d_U$ is the discrete logarithm of Q in base G.

The key-agreement algorithm $\mathcal{E}$ is played between two entities, each having a public/secret key pair, as explained earlier in this section. A shared field element z is produced, which is the first coordinate of a curve element computed as $DH(Q_U, Q_V)$ or - alternatively - as $DH'(Q_U, Q_V)$. By "shared", we mean that both entities U and V are able to obtain z, as will be seen in the sequel.

From the shared field element z, key material of appropriate length is generated. It is obtained by means of a hash function H, which is unkeyed and applied to z and - optionally - to publicly shared information. At this point, we wish to note that we have not been too careful with notations in the above description. For example, document [9] carefully explains how to treat z as a bit string or a byte string, providing 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.

The key derivation algorithm derives the shared field element as the first coordinate of a point P defined by the formula

$$P = d_U \cdot Q_V = d_V \cdot Q_U$$

or — if the cofactor Diffie-Hellman function was used —

$$P = h \cdot d_U \cdot Q_V = h \cdot d_V \cdot Q_U$$

and retrieves the key material from z, using the key derivation function.

2.2 Comments on the specification

Document [9] 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 $DH(X, Y) = h^{-1} \cdot DH'(X, Y)$. Thus, computing DH and DH' is strictly equivalent from a complexity theoretic viewpoint. Inclusion of the cofactor method eases the implementation. When a key pair (d_U, Q_U) is used in a key agreement protocol, there is no guarantee for the receiver that it has been correctly manufactured (unless it is directly certified by some form of certification authority). Hence, in order to avoid subtle attacks, the other party should check that Q_U is a non unit element of the subgroup generated by G, which means

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

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

In terms of security arguments, document [9] is rather sketchy. We quote the following comment, which is on a rather general level.

> The elliptic curve Diffie-Hellman scheme is a key agreement scheme based on ECC. It is designed to provide a variety of security goals depending on its application — goals it can provide include unilateral implicit key authentication, mutual implicit key authentication, known-key security, and forward secrecy — depending on issues whether or not public keys are exchanged in an authentic manner, and whether or not keys pairs are ephemeral or static.

The submission does not specifically discuss the authentication problems surrounding the Diffie-Hellman key-agreement protocol, on which the scheme builds. No indication either is given of how to implement the above security goals. When a fresh session key is established each time the protocol is executed, a security property usually expected is that, even if some previous session keys have been misused or corrupted, the new key should still be indistinguishable for the adversary. Furthermore, the property of *forward-secrecy*, quoted in [9], is also often required. It means that, even if the long-lived secret key of a participant is compromised, this should not help the attacker to learn anything about the previous values of the agreed keys. These questions are not further addressed in the self-evaluation report beyond the above comments.

On a more general level, one must regret that no formal security argument is offered and that no effort is made to precisely relate the scheme with a specified hard problem about elliptic curves.

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

As previously mentioned, document [9] does not attempt to measure the security of the scheme in terms of a hard problem related 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

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

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 [25], 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 [24]). 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 scheme

It appears that the security of the scheme is closely connected with the decisional Diffie-Hellman assumption. We will only consider the setting where the Diffie-Hellman function is applied, as the case of the cofactor Diffie-Hellman function is handled similarly. With the notations of section 2.1, z is the first coordinate of $DH(Q_U, Q_V)$. 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: if the equation of the elliptic curve E is

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

then, an integer x, $0 \leq x < p$ 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. A similar definition applies for curves over $\mathbb{F}_{2^m}$. We also let $\mathcal{H}$ be the image of $\mathcal{X}$ by the key derivation function H.

