# On the Unpredictability of Bits of the Elliptic Curve Diffie–Hellman Scheme

Dan Boneh[1]⋆ and Igor E. Shparlinski[2]⋆⋆

[1] Department of Computer Science, Stanford University, CA, USA
dabo@cs.stanford.edu
[2] Department of Computing, Macquarie University, Sydney, NSW 2109, Australia
igor@comp.mq.edu.au

**Abstract.** Let $\mathbb{E}/\mathbb{F}_p$ be an elliptic curve, and $G \in \mathbb{E}/\mathbb{F}_p$. Define the Diffie–Hellman function as $\mathsf{DH}_{\mathbb{E},G}(aG, bG) = abG$. We show that if there is an efficient algorithm for predicting the LSB of the $x$ or $y$ coordinate of $abG$ given $\langle \mathbb{E}, G, aG, bG \rangle$ for a certain family of elliptic curves, then there is an algorithm for computing the Diffie–Hellman function on all curves in this family. This seems stronger than the best analogous results for the Diffie–Hellman function in $\mathbb{F}_p^*$. Boneh and Venkatesan showed that in $\mathbb{F}_p^*$ computing approximately $(\log p)^{1/2}$ of the bits of the Diffie–Hellman secret is as hard as computing the entire secret. Our results show that just predicting one bit of the Elliptic Curve Diffie–Hellman secret in a family of curves is as hard as computing the entire secret.

## 1 Introduction

We recall how the Diffie–Hellman key exchange scheme works in an arbitrary finite cyclic group $\mathcal{G}$ of order $T$. Let $g$ be a generator $g$ of $\mathcal{G}$. Then to establish a common key, two communicating parties, *Alice* and *Bob* execute the following protocol, see [15, 25]: *Alice* chooses a random integer $x \in [1, T-1]$, computes and sends $X = g^x$ to *Bob*. *Bob* chooses a random integer $y \in [1, T-1]$, computes and sends $Y = g^y$ to *Alice*. Now both *Alice* and *Bob* can compute the common *Diffie–Hellman secret*

$$K = Y^x = X^y = g^{xy}.$$

The Computational Diffie–Hellman assumption (CDH) in the group $\mathcal{G}$ states that no efficient algorithm can compute $g^{xy}$ given $g, g^x, g^y$. However, this does not mean that one cannot compute a few bits of $g^{xy}$ or perhaps predict some bits of $g^{xy}$. In fact, to use the Diffie–Hellman protocol in an efficient system one usually relies on the stronger Decision Diffie–Hellman assumption (DDH) [3]. Ideally, one would like to show than an algorithm for DDH in the group $\mathcal{G}$ implies an algorithm for CDH in $\mathcal{G}$. As a first step we show that, in the group of points of an elliptic curve over a finite field, predicting the least significant bit ($\mathsf{LSB}$)

⋆ Supported by NSF and the Packard Foundation.
⋆⋆ Supported in part by ARC

of the Diffie–Hellman secret, for many curves in a family of curves, is as hard as computing the entire secret. Such results were previously known for the RSA function [1, 7] but not for Diffie-Hellman.

Let $p$ be prime and let $\lfloor s \rfloor_p$ denote the remainder of an integer $s$ on division by $p$. We also use $\log z$ to denote the binary logarithm of $z > 0$. In the classical settings $\mathcal{G}$ is selected as the multiplicative group $\mathbb{F}_p^*$ of a finite field of $p$ elements (and thus $g$ is a primitive root of $\mathbb{F}_p$). In this case, Boneh and Venkatesan [5] showed that about $\log^{1/2} p$ most significant bits of $\lfloor g^{xy} \rfloor_p$ are as hard to find as $\lfloor g^{xy} \rfloor_p$ itself. The result is based on lattice reduction techniques. A similar result holds for the least significant bits as well. González Vasco and Shparlinski [10] used exponential sums to extend this result to subgroups $\mathcal{G}$ of $\mathbb{F}_p^*$. It has turned out that the lattice reduction technique used in [5] coupled with the exponential sum technique lead to a series of new results about the bits security of some cryptographic constructions [11, 14, 22, 23] as well as to attacks on some of them [6, 13, 17, 18].

However the case where $\mathcal{G}$ is the point group of an elliptic curve has turned out to be much harder for applications of the lattice reduction based technique of [5] because of the inherited nonlinearity of the problem. Although some results have recently been obtained in [4] they are much weaker that those known for subgroups of $\mathbb{F}_p^*$. Here, using a very different technique, we show that working with a certain family of isomorphic curves (rather than with one fixed curve) allows to obtain results that are stronger than those known for subgroups of $\mathbb{F}_p^*$. By using certain twists of the given curve we show that predicting the least significant bit of the elliptic curve Diffie–Hellman secret in a family of curves is as hard as computing the entire secret. Since our techniques work with many curves at once they do not extend to the case of subgroups of $\mathbb{F}_p^*$.

## 2 Elliptic Curve Diffie–Hellman Scheme

Throughout the paper we let $p$ be a prime and let $\mathbb{F}_p$ be the finite field of size $p$. Let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$, given by an affine Weierstrass equation of the form

$$Y^2 = X^3 + AX + B, \qquad 4A^3 + 27B^2 \neq 0 \tag{1}$$

It is known [24] that the set $\mathbb{E}(\mathbb{F}_p)$ of $\mathbb{F}_p$-rational points of $\mathbb{E}$ form an Abelian group under an appropriate composition rule and with the point at infinity $\mathcal{O}$ as the neutral element. We also recall that

$$|N - p - 1| \le 2p^{1/2},$$

where $N = |\mathbb{E}(\mathbb{F}_p)|$ is the number of $\mathbb{F}_p$-rational points, including the point at infinity $\mathcal{O}$.

Let $G \in \mathbb{E}$ be a point of order $q$, that is, $q$ is the size of the cyclic group generated by $G$. Then the common key established at the end of the Diffie–Hellman protocol with respect to the curve $\mathbb{E}$ and the point $G$ is $abG = (x, y) \in \mathbb{E}$ for some integers $a, b \in [1, q - 1]$.

