Hardness of Computing Individual Bits for One-way Functions on Elliptic Curves

Alexandre Duc[1] and Dimitar Jetchev[2]

LASEC[1], LACAL[2], EPFL, 1015 Lausanne, Switzerland

Abstract. We prove that if one can predict any of the bits of the input to an elliptic curve based one-way function over a finite field, then we can invert the function. In particular, our result implies that if one can predict any of the bits of the input to a classical pairing-based one-way function with non-negligible advantage over a random guess then one can efficiently invert this function and thus, solve the Fixed Argument Pairing Inversion problem (FAPI-1/FAPI-2). The latter has implications on the security of various pairing-based schemes such as the identity-based encryption scheme of Boneh–Franklin, Hess' identity-based signature scheme, as well as Joux's three-party one-round key agreement protocol. Moreover, if one can solve FAPI-1 and FAPI-2 in polynomial time then one can solve the Computational Diffie–Hellman problem (CDH) in polynomial time.
Our result implies that all the bits of the functions defined above are hard-to-compute assuming these functions are one-way. The argument is based on a list-decoding technique via discrete Fourier transforms due to Akavia–Goldwasser–Safra as well as an idea due to Boneh–Shparlinski.

Keywords: One-way function, hard-to-compute bits, bilinear pairings, elliptic curves, fixed argument pairing inversion problem, Fourier transform, list decoding.

1 Introduction

One-way functions (OWF) are functions that are easy to compute but hard to invert. Yet, the definition of a one-way function does not say much about the security of a particular predicate over the input of this function. What if, for instance, the least significant bit is easy to compute? In this case, one might be able to leak partial information if one hides the secret key using this one-way function. Hence, proving that partial information is hard to predict is of primary interest.

In this paper, we study hardness of computing individual bits for one-way functions whose domains are subgroups of points on elliptic curves. Before we ask any question about extracting bits from the input of such functions, we need to specify a binary representation of the points of the subgroup. This amounts to specifying a particular short Weierstrass equation (model) representing the

isomorphism class of the elliptic curve and then specifying a binary representation for the finite field to obtain a representation of the coordinates of the points on this Weierstrass model. Since two distinct short Weierstrass equations could represent the same isomorphism class, it is important to distinguish between elliptic curves (taken up to isomorphism) and short Weierstrass equations.

From the point of view of the hardness of classical computational problems in cryptography (e.g., the elliptic curve discrete logarithm problem or the Diffie–Hellman problem), it makes no difference which short Weierstrass representation one chooses for a given isomorphism class since a solution of the problem on one Weierstrass representation yields (via the isomorphism transformation) a solution on any other Weierstrass representation for the same isomorphism class of elliptic curves. Yet, knowing that one can predict the least significant bit of the input for a particular Weierstrass equation does not necessarily imply that one can predict with the same advantage the least-significant bit on another short Weierstrass equation representing the same curve.

Suppose that we are interested in defining a function whose domain is a particular subgroup of points $\mathbb{G} \subseteq E(\mathbb{F}_p)$ of $\mathbb{F}_p$-points on the curve. By an efficiently computable elliptic curve based function from $\mathbb{G}$ to a set $\mathcal{Y}$ we mean a function $f\colon \mathbb{G} \to \mathcal{Y}$ that can be efficiently computed on any of the short Weierstrass representations of $\mathbb{G}$ and that is independent of the short Weierstrass representation for E. In other words, it is a function on the isomorphism class of E. Our main example for elliptic curve based functions will be functions arising from classical elliptic curve cryptographic pairings: if $\widehat{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a cryptographic pairing and if $Q \in \mathbb{G}$, then $f_Q\colon \mathbb{G} \to \mathbb{G}_T$ defined by $f_Q(P) := e(P, Q)$ is an elliptic curve based function. This function is conjectured to be one-way and is essential in Boneh and Franklin's identity-based encryption scheme (IBE) [5]. The one-wayness of f_Q (a problem known as the Fixed Argument Pairing Inversion 2 (FAPI-2)) is also linked to the security of Joux's three-party one-round key agreement protocol [23] and to the identity-based signature scheme by Hess [19]. This problem and potential approaches to solve it is studied in [11]. More generally, the hardness of f_Q is linked to the hardness of the bilinear Diffie–Hellman problem (BDH) [5].