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

$$(G, Q_U, Q_V, k)$$

where k is the common key material generated by the cryptosystem, from the analogous distribution with k replaced by a random element of $\mathcal{H}$. 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

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

with G, Q, R, P chosen at random in the subgroup of order n of E and

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

with $P = DH(Q, R)$. We run the key generation algorithm and generate an elliptic curve E together with a random element G of large prime order. 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 scenario by considering that the second and third elements of the input are the respective public keys Q_U and Q_V of two entities U and V. This implicitly defines two matching secret keys. Next we submit $(G, Q, R, H(z))$ to $\mathcal{A}$, where z is the first coordinate of P. If the original input is from $\mathbf{D}$, the last element of the tuple is exactly the common key material produced by the cryptosystem. On the other hand, if the input is from $\mathbf{R}$, the last coordinate is a random element of $\mathcal{H}$. Thus, we have obtained a distinguisher between $\mathbf{D}$ and $\mathbf{R}$, with almost 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 of the appropriate length, which would mean that the information k is semantically secure in the sense of the seminal paper [17]. However, we do not see any argument that would give such guarantee. Of course, one can declare that the key derivation function is a random oracle. It is also possible to obtain such a bit-string by applying a randomly keyed universal hash function, following the method described in [25] and also used in [32]. Recall that, if H_h is a universal hash function, keyed by h, with ℓ-bit outputs, then, the leftover hash lemma of [21] implies that hashing a set of 2^λ bit strings produces a distribution $(h, H_h(x))$ 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 - say - 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 weaker 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) and the assumption underlying the scheme is stronger than the 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 is the Pollard ρ-method from [26] which takes computing time equivalent to about $\sqrt{\pi n/2}$ group operations. In 1993, van Oorschot and Wiener in [33], 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 [16, 34]), 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 [8] led to the computation of a discrete logarithm in a group with 2^{109} elements (see [15]). 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 [7]) 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 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 ([30]), Rück ([27]), Smart ([29]) and Satoh-Araki ([28]) 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 [1] 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 [14] (see also [13]) 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 ECDHS 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 [23]. Considering that the appropriate warnings are given, this does not constitute a strong objection to the proposal under review.

4 Security Analysis

Document [9] does not include a thorough security analysis, and does even not consider the formal security notions that a key agreement scheme should satisfy. Thus, we have found necessary to undertake our own security analysis. We first review the security notions which have been defined in the literature. Then we consider the proposed schemes in view of these notions.

4.1 Formal framework

4.1.1 Key agreement

A key agreement scheme (without TTP) involves two participants, a *client* $\mathcal{A}$ and a *server* $\mathcal{B}$, who want to share a secret session key in order to thereafter possess a secure and virtually private channel. They communicate on a public channel and eventually compute a value that they both know but which nobody else knows. Many security models have been defined to cover this kind of schemes. Of these, the following two models have received more attention:

- The first model was proposed by Bellare and Rogaway [5, 6], and refined in [3]. Here, the adversary can interact with all participants and aims at learning some information about one session key. Therefore, the proper approach is to ensure indistinguishability of the session key (from a random key) for the adversary. In other words, any session key should be semantically secure [17].

- The second model was proposed by Bellare, Canetti and Krawczyk [2], and is based on the multi-party simulatability technique. This means that one first defines an idealized version of a key agreement scheme. Then, in order to prove that the real-world scheme is secure, one shows that any adversary in the real world has to behave like an adversary in the ideal game.

Shoup [31] has shown that the two models (with adequate refinements) are equivalent in preventing active adversaries to break *forward-secrecy.* This property is by now a basic requirement for any key agreement scheme. *Forward-secrecy* means that an adversary, who sees all the public communication (and possibly has access to all session keys *but one*) cannot obtain any information about *that last* session key, even if he later learns the long-term secret of any party.

4.1.2 Mutual authentication

When parties have established a common secret session key, most of the key agreement protocols, such as the Diffie–Hellman [11] key agreement scheme using public keys, *implicitly* assume that each party is actually partnered (by sharing the session key) with the party he wanted. However, it can be the case that no partnership has been established. Indeed, if an adversary uses the public key of Alice and Bob runs the key exchange process, then, upon completion, he thinks that the actual session key is shared with Alice. However, there is no actual partner since the adversary cannot extract the session key from the communication.