Throughout the paper we use the fact that the representation of $\mathbb{E}$ contains the field of definition of $\mathbb{E}$. With this convention, an algorithm given the representation of $\mathbb{E}/\mathbb{F}_p$ as input does not need to also be given $p$. The algorithm obtains $p$ from the representation of $\mathbb{E}$.

*Diffie-Hellman Function:* Let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$ and let $G \in \mathbb{E}$ be a point of prime order $q$. We define the Diffie-Hellman function as:

$$\mathsf{DH}_{\mathbb{E},G}(aG, bG) = abG$$

where $a, b$ are integers in $[1, q-1]$. The Diffie-Hellman problem on $\mathbb{E}$ is to compute $\mathsf{DH}_{\mathbb{E},G}(P, Q)$ given $\mathbb{E}, G, P, Q$. Clearly we mostly focus on curves in which the Diffie-Hellman problem is believed to be hard. Throughout we say that a randomized algorithm $\mathcal{A}$ computes the Diffie-Hellman function if $\mathcal{A}(\mathbb{E}, G, aG, bG) = abG$ holds with probability at least $1 - 1/p$. The probability is over the random bits used by $\mathcal{A}$.

*Twists on elliptic curves:* Let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$ given by the *Weierstrass equation* $y^2 = x^3 + Ax + B$. Our proofs rely on using certain *twists* of the elliptic curve. For $\lambda \in \mathbb{F}_p^*$ define $\phi_\lambda(\mathbb{E})$ to be the (twisted) elliptic curve:

$$Y^2 = X^3 + A\lambda^4 X + B\lambda^6. \tag{2}$$

We remark that $4(A\lambda^4)^3 + 27(B\lambda^6)^2 = (4A^3 + 27B^2)\lambda^{12} \neq 0$ for $\lambda \in \mathbb{F}_p^*$. Hence, $\phi_\lambda(\mathbb{E})$ is an elliptic curve for any $\lambda \in \mathbb{F}_p^*$. Throughout the paper we are working with the family of curves $\{\phi_\lambda(\mathbb{E}_0)\}_{\lambda \in \mathbb{F}_p^*}$ associated with a given curve $\mathbb{E}_0$.

It is easy to verify that for any point $P = (x, y) \in \mathbb{E}$ and any $\lambda \in \mathbb{F}_p^*$ the point $P_\lambda = (x\lambda^2, y\lambda^3) \in \phi_\lambda(\mathbb{E})$. Moreover, from the explicit formulas for the group law on $\mathbb{E}$ and $\phi_\lambda(\mathbb{E})$, see [2, 24], we derive that for any points $P, Q, R \in \mathbb{E}$ with $P + Q = R$ we also have $P_\lambda + Q_\lambda = R_\lambda$. In particular, for any $G \in \mathbb{E}$ we have:

$$xG_\lambda = (xG)_\lambda, \qquad yG_\lambda = (yG)_\lambda, \qquad xyG_\lambda = (xyG)_\lambda.$$

Hence, the map $\phi_\lambda : \mathbb{E} \to \phi_\lambda(\mathbb{E})$ mapping $P \in \mathbb{E}$ to $P_\lambda \in \phi_\lambda(\mathbb{E})$ is a homomorphism. In fact, it is easy to verify that $\phi_\lambda$ is an isomorphism of groups. This means that

$$\mathsf{DH}_{\phi_\lambda(\mathbb{E}),G_\lambda}(P_\lambda, Q_\lambda) = \phi_\lambda[\mathsf{DH}_{\mathbb{E},G}(P, Q)].$$

Hence, if the Diffie-Hellman function is hard to compute in $\mathbb{E}$ then it is also hard to compute for all curves in $\{\phi_\lambda(\mathbb{E})\}_{\lambda \in \mathbb{F}_p^*}$.

## 3 Main Results

We denote by $\mathsf{LSB}(z)$ the *least significant bit* of an integer $z \geq 0$. When $z \in \mathbb{F}_p$ we let $\mathsf{LSB}(z)$ be $\mathsf{LSB}(x)$ for the unique integer $x \in [0, p-1]$ such that $x \equiv z \bmod p$.

Let $p$ be a prime, and let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$. Let $G \in \mathbb{E}$ be a point of order $q$, for some prime $q$. We say that an algorithm $\mathcal{A}$ has advantage $\epsilon$ in predicting the $\mathsf{LSB}$ of the $x$-coordinate of the Diffie-Hellman function on $\mathbb{E}$ if:

$$\mathrm{Adv}^X_{\mathbb{E},G}(\mathcal{A}) = \left| \Pr_{a,b}[A(\mathbb{E}, G, aG, bG) = \mathsf{LSB}(x)] - \frac{1}{2} \right| > \varepsilon$$

where $abG = (x, y) \in \mathbb{E}$ and $a, b$ are chosen uniformly at random in $[1, q-1]$. We write $\mathrm{Adv}^X_{\mathbb{E},G}(\mathcal{A}) > \varepsilon$. Similarly, we say that algorithm $\mathcal{A}$ has advantage $\epsilon$ in predicting the $\mathsf{LSB}$ of the $y$-coordinate of the Diffie-Hellman function if:

$$\mathrm{Adv}^Y_{\mathbb{E},G}(\mathcal{A}) = \left| \Pr_{a,b}[A(\mathbb{E}, G, aG, bG) = \mathsf{LSB}(y)] - \frac{1}{2} \right| > \varepsilon$$

where $abG = (x, y) \in \mathbb{E}$. We write $\mathrm{Adv}^Y_{\mathbb{E},G}(\mathcal{A}) > \varepsilon$.

The following result shows that no algorithm can have a non-negligible advantage in predicting the $\mathsf{LSB}$ of the $x$ or $y$ coordinates of the Diffie-Hellman secret for many curves in $\{\phi_\lambda(\mathbb{E}_0)\}_{\lambda \in \mathbb{F}_p^*}$, unless the Diffie–Hellman problem is easy on $\mathbb{E}_0$.