In our main result (Theorem 1), we show that given an elliptic curve based function, if one can predict (with non-negligible advantage over the point in $\mathbb{G}$ as well as the short Weierstrass equation for E) the kth bit of the input to f then one can efficiently invert f. More precisely, we consider an elliptic curve based function $f\colon \mathbb{G} \to \mathcal{Y}$ on a cyclic subgroup $\mathbb{G} \subseteq E(\mathbb{F}_p)$ of points and we show that if there is an algorithm that takes as input $f(R)$ for some hidden element $R \in \mathbb{G}$ and a short Weierstrass model for E and predicts the kth bit of the x-coordinate of R on that model, then we can invert f (here, the prediction algorithm is imperfect and the only requirement is that it works with non-negligible advantage measured over a random point $R \in \mathbb{G}$ as well as a random short Weierstrass equation representing E). The major consequence of our result is that *all* the bits of the input to the pairing-based one-way function are hard-to-compute assuming that FAPI-2 is hard. More generally, this result

applies to any elliptic curve based one-way function. Our proof uses methods developed by Akavia et al. [2] based on list-decoding via discrete Fourier transforms. We introduce a new code, the *elliptic curve multiplication code*, similar to the multiplication code presented in [2] but whose predicates are evaluated over different short Weierstrass equations. We show that, given access to the kth bit of the x-coordinate of R, we have access to a noisy codeword that can be list-decoded to recover R resulting in an inversion of the one-way function. We believe this code might be of independent interest.

1.1 Previous Work

The first hard-to-compute predicate for a one-way function was found by Blum and Micali [4] for the discrete logarithm (DL) one-way function over a finite field $\mathbb{F}_p$. Subsequently, the question of constructing predicates that are hard-to-compute from one-way functions was studied extensively. For instance, Håstad and Näslung showed that every bit in the RSA [33] one-way function is hard-to-compute [17] using a result of Alexi, Chor, Goldreich and Schnorr [3]. Similarly, for the DL one-way function, Håstad, Schrift and Shamir showed that all the bits are hard-to-compute if the DL is taken modulo a Blum integer [18] following the work of Schrift and Shamir [36]. By changing the way the bits are represented, Schnorr showed that almost all of the bits in the DL function are hard-to-compute [35]. A similar hardness result (independent of the bit representation) was proven in [17]. The last two results also hold for the elliptic curve discrete logarithm (ECDL). For the elliptic curve Diffie–Hellman problem, the hardness of the LSB of the x- and y-coordinates has been studied as well [6,22].

However, all these results apply to a specific one-way function and have to be significantly modified to be used on another OWF (or sometimes cannot be modified at all). Thus, finding generic hard-to-compute predicates that apply to more general collections of one-way functions is highly desirable [14,13].

In 2003, Akavia, Goldwasser and Safra presented a new method to prove that some predicates are hard-to-compute for a one-way function [2]. Their work follows the work by Goldreich and Levin [14]. Using their methodology, security results can be proven for entire classes of functions. Furthermore, it is elegant to use and hides the cumbersome bit manipulations that appeared in the previous proofs. The method relies on the construction of a code that encodes the preimages of the one-way function we try to invert. This means that given a one way function $f\colon \mathcal{X} \to \mathcal{Y}$ and a predicate $P(x)$ for $x \in \mathcal{X}$, we construct a code $\mathcal{C}^P$ that associates to $x \in \mathcal{X}$ a codeword $C_x^P \in \mathcal{C}^P$. If we have access to a corrupted codeword w (which we can get through the bit prediction oracle) and if there is a PPT algorithm that computes a list of all $x \in \mathcal{X}$ such that C_x^P is close to w (in the sense of Hamming distance), then the predicate P is hard-to-compute for f.

The method is used to prove the security of certain bits in RSA, in the Rabin cryptosystem [32], in the DL problem and in the ECDL problem. More precisely, one can prove the security of the $\mathcal{O}(\log\log n)$ least and most significant bits of these functions, where n is the size of the domain of the one-way function.

In 2009, Morillo and Ràfols [27] extended these results and were able to prove the security of *all* bits in RSA, Rabin and DL for prime orders or RSA moduli using a careful analysis of the Fourier coefficients of the function that maps an element of $\mathbb{Z}/n\mathbb{Z}$ to the value of the kth bit of its corresponding representative in $[0, n-1]$. They also extended the result to the Paillier trapdoor permutation [31].