Accordingly, one usually wants to furthermore verify the actual partnership. Such property of a key exchange scheme is called *mutual authentication.* However, as presented in [3], an *implicitly authenticated* key agreement scheme can be easily trans-

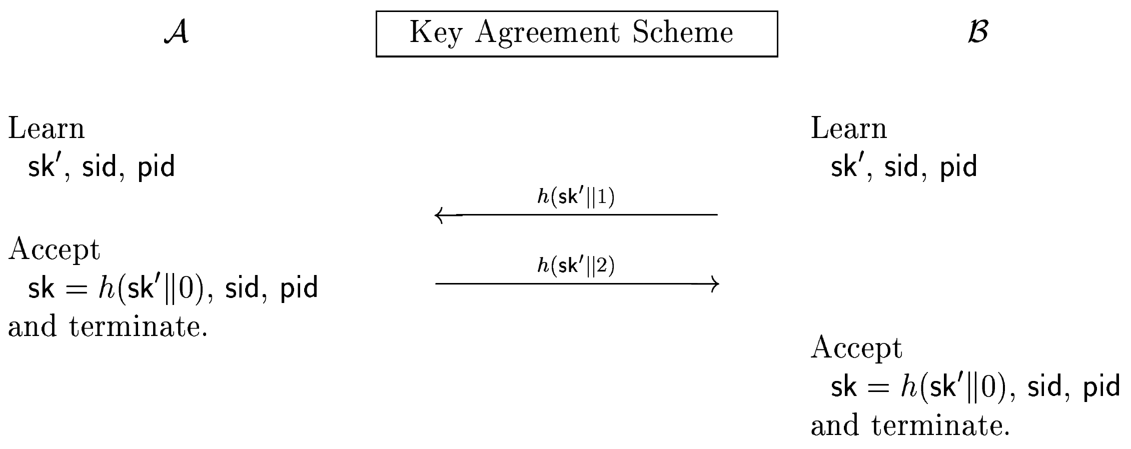

Figure 1: Key Agreement + Mutual Authentication

formed, in the random oracle model [4], into a scheme that provides mutual authentication, by simply adding one more flow (see figure 1).

4.2 Security model

At the end of each execution of the protocol (see figure 1), when a party $\mathcal{U}$ has accepted, it gets a session key, denoted by sk_U, and a session ID, denoted by sid_U which is part of the flow of data. The session ID's are made public, while session keys clearly remain secret. Indeed, the session keys are the common secret shared by the two parties at the end of the protocol. The session ID's have a technical significance: they are used to define partnership. The partner of a party is an entity which has a similar session ID. Since the session ID's are public, the partnership is also public. With such a definition of partnership, one can remark that a party may have several partners, although it is quite unlikely, in general.

In the model defined by Bellare and Rogaway [5, 6], with additional refinements in order to handle forward-secrecy (*cf.* Shoup [31]), any instance of each party, $\mathcal{A}$ or $\mathcal{B}$, is seen as an oracle (see figure 2). Furthermore, it is assumed that the entire communication network is managed by the adversary $\mathcal{C}$, who may schedule interactions arbitrarily, and who may inject and drop messages arbitrarily as well. Thus, the adversary can interact, as a man-in-the-middle, with all parties, or more formally with several instances of them ($\mathcal{A}_i$ for the client and $\mathcal{B}_j$ for the server) as many times as he wants in a concurrent way. He can ask the following queries

- Send ($\mathcal{U}$, i, *string*) – which means that the adversary sends the message *string* to the oracle $\mathcal{U}_i$ (either a server or a client). The oracle makes the requested computations according to the protocol and returns the answer.