**Theorem 1.** *Let $\epsilon, \delta \in (0, 1)$. Let $p$ be a prime, and let $\mathbb{E}_0$ be an elliptic curve over $\mathbb{F}_p$. Let $G \in \mathbb{E}_0$ be a point of prime order. Suppose there is a $t$-time algorithm $\mathcal{A}$ such that either:*

1. *$Adv^X_{\phi_\lambda(\mathbb{E}_0),\phi_\lambda(G)}(\mathcal{A}) > \varepsilon$ for at least a $\delta$-fraction of the $\lambda \in \mathbb{F}_p^*$, or*
2. *$Adv^Y_{\phi_\lambda(\mathbb{E}_0),\phi_\lambda(G)}(\mathcal{A}) > \varepsilon$ for at least a $\delta$-fraction of the $\lambda \in \mathbb{F}_p^*$.*

*Then the Diffie–Hellman function $\mathsf{DH}_{\mathbb{E}_0,G}(P, Q)$ can be computed in expected time $t \cdot T(\log p, \frac{1}{\varepsilon\delta})$ where $T$ is some fixed polynomial independent of $p$ and $\mathbb{E}_0$.*

Theorems 1 shows that, if the Diffie-Hellman problem is hard in $\mathbb{E}_0$, then no efficient algorithm can predict the least significant bit of the $X$ or $Y$ coordinates of the Diffie–Hellman function for a non-negligible fraction of the curves in $\{\phi_\lambda(\mathbb{E}_0)\}_{\lambda \in \mathbb{F}_p^*}$. The proof of Theorem 1 is given in Section 6. Note the theorem does not give a curve in $\{\phi_\lambda(\mathbb{E}_0)\}_{\lambda \in \mathbb{F}_p^*}$ for which the $\mathsf{LSB}$ of the $X$ coordinate is a hard-core bit — it can still be the case that for every curve $\mathbb{E} \in \{\phi_\lambda(\mathbb{E}_0)\}_{\lambda \in \mathbb{F}_p^*}$ there is an efficient algorithm that predicts the $\mathsf{LSB}$ of $\mathsf{DH}_{\mathbb{E},G}$ for that curve only. However, there cannot be a single efficient algorithm that predicts this $\mathsf{LSB}$ for a non-negligible fraction of the curves in $\{\phi_\lambda(\mathbb{E}_0)\}_{\lambda \in \mathbb{F}_p^*}$.

An immediate corollary of Theorem 1 gives a hard core predicate for a simple extension of the Diffie-Hellman function. Let $\overline{\mathsf{DH}}_{\mathbb{E},G}$ be the function:

$$\overline{\mathsf{DH}}_{\mathbb{E},G}(P, Q, \lambda) = \mathsf{DH}_{\phi_\lambda(\mathbb{E}),G_\lambda}(P_\lambda, Q_\lambda)$$

where $G_\lambda = \phi_\lambda(G)$ and similarly $P_\lambda, Q_\lambda$. Note that this function basically uses $\lambda$ as an index indicating in which group to execute the Diffie-Hellman protocol. Then the $\mathsf{LSB}$ of the $X$ or $Y$ coordinates is a hard-core bit of this function assuming the Diffie-Hellman problem is hard in $\mathbb{E}$.

**Corollary 1.** *Let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$ and let $G \in \mathbb{E}$ be of prime order $q$. Suppose there is a $t$-time algorithm $\mathcal{A}$ such that*

$$\Pr_{a,b,\lambda}[\mathcal{A}(\mathbb{E}, G, aG, bG, \lambda) = \mathsf{LSB}(x)] > \frac{1}{2} + \varepsilon$$

*where $\overline{\mathsf{DH}}_{\mathbb{E},G}(aG, bG, \lambda) = (x, y) \in \phi_\lambda(\mathbb{E})$. Here $a, b$ are uniformly chosen in $[1, q-1]$ and $\lambda \in \mathbb{F}_p^*$. Then the Diffie–Hellman function $\mathsf{DH}_{\mathbb{E}_0,G}$ can be computed in expected time $t \cdot T(\log p, \frac{1}{\varepsilon})$ where $T$ is some fixed polynomial independent of $p$ and $\mathbb{E}_0$.*

We note that there are other ways of extending the Diffie-Hellman function to obtain a hard-core bit [8, 12].

## 4 Review of the ACGS algorithm

The proof of Theorem 1 uses an algorithm due to Alexi, Chor, Goldreich, and Schnorr [1]. We refer to this algorithm as the ACGS algorithm. For completeness, we briefly review the algorithm here. First, we define the following variant of the Hidden Number Problem (HNP) presented in [5].

$\mathsf{HNP\text{-}CM}$*:* Fix an $\varepsilon > 0$. Let $p$ be a prime. For an $\alpha \in \mathbb{F}_p$ let $L : \mathbb{F}_p^* \to \{0, 1\}$ be a function satisfying

$$\Pr_{t \in \mathbb{F}_p^*}\left[L(t) = \mathsf{LSB}\left(\lfloor \alpha \cdot t \rfloor_p\right)\right] \geq \frac{1}{2} + \varepsilon. \tag{3}$$

The $\mathsf{HNP\text{-}CM}$ problem is: given an oracle for $L(t)$, find $\alpha$ in polynomial time (in $\log p$ and $1/\epsilon$). Clearly we wish to show an algorithm for this problem that works for the smallest possible $\varepsilon$. For small $\epsilon$ there might be multiple $\alpha$ satisfying condition (3) (polynomially many in $\varepsilon^{-1}$). In this case the list-$\mathsf{HNP\text{-}CM}$ problem is to find the list of all such $\alpha \in \mathbb{F}_p$. Note that it is easy to verify that a given $\alpha$ belongs to the list of solutions by picking polynomially many random samples $x \in \mathbb{F}_p$ (say, $O(1/\varepsilon^2)$ samples suffice) and testing that $L(x) = \mathsf{LSB}(\lfloor \alpha x \rfloor_p)$ holds sufficiently often.