Previous results exist on bit-security for pairing-based protocols [12,34,39] as well as for other bit security results [15,16,28,29]. For instance, a version of the hidden number problem [6,7,8,9,20,26,30,37,38] has been studied by Galbraith et al. in [12] shedding some light on the security of protocols such as Joux's three-party key-agreement protocol [23]. Yet, this is not directly related to FAPI-2.

Overview. The paper is organized as follows: in Section 2, we introduce basic definitions and present our main theorem. In Section 3, we recall basic notions from discrete Fourier transforms on abelian groups used to prove our theorem as well as two important properties from list-decoding, namely Fourier concentration and recoverability. We also mention one of the key ingredients - an algorithm due to Akavia–Goldwasser–Safra that efficiently recovers the heavy Fourier coefficients of a noisy codeword. In Section 4, we prove our main theorem by introducing a new code that we call the elliptic curve multiplication code (ECMC) and by showing that it satisfies the two properties by using techniques due to Boneh–Shparlinski as well as Morillo and Ràfols. We explicitly present the full reduction algorithm by referring to the appendix for various technical details. In Section 5, we show how our result applies to pairing-based one-way functions and also discuss the implications.

2 Main Theorem

2.1 Preliminaries

Let p be a prime and let E be an elliptic curve over $\mathbb{F}_p$. Throughout, we always represent the elements of $\mathbb{F}_p$ via the binary representations of the integers $\{0, 1, \ldots, p-1\}$.

In order to discuss the individual bits of a point on E, we fix a short Weierstrass equation $W\colon y^2 = x^3+ax+b$, $a, b \in \mathbb{F}_p$, $4a^3+27b^2 \neq 0$ representing E. Let $\mathcal{W}(E)$ be the set of all such short Weierstrass equations. Two short Weierstrass equations $y^2 = x^3+ax+b$ and $y^2 = x^3+a'x+b'$ represent the same elliptic curve E over $\mathbb{F}_p$ if and only if there exists an element $\lambda \in \mathbb{F}_p^\times$ such that $a' = \lambda^4 a$ and $b' = \lambda^6 b$. Hence, the set $\mathcal{W}(E)$ is in bijection with $\mathbb{F}_p^\times$. For a point $R \in E(\mathbb{F}_p)$ and $W \in \mathcal{W}(E)$, the x- and y-coordinates of R on the short Weierstrass model W are denoted by $(R_W)_x$ and $(R_W)_y$, respectively. Once a short Weierstrass equation $W\colon y^2 = x^3 + ax + b$ is fixed, we denote the short Weierstrass equation $y^2 = x^3+\lambda^4 ax+\lambda^6 b$ by W_λ. Given any point Q on W, the point $Q_\lambda = (\lambda^2 x, \lambda^3 y)$ is on W_λ for any $\lambda \in \mathbb{F}_p^\times$.

Next, a function $\nu\colon \mathbb{N} \to \mathbb{R}$ is called *negligible* if for every constant $c \in \mathbb{R}_{>0}$, there exists $k_0 \in \mathbb{N}$ such that $|\nu(k)| < k^{-c}$ for all $k > k_0$. A function $\rho\colon \mathbb{N} \to \mathbb{R}$

is *non-negligible* if there exists a constant $c \in \mathbb{R}_{>0}$ and a $k_0 \in \mathbb{N}$ such that $|\rho(k)| > k^{-c}$ for all $k > k_0$.

Remark 1. Note that a function being *non-negligible* is a stronger requirement than a function not being negligible.[1] By the above definitions, every non-negligible function is not negligible. However, there are functions that are not negligible, but are not non-negligible. For example, take any non-negligible function $f\colon \mathbb{N} \to \mathbb{R}$ and define the function

$$g(n) = \begin{cases} 0 & \text{if } n \text{ is even,} \\ f(n) & \text{otherwise.} \end{cases}$$

Obviously, g is neither negligible nor non-negligible.

Let $\mathcal{X}$ be a finite set and let $\mathcal{D}$ be a probability distribution on $\mathcal{X}$. We write $x \in_{\mathcal{D}} \mathcal{X}$ if the element x is chosen according to the distribution $\mathcal{D}$ from $\mathcal{X}$. We now recall some basic notions from cryptography:

Definition 1 (One-way function). *A function $f\colon \mathcal{X} \to \mathcal{Y}$ is a* one-way function *(OWF) if the following conditions hold:*

- *Given $x \in \mathcal{X}$, one can compute $f(x)$ in polynomial time in $\log|\mathcal{X}|$,*
- *For every probabilistic polynomial time (PPT) (in $\log|\mathcal{X}|$) algorithm $\mathcal{A}$, there exists a negligible function ν_A, such that*

$$\Pr[f(z) = y | y = f(x), z = \mathcal{A}(y)] < \nu_{\mathcal{A}}(\log|\mathcal{X}|) \,,$$

  where the probability is taken over $x \in \mathcal{X}$ chosen uniformly at random. In other words, for every PPT (in $\log|\mathcal{X}|$) algorithm $\mathcal{A}$, the advantage to invert f is negligible.