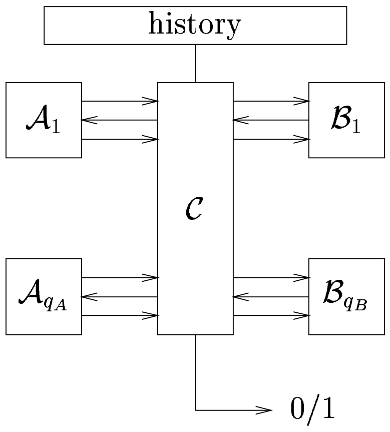

Figure 2: Security Model

- Reveal $(\mathcal{U},\ i)$ – provided oracle $\mathcal{U}_i$ has accepted (the tag acc has been set to True), it returns the session key sk_U^i. This models the misuse of a session key by the parties once this session key has been established.
- Test $(\mathcal{U},\ i)$ – granted that oracle $\mathcal{U}_i$ has accepted, one tosses a coin b. If $b = 1$ then the session key sk_U^i is returned, else a random string is returned. The aim of the attack is to guess this bit b. Therefore, the following restrictions on this query apply:
  - the query is asked once;
  - no Reveal-query is asked to $\mathcal{U}_i$;
  - no Reveal-query is asked to $\mathcal{U}_j$, where $\mathcal{U}_j$ is partnered with $\mathcal{U}_i$.
- Execute $(\mathcal{A}, i, \mathcal{B}, j)$ – in order to obtain the transcript corresponding to the communication between two parties (and build a history), the adversary may ask the parties to run the protocol. Then the transcript is returned to the adversary. Such a transcript can also be obtained using Send-queries.
- Corrupt $(\mathcal{U})$ – in order to deal with forward-secrecy, one allows the adversary to corrupt the parties. By this we mean that it obtains the secret key (the long-term secret key x_U) of the corrupted party $\mathcal{U}$. This induces a further restriction on the Test-query, which can only be asked to an instance of a party (or one of his partners) who has accepted before it gets corrupted.

The above game, with the Test-query, just deals with the semantic security of the session key, the basic security notion of a key agreement scheme, but not with (mutual) authentication. We say that the protocol provides mutual authentication if no instance accepts unless it has exactly one partner. Otherwise, it would mean that the adversary has impersonated a party. For example, if an instance of the server accepts without a partner, it means that the adversary has impersonated the client, and therefore broken client-to-server authentication. Similarly, if an instance of the client accepts without a partner, it means that the adversary has impersonated the server, and therefore broken server-to-client authentication. A key agreement scheme guarantees mutual authentication if, for any adversary, the probabilities of breaking the client-to-server authentication or the server-to-client authentication are both negligible.

4.3 Analysis of the proposal

4.3.1 The basic protocol

A passive adversary does not make use of the Send ($\mathcal{U}$, i, *string*) or Corrupt ($\mathcal{U}$) facilities. It may or may not ask Reveal-queries. If it does not, then it is easily seen that it would have to break the ECDDH problem in order to get an advantage in distinguishing the real session key from a "random" one (derived from a random point on the curve), as proven in theorem 1.

In the case of a passive adversaries with Reveal-queries, the scheme is not secure, since the session key is always the same. As soon as the adversary has asked a Reveal-query, he can break the semantic security of any other session key (since they are the same).

Finally, active adversaries cannot really be considered, since there are no interactions in this scheme, where only the public keys are used. Similarly, the scheme does not provide forward-secrecy, since the session key is completely and easily determined from the public keys, and the secret key of any participant.

4.3.2 One-time keys

We now turn to the situation where the two parties refresh their keys at each execution of the key agreement scheme. Notice that this scenario is not explicitly envisioned in document [9]. Still, it is needed in order to withstand active attacks. In this context, it is necessary to authenticate the flow of data by means of a digital signature. Thus, the long-term key on which the protocol relies is actually a signing key. The key-agreement scheme between two entities U and V, with respective identities ID_U, ID_V, can be played as follows (as proposed in [31].)

1. U generates a one-time key pair (d_U, Q_U) and sends Q_U, ID_V to V together with a signature $sig_U(Q_U, ID_V)$ of these data and a certificate for his public signing key

2. V generates a one-time key pair (d_V, Q_V) and sends Q_V, to U together with a signature $sig_V(Q_U, Q_V, ID_U)$ and a certificate for his public signing key

The key material can be derived by U and V by computing

$$P = d_U \cdot Q_V = d_V \cdot Q_U$$

or — if the cofactor Diffie-Hellman function was used —

$$P = h \cdot d_U \cdot Q_V = h \cdot d_V \cdot Q_U$$

Once this is done, the key material is retrieved using the key derivation function.