We refer to the above problem as $\mathsf{HNP\text{-}CM}$ to denote the fact that we are free to evaluate $L(t)$ at any multiplier $t$ of our choice (the $\mathsf{CM}$ stands for Chosen Multiplier). In the original $\mathsf{HNP}$ studied in [5] one is only given samples $(t, L(t))$ for random $t$. The following theorem shows how to solve the $\mathsf{HNP\text{-}CM}$ for any $\varepsilon > 0$. The proof of the theorem (using different terminology) can be found in [1] and [7].

**Theorem 2 (ACGS).** *Let $p$ be an $n$-bit prime and let $\varepsilon > 0$. Then, given $\varepsilon$, the list-*$\mathsf{HNP\text{-}CM}$ *problem can be solved in expected polynomial time in $n$ and $1/\varepsilon$.*

**Proof Sketch** For $\alpha \in \mathbb{F}_p^*$ let $f_\alpha(t) : \mathbb{F}_p \to \{0, 1\}$ be a function such that $f_\alpha(t) = \mathsf{LSB}\left(\lfloor \alpha t \rfloor_p\right)$ for all $t \in \mathbb{F}_p$. It is well known that given an oracle for

$f_\alpha(t)$ it is possible to recover $\alpha$ using polynomially many queries (polynomial in $\log p$). See [1, 7] or Theorem 7 of [5]. In fact, using the method of [1], it suffices to make queries only at $t$ for which $\lfloor t\alpha \rfloor_p < p \cdot \varepsilon/2$ (as a result the run time is polynomial in $\log p$ and $1/\varepsilon$). Hence, the main challenge is in building an oracle for $f_\alpha(t)$ from an oracle for $L(t)$. The ACGS algorithm constructs an oracle for $f_\alpha(t)$ for every $\alpha \in \mathbb{F}_p^*$ that satisfies the condition (3). This construction is at the heart of the ACSG algorithm.

Let $m = n \cdot \frac{1}{\varepsilon^2}$. We show how to evaluate $f_\alpha(t)$ given an oracle for the function $L(t)$. We first pick random $u, v \in \mathbb{F}_p$. We use the same $u, v$ to answer all queries to $f_\alpha(t)$. We assume that we know the $2\log m$ most significant bits and the least significant bit of $\lfloor u\alpha \rfloor_p, \lfloor v\alpha \rfloor_p$. This assumption is valid since we intend to run the ACGS algorithm with all possible values for these $2 + \lceil 4\log m \rceil$ bits. In one of these iterations we obtain the correct values for the $2 + \lceil 4\log m \rceil$ most significant bits and least significant bit of $\lfloor u\alpha \rfloor_p, \lfloor v\alpha \rfloor_p$. Note that different guesses for these bits will lead to oracles for $f_\alpha(t)$ for different values of $\alpha$.

For $i = 1, \ldots, m$ let $r_i = \lfloor iu + v \rfloor_p$. Then $r_1, \ldots, r_m$ are pair wise independent values in $\mathbb{F}_p$ (over the choice of $u, v$). One can easily show (as in [1, 7]) that using the knowledge of the most significant bits of $u\alpha, v\alpha \bmod p$ and the least significant bit, it is easy to determine $b_i = \mathsf{LSB}(\lfloor r_i\alpha \rfloor_p)$ for $i = 1, \ldots, m$. Therefore, to evaluate $f_\alpha(t)$ do the following:

1. Evaluate $a_i = L(t + r_i)$. Set $f_i = a_i \oplus b_i$, for $i = 1, \ldots, m$, where $\oplus$ denotes addition modulo 2.
2. Respond with $f_\alpha(t) = \text{Majority}(f_1, \ldots, f_m)$.

For a given $i \in [1, m]$ we say that $a_i$ is correct if $a_i = \mathsf{LSB}(\lfloor \alpha(t + r_i) \rfloor_p)$. Recall that we only make $f_\alpha(t)$ queries at $t$ satisfying $\lfloor t\alpha \rfloor_p < p \cdot \varepsilon/2$. Therefore, $\lfloor \alpha(t + r_i) \rfloor_p = \lfloor \alpha t \rfloor_p + \lfloor \alpha r_i \rfloor_p$, as integers, with probability at least $1 - \varepsilon/2$. Then $\mathsf{LSB}(\lfloor \alpha(t + r_i) \rfloor_p) = \mathsf{LSB}(\lfloor \alpha t \rfloor_p) \oplus \mathsf{LSB}(\lfloor \alpha r_i \rfloor_p)$. It follows that if $a_i$ is correct then $f_i = \mathsf{LSB}(\lfloor t\alpha \rfloor_p)$ with probability at least $1 - \varepsilon/2$.

Since each $r_i$ is uniformly distributed in $\mathbb{F}_p$ (over the choice of $u, v$) it follows that each $a_i$ is correct with probability at least $\frac{1}{2} + \epsilon$. Since the $r_i$'s are pair wise independent it follows that the $f_i$'s are pair wise independent. Therefore, by Chebychev's inequality we obtain the correct value of $f_\alpha(t)$ with probability $1 - 1/n$. The exact analysis is given in [1]. Since we are able to construct an almost perfect subroutine for $f_\alpha(t)$ for all $\alpha$ satisfying the condition (3) the ACGS algorithm will produce a polynomial (in $\log p$) length list of candidates containing all required $\alpha$. Note that it is easy to verify that a given $\alpha$ in the resulting list satisfies the condition (3) by picking polynomially many random samples $x \in \mathbb{F}_p$ and testing that $L(x) = \mathsf{LSB}(\lfloor \alpha x \rfloor_p)$ holds sufficiently often. $\square$

We note that Fischlin and Schnorr [7] presented a more efficient algorithm for the $\mathsf{HNP\text{-}CM}$. They rely on sub-sampling in Step 2 above to reduce the number of queries to the oracle for $L$.