4.3.3 Digital Signature Schemes

Digital signature schemes are the electronic version of handwritten signatures for digital documents: a user's signature on a message M is a string which depends on M, on the secret key of the user and –possibly– on randomly chosen data, in such a way that anyone can check the validity of the signature by using the public key only.

Definitions. A signature scheme is defined by three algorithms $(\mathcal{K}, \Sigma, \mathcal{V})$:

- The *key generation algorithm* $\mathcal{K}$. On input 1^m, where m is the security parameter, the algorithm $\mathcal{K}$ produces a pair $(\mathsf{pk}, \mathsf{sk})$ of matching public and secret keys. Algorithm $\mathcal{K}$ is probabilistic.

- The *signing algorithm* Σ. Given a message M and a pair of matching public and secret keys $(\mathsf{pk}, \mathsf{sk})$, Σ produces a signature σ. The signing algorithm might be probabilistic.

- The *verification algorithm* $\mathcal{V}$. Given a signature σ, a message M and a public key pk, $\mathcal{V}$ tests whether σ is a valid signature of M with respect to pk. In general, the verification algorithm need not be probabilistic.

Security. Various security notions have been formalized by [18, 19], based on the *goal* of the adversary, and the *means* available to the adversary to achieve this goal.

- Disclosing the secret key of the signer. It is the most serious attack. This attack is termed *total break.*

- Constructing an efficient algorithm which is able to sign messages with significant probability of success. This is called *universal forgery.*

- Providing a new message-signature pair. This is called *existential forgery.*

In many cases the latter is not dangerous, because the output message is likely to be meaningless. Nevertheless, a signature scheme which is not existentially unforgeable (and thus that admits existential forgeries) does not guarantee by itself the identity of the signer. For example, it cannot be used to certify randomly looking elements, such as keys, which is the context considered in this report.

Two different kinds of attacks have been considered depending on the availability of signed messages for the adversary. In a first scenario, the attacker only knows the public key of the signer. In another, the attacker has access to a list of valid message-signature pairs. The strongest model is the *adaptively chosen-message attack*, where the attacker can ask the signer to sign any message, except the message for which forgery is finally achieved.

A signature scheme is *secure* if an existential forgery is computationally impossible, even under an adaptively chosen-message attack. We denote by $\mathsf{Succ}^{\mathcal{K},\Sigma,\mathcal{V}}(t,q)$ the maximal success probability of any adversary in performing an existential forgery after at most q queries to a signing oracle within time bound t.

4.3.4 Security analysis for one-time keys

We now analyze the security of the variant of the protocol, which uses one-time keys and digital signatures.

Passive adversaries without Reveal-queries. Again, it is easily seen that a passive adversary which does not ask any Reveal-query would have to break the ECDDH problem in order to get any advantage in distinguishing the real session key from a "random" one (derived from two random points on the curves), with a proof similar to the proof of theorem 1.

Passive adversaries with Reveal-queries. In this case, it can be shown that the scheme is secure relative to the ECDDH problem. More accurately, we prove the following result.

Theorem 2 *Let $\mathcal{A}$ be an adversary breaking the semantic security of the key agreement scheme, with advantage ε and within time bound t, having observed q_p transcripts and asked q_r Reveal-queries. Then the ECDDH problem can be solved with advantage greater than ε/q_p and within time bound $t + 3q_p \cdot \tau$, where τ accounts for the cost of an elliptic curve operation and is bounded by $\mathcal{O}(m^3)$.*

Proof. Let $\mathcal{A}$ be a passive adversary that guesses the bit b involved in the Test-query. We construct a distinguisher between the distributions $\mathbf{D}$ and $\mathbf{R}$, where

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

with all four elements taken at random and

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

with $\log_G(R) = \log_Q(P)$.

We run the key generation algorithm and generate an elliptic curve with a subgroup of order n as prescribed. We also run the key generation algorithm of the signature scheme. We next show how to use $\mathcal{A}$ to break the ECDDH on E: we take the base point of the cryptosystem to be the first coordinate of the input to $\mathcal{A}$.

The view of the adversary $\mathcal{A}$ can be simulated as follows:

- choose a random d_U and send $Q_U = d_U \cdot G$, with signature and certificate
- choose a random d_V and send $Q_V = d_V \cdot G$, with signature and certificate
- derive the shared session key k

In order to use the adversary to distinguish the above distributions, we randomly pick one index $i \in \{1, \ldots, q_p\}$ and modify the i-th execution. Instead of the above simulation, one uses the second component Q of the input to $\mathcal{A}$ in the first message and the third component R for the answer. The shared session key is obtained as $k = H(z)$, where z is the first coordinate of P.

The Reveal-queries can easily be simulated by means of the computed shared key. Similarly, the Test-query can be simulated, using a random coin b. Thus, provided that the Test-query is asked at the i-th execution, we see that the advantage of $\mathcal{A}$ is exactly the same as in the real game, when the input to $\mathcal{A}$ comes from $\mathbf{D}$. On the other hand, when the input to $\mathcal{A}$ comes from $\mathbf{R}$, the advantage of $\mathcal{A}$ is exactly 0. Since the choice of i is independent from the view of the adversary, the advantage of our distinguisher is ε/q_p, and its running time is $t' = t + 3q_p \cdot \tau$, where τ accounts for the cost of the elliptic curve operation needed to compute the data to be handled to $\mathcal{A}$.

Using an improved reduction, similar to [10], one can state

Theorem 3 *Let $\mathcal{A}$ be an adversary breaking the semantic security of the key agreement scheme, with advantage ε and within time bound t, having observed q_p transcripts and asked q_r* Reveal*-queries, then the ECDDH problem can be solved with advantage greater than $e^{-1} \cdot \varepsilon/q_r$ and within time bound $t + 3q_p \cdot \tau$, where τ accounts for the cost of the elliptic curve operation and is bounded by $\mathcal{O}(m^3)$.*

Proof. The proof is similar to the above, but the simulation of the view of the adversary $\mathcal{A}$ is slightly different. With probability $1 - \pi$, one performs the standard simulation, having knowledge of the discrete logarithms in use, and with probability π the simulation uses inputs derived from (Q, R, P) using random self-reducibility (their discrete logarithms are unknown). A similar analysis can be performed, provided that all the Reveal-queries have been asked to correctly simulated transcript, and that the Test-query has been asked on a transcript involving a key derived from (Q, R, P). This occurs with probability $(1 - \pi)^{q_r}\pi$. If we define $\pi \approx 1/q_r$, the above probability is approximately equal to e^{-1}/q_r, which concludes the proof.

Remark. As observed at the end of section 3.1.2, one cannot guarantee that the session key k is indistinguishable from a random string of the same length (but only from a string obtained from a random curve element.) This has the unpleasant consequence that the session key, seen as a bit string, is not proven semantically secure, unless H is assumed a random oracle.

Active adversaries. We now cover the case of active adversaries, allowing the use of Send ($\mathcal{U}$, i, *string*) but not of the Corrupt ($\mathcal{U}$) facility, at this point.

Due to the signature, the scheme can be shown secure relative to the ECDDH problem. More accurately, we prove the following:

Theorem 4 *Let $\mathcal{A}$ be an adversary breaking the semantic security of the key agreement scheme, with advantage ε and within time t, having asked q_r* Reveal*-queries, and interacted q times with U and V. Then*

$$\begin{aligned} \varepsilon &\leq 2 \times \mathsf{Succ}^{\mathcal{K},\Sigma,\mathcal{V}}(t', q) + q^2 \times \mathsf{AdvDDH}(t') \\ \textit{with } t' &\leq t + 2q \cdot \tau \end{aligned}$$

where τ accounts for the cost of an elliptic curve operation and is bounded by $\mathcal{O}(m^3)$.

Remark. In the above theorem, we assume that passive observations (the Execute-queries) are built from interactions with U and V (using Send-queries).

Proof. Let $\mathcal{A}$ be an active adversary that guesses the bit b involved in the Test-query after q interactions with U and V (q_u and q_v respectively), and q_r Reveal-queries.