## 5 Quadratic and Cubic Hidden Number Problems

To prove the main results of Section 3 we actually need an algorithm for the following variant of the $\mathsf{HNP\text{-}CM}$ problem.

$\mathsf{HNP\text{-}CM}^d$*:* Fix an integer $d > 0$ and an $\varepsilon > 0$. Let $p$ be a prime. For an $\alpha \in \mathbb{F}_p^*$ let $L^{(d)} : \mathbb{F}_p^* \to \{0,1\}$ be a function satisfying

$$\Pr_{t \in \mathbb{F}_p^*}\left[L^{(d)}(t) = \mathsf{LSB}\left(\lfloor \alpha t^d \rfloor_p\right)\right] \geq \frac{1}{2} + \varepsilon. \tag{4}$$

The $\mathsf{HNP\text{-}CM}^d$ problem is: given an oracle for $L^{(d)}(t)$, find $\alpha$ in polynomial time. For small $\varepsilon$ there might be multiple $\alpha$ satisfying condition (4) (polynomially many in $\varepsilon^{-1}$). In this case the list-$\mathsf{HNP\text{-}CM}^d$ problem is to find all such $\alpha \in \mathbb{F}_p^*$. We prove the following simple result regarding the list-$\mathsf{HNP\text{-}CM}^d$ problem. We use this theorem for $d = 2$ and $d = 3$.

**Theorem 3.** *Fix an integer $d > 1$. Let $p$ be a $n$-bit prime and let $\varepsilon > 0$. Then, given $\varepsilon$, the $\mathsf{HNP\text{-}CM}^d$ problem can be solved in expected polynomial time in $\log p$ and $d/\varepsilon$.*

*Proof.* Let $L^{(d)}$ be a function satisfying the condition (4). Let $R : \mathbb{F}_p \to \{0,1\}$ be a random function chosen uniformly from the set of all functions from $\mathbb{F}_p$ to $\{0,1\}$. Let $S : \mathbb{F}_p^d \to \mathbb{F}_p$ be a function satisfying $S(x)^d \equiv x \bmod p$ for all $x \in \mathbb{F}_p^d$ and chosen at random from the set of such functions. Here $\mathbb{F}_p^d$ is the set of $d$'th powers in $\mathbb{F}_p$. The function $S$ is simply a function mapping a $d$'th power $x \in \mathbb{F}_p^d$ to a randomly chosen $d$'th root of $x$. Next, define the following function $L(t)$:

$$L(t) = \begin{cases} L^{(d)}(S(t)) & \text{if } t \in \mathbb{F}_p^d, \\ R(t) & \text{otherwise.} \end{cases}$$

We claim that for any $\alpha \in \mathbb{F}_p^*$ satisfying the condition (4) we have that $L(t)$ satisfies

$$\Pr_{t,R,S}\left[L(t) = \mathsf{LSB}\left(\lfloor \alpha \cdot t \rfloor_p\right)\right] \geq \frac{1}{2} + \varepsilon/d.$$

To see this, fix an $\alpha \in \mathbb{F}_p$ satisfying the condition (4). Let $\mathcal{B}_t$ be the event that $L(t) = \mathsf{LSB}\left(\lfloor \alpha \cdot t \rfloor_p\right)$. Let $\mathcal{B}_t^d$ be the event that $L^{(d)}(t) = \mathsf{LSB}\left(\lfloor \alpha \cdot t^d \rfloor_p\right)$. Observe that if $t$ is uniform in $\mathbb{F}_p^d \setminus \{0\}$ then $S(t)$ is uniform in $\mathbb{F}_p^*$. Let $e = \gcd(p-1, d)$.

If $e = 1$ then $\mathbb{F}_p = \mathbb{F}_p^d$ and therefore:

$$\Pr_{t,R,S}[\mathcal{B}_t] = \Pr_{t,R,S}\left[\mathcal{B}_{S(t)}^d\right] = \Pr_{x \in \mathbb{F}_p^*}\left[\mathcal{B}_x^d\right] \geq \frac{1}{2} + \varepsilon.$$

Hence, in this case the claim is correct. When $e > 1$ then the size of $\mathbb{F}_p^d \setminus \{0\} = \mathbb{F}_p^e \setminus \{0\}$ is $\frac{p-1}{e}$. Therefore:

$$\begin{aligned}
\Pr_{t,R,S}[\mathcal{B}_t] &= \frac{1}{e} \Pr_{t,R,S}\left[\mathcal{B}_t \mid t \in \mathbb{F}_p^d\right] + \left(1 - \frac{1}{e}\right) \Pr_{t,R,S}\left[\mathcal{B}_t \mid t \notin \mathbb{F}_p^d\right] \\
&= \frac{1}{e} \Pr_{t,R,S}\left[\mathcal{B}_{S(t)}^d \mid t \in \mathbb{F}_p^e\right] + \left(1 - \frac{1}{e}\right) \cdot \frac{1}{2} \\
&\geq \frac{1}{e}\left(\frac{1}{2} + \varepsilon\right) + \left(1 - \frac{1}{e}\right) \cdot \frac{1}{2} = \frac{1}{2} + \frac{\varepsilon}{e} \geq \frac{1}{2} + \frac{\varepsilon}{d}
\end{aligned}$$

and hence the claim holds in this case as well.

We see that an oracle for $L^{(d)}$ with advantage $\epsilon$ immediately gives rise to an oracle for $L$ with advantage $\epsilon/d$. Hence, we can use the ACGS algorithm to find the list of solutions to the given $\mathsf{HNP\text{-}CM}^d$ problem. When the ACGS algorithm runs we build the functions $R$ and $S$ as they are needed to respond to ACGS's queries to $L$. The ACGS algorithm will produce a super set of the solution set to the list-$\mathsf{HNP\text{-}CM}^d$ within the required time bound. Note that we may need to prune some of the solutions produced by the ACGS algorithm: we only output the $\alpha$'s for which the condition (4) holds. $\square$

## 6 Proof of Main Results

We are now ready to prove Theorem 1. The proof reduces the problem of computing the Diffie–Hellman function to the Hidden Number Problem described in Section 5. We also use the following two simple lemmas. For a curve $\mathbb{E}/\mathbb{F}_p$ and $G \in \mathbb{E}$ of order $q$ define:

$$F_{\mathbb{E},G,\lambda}(\mathcal{B}) = \Pr_{a,b}[\mathcal{B}(\phi_\lambda(\mathbb{E}), \phi_\lambda(G), \phi_\lambda(aG), \phi_\lambda(bG)) = \mathsf{LSB}(x_\lambda)]$$

where $\phi_\lambda(abG) = (x_\lambda, y_\lambda) \in \phi_\lambda(\mathbb{E})$ and $a, b$ are uniform in $[1, q-1]$. Note that the probability space includes the random bits used by $\mathcal{B}$.

**Lemma 1.** *Let $p$ be a prime, and let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$. Let $G \in \mathbb{E}$. Suppose there is a $t$-time algorithm $\mathcal{A}$ such that $Adv^X_{\phi_\lambda(\mathbb{E}),\phi_\lambda(G)}(\mathcal{A}) > \varepsilon$ for at least a $\delta$-fraction of the $\lambda \in \mathbb{F}_p^*$.*
*Then, given $\epsilon, \delta$, there is a $t'$-time algorithm $\mathcal{B}$ such that:*
*(1) for at least a $\delta$-fraction of the $\lambda \in \mathbb{F}_p^*$ we have that: $F_{\mathbb{E},G,\lambda}(\mathcal{B}) > \frac{1}{2} + \epsilon/2$, and*
*(2) for the remaining $\lambda \in \mathbb{F}_p^*$ we have that: $F_{\mathbb{E},G,\lambda}(\mathcal{B}) > \frac{1}{2} - \frac{\varepsilon\delta}{4}$*
*Furthermore, $t' = t \cdot T(1/\epsilon\delta)$ for some fixed polynomial $T$ independent of $p, \mathbb{E}$.*

*Proof.* On input $\langle \mathbb{E}, G, P, Q \rangle$ algorithm $\mathcal{B}$ works as follows:
1. Pick $u = (4/\epsilon\delta)^3$ random $a, b \in [1, q-1]$ pairs and run $\mathcal{A}$ on all tuples $\langle \mathbb{E}, G, aG, bG \rangle$.

2. let $v$ be the number of runs in which $\mathcal{A}$ correctly outputs $\mathsf{LSB}((abG)_x)$.
3. if $v > u/2$ then $\mathcal{B}$ outputs $\mathcal{A}(\mathbb{E}, G, P, Q)$, otherwise $\mathcal{B}$ output the complement of $\mathcal{A}(\mathbb{E}, G, P, Q)$.

Let $\tau \geq \epsilon\delta/4$. For all $\lambda \in \mathbb{F}_p^*$ for which $\mathrm{Adv}^X_{\phi_\lambda(\mathbb{E}),\phi_\lambda(G)}(\mathcal{A}) > \tau$ we have that $\mathcal{B}$ satisfies: $F_{\mathbb{E},G,\lambda}(\mathcal{B}) > \frac{1}{2} + \tau/2$. This follows directly from Chebychev's inequality. For all other $\lambda$'s, by definition of $\mathrm{Adv}(\mathcal{A})$ we have $F_{\mathbb{E},G,\lambda}(\mathcal{B}) > \frac{1}{2} - \epsilon\delta/4$. Hence, both conditions 1 and 2 are satisfied. □

**Lemma 2.** *Let $\mathcal{B}$ be an algorithm satisfying the two conditions of Lemma 1. Then*

$$\Pr_{\lambda \in \mathbb{F}_p^*}[\mathcal{B}(\phi_\lambda(\mathbb{E}), \phi_\lambda(G), \phi_\lambda(aG), \phi_\lambda(bG)) = \mathsf{LSB}(x_\lambda)] \geq \frac{1}{2} + \frac{\varepsilon\delta}{4}$$

*holds with probability at least $\frac{\varepsilon\delta}{8}$ over the choice of $a, b \in [1, q-1]$, where $\phi_\lambda(abG) = (x_\lambda, y_\lambda)$.*

*Proof.* The proof uses a standard counting argument. Algorithm $\mathcal{B}$ induces a matrix $M$ whose entries are real numbers in $[0,1]$. There is a column in $M$ for every $\lambda \in \mathbb{F}_p^*$ and a row for every $(a,b) \in [1, q-1]^2$. The entry at the $\lambda$'th column and $(a,b)$'th row is simply

$$\Pr\left[\mathcal{B}(\phi_\lambda(\mathbb{E}), \phi_\lambda(G), \phi_\lambda(aG), \phi_\lambda(bG)) = \mathsf{LSB}(x_\lambda)\right].$$

The probability is over the random bits used by $\mathcal{B}$. Suppose the matrix $M$ has $n$ columns and $m$ rows. Since $\mathcal{B}$ satisfies the two condition of Lemma 1 we know that the sum of all the entries in $M$, which we call the weight of $M$ denote by $\mathrm{weight}(M)$ is at least

$$\mathrm{weight}(M) > nm\left[\delta\left(\frac{1}{2} + \frac{\varepsilon}{2}\right) + (1-\delta)\left(\frac{1}{2} - \frac{\varepsilon\delta}{4}\right)\right] > nm\left(\frac{1}{2} + \frac{\delta\varepsilon}{4}\right).$$

Let $R$ be the number of the rows in $M$ must have weight at least $n[\frac{1}{2} + \frac{\varepsilon\delta}{8}]$ (the weight of a row is the sum of the entries in that row). We have

$$Rn + (m-R)n\left[\frac{1}{2} + \frac{\varepsilon\delta}{8}\right] \geq \mathrm{weight}(M) > nm\left(\frac{1}{2} + \frac{\delta\varepsilon}{4}\right).$$

Therefore

$$R\left[\frac{1}{2} - \frac{\varepsilon\delta}{8}\right] > \frac{\varepsilon\delta}{8}m.$$

The result now follows. □

We also need to review a theorem due to Shoup (Theorem 7 of [21]). The theorem shows that an algorithm that outputs a list of candidates for the Diffie-Hellman function can be easily converted into an algorithm that computes the Diffie-Hellman function. For concreteness we state the theorem as it applies to elliptic curves over $\mathbb{F}_p$.