In order to prove the above security result, we will envision several games:

- game $\mathcal{G}_1$, where the signatures are generated by the actual signature algorithms (by means of the secret keys)
- game $\mathcal{G}_2$, where all messages to V which involve a fresh signature (i.e. a signature not produced by our simulation of U) are rejected
- game $\mathcal{G}_3$, where all messages which involve a fresh signature (i.e. a signature not produced by our simulators) are rejected

The probability that the adversary correctly guesses b in $\mathcal{G}_1$ is exactly $1/2 + \varepsilon$. Indeed, game $\mathcal{G}_1$ provides the adversary with the real-life setting.

In the following, we bound the difference of probabilities that $\mathcal{G}_1$ and $\mathcal{G}_3$ successfully guess the query bit b, and we relate the advantage in the game $\mathcal{G}_3$ with the ability to break the ECDDH. This uses the simple yet useful lemma from [32]

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.

Going from game $\mathcal{G}_1$ to $\mathcal{G}_2$ produces a difference if and only if a message produced by the adversary involves a fresh signature, accepted under the public key pk_u. In order to estimate the probability that such event happens, we define a simulation. This simulation runs the key generation algorithm anew and generates an elliptic curve and a base point G of order n as prescribed. It also runs the key generation algorithm of the signature scheme to get a public key pk_u for U, and a pair of keys $(\mathsf{sk}_v, \mathsf{pk}_v)$ for V, as well as the certificates for the public keys. At this point, one can simulate the view of the adversary $\mathcal{A}$, with the help of a signing oracle for U:

- to simulate U, choose a random d_U and compute $Q_U = d_U \cdot G$, then ask the signing oracle the signature $sig_U(Q_U, ID_V)$. Return Q_U, the signature and a certificate for pk_u
- to simulate V, perform similarly, using the secret signing key in place of the oracle. More explicitly, upon receiving data produced by the simulation of U, abort if the received signature is fresh (i.e. has not been created by the above U-simulator.) Otherwise, choose a random d_V and compute $Q_V = d_V \cdot G$; next, produce the signature $sig_V(Q_U, Q_V, ID_U)$. Return Q_V, the signature and a certificate for pk_v.
- U accepts if and only if the received signature is correct. Then each of U, V derives the shared session key k

This also simulates Execute-queries in $\mathcal{G}_2$. The case of Reveal-queries is easily handled, using the computed shared key. Similarly, Test-queries can be simulated, by means of a random coin b. Now, game $\mathcal{G}_2$ differs from game $\mathcal{G}_1$ if a fresh signature is valid, but this is exactly the probability that the simulation provides an existential forgery, with less than q_u queries to the signing oracle and within time bound $t' = t + 2q \cdot \tau$. Using the lemma, this bounds the difference between the success probabilities by $\mathsf{Succ}^{\mathcal{K},\Sigma,\mathcal{V}}(t', q_u)$. A similar analysis bounds the difference between games $\mathcal{G}_2$ and $\mathcal{G}_3$ by $\mathsf{Succ}^{\mathcal{K},\Sigma,\mathcal{V}}(t', q_v)$.

We are thus led to study the advantage that the adversary can get in game $\mathcal{G}_3$. We relate this advantage to a distinguisher between the two distributions $\mathbf{D}$ and $\mathbf{R}$, that we now describe. We run the key generation algorithm and generate an elliptic curve and a base point of order n as prescribed. We also run the key generation algorithm of the signature scheme to get the signing and verification keys, as well as the certificates. We next show how to use $\mathcal{A}$, on input (G, Q, R, P), to break the ECDDH. We take the base point of the cryptosystem to be the first coordinate G of the input to $\mathcal{A}$. We then simulate the view of the adversary $\mathcal{A}$, as follows:

- to simulate U, choose a random d_U and send $Q_U = d_U \cdot G$, with signature and certificate
- to simulate V, choose a random d_V and send $Q_V = d_V \cdot G$, with signature and certificate
- then each of U and V derives the shared session key k.