**Theorem 4 (Shoup).** *Let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$ and let $G \in \mathbb{E}$ be an element of prime order $q$. Suppose there is a $t$-time algorithm $\mathcal{A}$ that given $aG, bG \in \mathbb{E}$ outputs a set of size $m$ satisfying $\mathsf{DH}_{\mathbb{E},G}(aG, bG) \in A(\mathbb{E}, G, aG, bG)$ with probability at least 7/8. Then there is an algorithm $\mathcal{B}$ that computes the Diffie-Hellman function in $\mathbb{E}$ in time $t' = t(\log p) + T(m, \log p)$. Here $T$ is a fixed polynomial independent of $p$ and $\mathbb{E}$.*

*Proof of Theorem 1:* Let $\mathbb{E}$ be a curve over $\mathbb{F}_p$ and $G \in \mathbb{E}$ of prime order $q$. Suppose there is an expected $t$-time algorithm $\mathcal{A}$ such that $\mathrm{Adv}^X_{\phi_\lambda(\mathbb{E}),\phi_\lambda(G)}(\mathcal{A}) > \varepsilon$ for at least a $\delta$-fraction of the $\lambda \in \mathbb{F}_p^*$. We show how to compute the Diffie–Hellman function $\mathsf{DH}_{\mathbb{E},G}$.

We are given $A = aG$ and $B = bG$ in $\mathbb{E}$. We wish to compute the point $C = abG \in \mathbb{E}$. We first randomize the problem by computing $A' = a_0 A$ and $B' = b_0 B$ for random $a_0, b_0 \in [1, q-1]$. If $C' = \mathsf{DH}_{\mathbb{E},G}(A', B')$ then $C = c_0 C'$ where $c_0 \equiv (a_0 b_0)^{-1} \bmod q$. Hence, it suffices to find $C'$. Write $C' = (x_0, y_0)$.

Since $\phi_\lambda : \mathbb{E} \to \phi_\lambda(\mathbb{E})$ is an isomorphism it follows that

$$\mathsf{DH}_{\phi_\lambda(\mathbb{E}),\phi_\lambda(G)}(\phi_\lambda(A'), \phi_\lambda(B')) = \phi_\lambda(C') = (\lambda^2 x_0, \lambda^3 y_0).$$

Since $A', B'$ are uniformly distributed in the group generated by $G$ (excluding $\mathcal{O}$) we can apply both Lemma 1 and Lemma 2 to obtain an algorithm $\mathcal{B}$ satisfying:

$$\Pr_\lambda[\mathcal{B}(\phi_\lambda(\mathbb{E}), \phi_\lambda(G), \phi_\lambda(A'), \phi_\lambda(B')) = \mathsf{LSB}(\lambda^2 x_0)] > \frac{1}{2} + \frac{\varepsilon\delta}{8} \tag{5}$$

is true with probability at least $\varepsilon\delta/8$ over the choice of $a_0, b_0$ in $[1, q-1]$.

For now we assume that (5) holds. We obtain an $\mathsf{HNP}\text{-}\mathsf{CM}^2$ problem where $x_0$ is the hidden number. To see this, define:

$$L^{(2)}(\lambda) = \mathcal{A}(\phi_\lambda(\mathbb{E}), \phi_\lambda(G), \phi_\lambda(A'), \phi_\lambda(B')).$$

Then the condition 5 implies that $\Pr_\lambda[L^{(2)}(\lambda) = \mathsf{LSB}(\lambda^2 x_0)] > \frac{1}{2} + \frac{\varepsilon\delta}{8}$. We can therefore use the algorithm of Theorem 3 to find a list of candidates $x_1, \ldots, x_n \in \mathbb{F}_p$ containing the desired $x_0$.

To ensure that condition (5) holds, we repeat this process $\lceil 8/\varepsilon\delta \rceil$ times and build a list of candidates of size $O(n/\delta\varepsilon)$. Then condition (5) holds with constant probability during one of these iterations. Therefore, the list of candidates contains the correct $x_0$ with constant probability. By solving for $y$ we obtain a list of candidates for $C'$. That is, we obtain a set $S'$ such that $C' \in S' \subseteq \mathbb{E}$. This list $S'$ can be easily converted to a list of candidates $S$ for $C$ by setting $S = \{c_0 P \mid P \in S'\}$.

Therefore, we just constructed a polynomial time algorithm (in $\log p$ and $\frac{1}{\varepsilon\delta}$) that for any $aG, bG \in \mathbb{E}$ outputs a polynomial size list containing $C$ with constant probability. Using Theorem 4 this algorithm gives an algorithm for computing the Diffie-Hellman function in $\mathbb{E}$ in the required time bound.

To complete the proof of the theorem we also need to consider an algorithm predicting the LSB of the $y$-coordinates. That is, suppose there is an expected

$t$-time algorithm $\mathcal{A}$ such that $\mathrm{Adv}^{Y}_{\phi_\lambda(\mathbb{E}),\phi_\lambda(G)}(\mathcal{A}) > \varepsilon$ for a $\delta$-fraction of $\lambda \in \mathbb{F}_p^*$. We show how to compute the Diffie–Hellman function $\mathsf{DH}_{\mathbb{E},G}$. The proof in this case is very similar to the proof for the $x$-coordinate. The only difference is that since we are using the $Y$ coordinate we obtain an $\mathsf{HNP\text{-}CM}^3$ problem. We use Lemma 1 and Lemma 2 to obtain an $\mathsf{HNP\text{-}CM}^3$ oracle with advantage $\varepsilon\delta/8$ in predicting $\mathsf{LSB}(\lambda^3 y_0)$. The theorem now follows from the algorithm for $\mathsf{HNP\text{-}CM}^3$ given in Theorem 3.