In the above simulations of U and V, they only accept signatures generated by the simulator. In order to use the adversary to distinguish the above distributions, we randomly choose two indices $i, j \in \{1, \ldots, q\}$. The i-th simulation of U is modified, using the second component Q of the input to $\mathcal{A}$ to define Q_U. Similarly, the j-th simulation of V, provided it answers a message involving Q, uses the third component R of the input to $\mathcal{A}$ to define Q_V. The shared session key is obtained as $k = H(z)$, where z is the x coordinate of P.

The Reveal-queries are simulated by answering the computed shared key. Similarly, the Test-query are simulated, by means of a random coin b. Thus, provided that the Test-query is asked at a point where Q and R are involved, we see that the advantage of $\mathcal{A}$ is exactly the same as in the real game, when the input to $\mathcal{A}$ comes from $\mathbf{D}$. On the other hand, when the input to $\mathcal{A}$ comes from $\mathbf{R}$, the advantage is exactly 0. Since the choice of i and j is independent from the view of the adversary, the advantage of our distinguisher is ε/q^2, and its running time is $t' = t + 2q \cdot \tau$. This finishes the proof.

Forward-secrecy. The above result can be extended to the case where $\mathcal{A}$ is allowed to use the $\mathsf{Corrupt}$ $(\mathcal{U})$ facility. However, as pointed out in [31], its use should not be too liberal: one sees that the above proof collapses if the adversary corrupts V right after a Test-query. The proper restriction can be expressed in terms of the session ID, as follows:

- a session ID is *established* only after two parties $\mathcal{U}$ and $\mathcal{V}$ have interacted to share this ID

- if a Test-query is asked to a party $\mathcal{U}$, no Reveal-query can be issued with the session ID involved in the Test-query
- if a Test-query is asked to a party $\mathcal{U}$ which later gets corrupted, the corruption cannot take place before the session ID involved in the Test-query has been established.

As shown by Shoup [31], in order to get the most general version of forward security, it is necessary to add at least one key confirmation flow.

4.3.5 Mutual Authentication

As already studied (see figure 1), by simply adding the key confirmation flows (only one, or both), one gets the explicit unilateral or mutual authentication. Without such a key confirmation, the users may accept without having any actual partner.

5 Conclusion

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

- The specification does not include any formal discussion of security. As proposed, the scheme is only secure in the weakest formal security model.
- The scheme does not, by itself, provide forward secrecy.

We wish to note additionally that we have been able to show that the scheme could be used securely and provide forward secrecy, in conjunction with digital signatures. Nevertheless, no indication on how to use such setting is included in the specification.

References

[1] 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.

[2] M. Bellare, R. Canetti, and H. Krawczyk. A Modular Approach to the Design and Analysis of Authentication and Key Exchange Protocols. In *Proc. of the 30th STOC.* ACM Press, New York, 1998.

[3] M. Bellare, D. Pointcheval, and P. Rogaway. Authenticated Key Exchange Secure Against Dictionary Attacks. In *Eurocrypt '2000*, LNCS 1807, pages 139–155. Springer-Verlag, Berlin, 2000.

[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. Entity Authentication and Key Distribution. In *Crypto '93*, LNCS 773, pages 232–249. Springer-Verlag, Berlin, 1994.

[6] M. Bellare and P. Rogaway. Provably Secure Session Key Distribution: the Three Party Case. In *Proc. of the 27th STOC*. ACM Press, New York, 1995.

[7] 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

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

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

[10] J.-S. Coron. On the Exact Security of Full-Domain-Hash. In *Crypto '2000*, LNCS 1880, pages 229–235. Springer-Verlag, Berlin, 2000.

[11] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT–22(6):644–654, November 1976.

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

[13] 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.

[14] 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.

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

[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] S. Goldwasser, S. Micali, and R. Rivest. A "Paradoxical" Solution to the Signature Problem. In *Proc. of the 25th FOCS*, pages 441–448. IEEE, New York, 1984.

[19] S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptative Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, April 1988.

[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] 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.

[24] 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.

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

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

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

[28] 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.

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

[30] 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.

[31] V. Shoup. On Formal Models for Secure Key Exchange. Technical Report RZ 3120, IBM Research, April 1999.

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

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

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