$\square$

## 7 Conclusions

We have showed that no algorithm can predict the $\mathsf{LSB}$ of the $X$ and $Y$ coordinates of the elliptic curve Diffie–Hellman secret for a non-negligible fraction of the curves in $\{\phi_\lambda(\mathbb{E}_0)\}_{\lambda\in\mathbb{F}_p^*}$, assuming the Diffie–Hellman problem is hard on some curve $\mathbb{E}_0 \in \{\phi_\lambda(\mathbb{E}_0)\}_{\lambda\in\mathbb{F}_p^*}$. Our proofs use reductions between many curves by randomly twisting the curve $\mathbb{E}_0$. We hope these techniques will eventually lead to a proof that if CDH is hard on a certain curve $\mathbb{E}$ then the $\mathsf{LSB}$ of Diffie-Hellman is a hard core predicate on that curve.

## References


1. W. Alexi, B. Chor, O. Goldreich, and C. Schnorr. 'RSA and Rabin functions: Certain parts are as hard as the whole', *SIAM J. Computing*, **17**(1988), 194–209, Nov. 1988.
2. I. Blake, G. Seroussi, and N. Smart, *Elliptic Curves in Cryptography*, London Mathematical Society, Lecture Notes Series, **265**, Cambridge University Press, 1999.
3. D. Boneh, 'The decision Diffie–Hellman problem', *In Proc. 3rd Algorithmic Number Theory Symposium*, Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, **1423** (1998), 48–63.
4. D. Boneh, S. Halevi and N. A. Howgrave-Graham, 'The modular inversion hidden number problem', *Preprint*, 2001.
5. D. Boneh and R. Venkatesan, 'Hardness of computing the most significant bits of secret keys in Diffie–Hellman and related schemes', *In Proc. Crypto '96*, Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, **1109** (1996), 129–142. Recent version available at `http://crypto.stanford.edu/~dabo/`.
6. E. El Mahassni, P. Q. Nguyen and I. E. Shparlinski, 'The insecurity of Nyberg–Rueppel and other DSA-like signature schemes with partially known nonces', *Proc. Workshop on Lattices and Cryptography*, Boston, MA, 2001 (to appear).
7. R. Fischlin, C. Schnorr, 'Stronger security proofs for RSA and Rabin bits', *J. Cryptology*, **13** (2000), 221–244.
8. O. Goldreich, L. Levin, 'A hard core predicate for any one way function', *In Proc. 21st ACM Symp. on Theory of Comput.*, 1989, 25–32.
9. M. I. González Vasco and M. Näslund, 'A survey of hard core functions', *In Proc. Workshop on Cryptography and Computational Number Theory*, Singapore 1999, Birkhäuser, 2001, 227–256.

10. M. I. González Vasco and I. E. Shparlinski, 'On the security of Diffie–Hellman bits', *Proc. Workshop on Cryptography and Computational Number Theory*, Singapore 1999, Birkhäuser, 2001, 257–268.
11. M. I. González Vasco and I. E. Shparlinski, 'Security of the most significant bits of the Shamir message passing scheme', *Math. Comp.* (to appear).
12. M. Naslund, 'All bits in $ax+b$ mod $p$ are hard', *In Proc. Crypto '96*, Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, **1109** (1996), 114–128.
13. N. A. Howgrave-Graham and N. P. Smart, 'Lattice attacks on digital signature schemes', *Designs, Codes and Cryptography* (to appear).
14. N. A. Howgrave-Graham, P. Q. Nguyen and I. E. Shparlinski, 'Hidden number problem with hidden multipliers, timed-release crypto and noisy exponentiation', *Preprint*, 2000, 1–26.
15. A. J. Menezes, P. C. van Oorrschot and S. A. Vanstone, *Handbook of applied cryptography*, CRC Press, Boca Raton, FL, 1996.
16. P. Q. Nguyen, 'The dark side of the hidden number problem: Lattice attacks on DSA', *Proc. Workshop on Cryptography and Computational Number Theory*, Singapore 1999, Birkhäuser, 2001, 321–330.
17. P. Q. Nguyen and I. E. Shparlinski, 'The insecurity of the Digital Signature Algorithm with partially known nonces', *Preprint*, 2000, 1–26.
18. P. Q. Nguyen and I. E. Shparlinski, 'The insecurity of the elliptic curve Digital Signature Algorithm with partially known nonces', *Preprint*, 2000, 1–24.
19. P. Q. Nguyen and J. Stern, 'Lattice reduction in cryptology: An update', *In Proc. 4th Algorithmic Number Theory Symposium*, Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, **1838** (2000), 85–112.
20. H. Niederreiter, *Random number generation and quasi–Monte Carlo methods*, SIAM, Philadelphia, 1992.
21. V. Shoup, 'Lower bounds for discrete logarithms and related problems', *In Proc. Eurocrypt '97*, Lect. Notes in Comp. Sci., Springer-Verlag, Berlin, **1233** (1997), 256–266.
22. I. E. Shparlinski, 'Sparse polynomial approximation in finite fields', *Proc. 33rd ACM Symp. on Theory of Comput.*, Crete, Greece, July 6-8, 2001 (to appear).
23. I. E. Shparlinski, 'On the generalized hidden number problem and bit security of XTR', *In Proc. the 14th Symp. on Appl. Algebra, Algebraic Algorithms, and Error-Correcting Codes*, Lect. Notes in Comp. Sci., Springer-Verlag, Berlin (to appear).
24. J. H. Silverman, *The arithmetic of elliptic curves*, Springer-Verlag, Berlin, 1995.
25. D. R. Stinson, *Cryptography: Theory and practice*, CRC Press, Boca Raton, FL, 1995.