Remark 2. When we are dealing with complexities, we consider sets in their bit representation as a computer would do. For instance, in the above definition, we can see the function f as $f\colon \{0,1\}^n \to \{0,1\}^m$ for $m, n \in \mathbb{N}$. This means that, f is one-way if one can compute $f(x)$ in $\mathrm{poly}(n)$ time and if there is no PPT algorithm that can find a preimage in $\mathrm{poly}(n)$ time.

Definition 2 (Elliptic curve based OWF). *A one-way function whose domain is a subgroup $\mathbb{G} \subseteq E(\mathbb{F}_p)$ is called* elliptic curve based OWF *if its output does not depend on the short Weierstrass equation representing the isomorphism class[2] of E.*

Definition 3 (maj_g). *Given a boolean function $g\colon \mathcal{X} \to \{a_1, a_2\}$, we define*

$$\mathrm{maj}_g \coloneqq \max_{b \in \{a_1, a_2\}} \Pr_{x \in_U \mathcal{X}}[g(x) = b] \,.$$

In other words, maj_g is the probability of the most probable of the two outcomes. This notion is useful when we deal with biased predicates.

[1] Confusing the two notions seems to be a common mistake in the cryptographic literature.

[2] Here, by an isomorphism class of E we really mean the $\mathbb{F}_p$-isomorphism class.

Remark 3. For the rest of the paper, we will be using the majority values of the predicates B_k on $\mathbb{F}_p$ that return the kth least significant bit. If $x \in \mathbb{F}_p$ is viewed as an element of $[0, p-1]$ then we will be using $\delta_p(k) \coloneqq \mathrm{maj}_{B_k}$ for the probability of occurrence of the majority value.

Definition 4 (Efficiently computable predicate). *A boolean predicate* $P\colon \mathcal{X} \to \{0,1\}$ *is* efficiently computable *with respect to a one-way function* $f\colon \mathcal{X} \to \mathcal{Y}$ *if there exists a PPT (in* $\log|\mathcal{X}|$*) algorithm* $\mathcal{A}$ *that can compute* $P(x)$ *from* $f(x)$ *with a non-negligible advantage over the majority value, i.e., such that*

$$\Pr_{x \in_U \mathcal{X}}[\mathcal{A}(f(x)) = P(x)] \geq \mathrm{maj}_P + \frac{1}{\mathrm{poly}(\log|\mathcal{X}|)} ,$$

for some polynomial that is independent of $|\mathcal{X}|$.

Definition 5 (Hard-to-compute predicate). *A boolean predicate* P *is* hard-to-compute *with respect to a one-way function* f *if it is not efficiently computable.*

Remark 4. Note that often, the term *hard-core* predicate is misused for *hard-to-compute* predicate. In this paper, we never use the term hard-core predicate (which, to the best of our knowledge, means that every algorithm that predicts P has negligible advantage over a random guessing; the latter is a strong definition and is not suitable for computational purposes). The difference between hard-core and hard-to-compute comes from the difference between a non-negligible function and a function that is not negligible as showed before.

2.2 Main Result on Hard-to-compute Bits

Throughout, we consider bits that take values in $\{\pm 1\}$ instead of $\{0,1\}$ where we substitute -1 for 1 and 1 for 0, i.e., the new value is $(-1)^b$ with $b \in \{0,1\}$. For a prime field $\mathbb{F}_p$, let $B_k\colon \mathbb{F}_p \to \{\pm 1\}$ be the predicate returning the value of the kth bit of $x \in \mathbb{F}_p$ viewed as an integer in $\{0, 1, \ldots, p-1\}$. Suppose that E is an elliptic curve over $\mathbb{F}_p$ and $\mathbb{G} \subseteq E(\mathbb{F}_p)$ is a subgroup of prime order n of cryptographically meaningful size (i.e., $n = \Theta(p)$).

Let $R \in \mathbb{G}$ be a hidden point, let $\mathcal{Y}$ be any set and let $f\colon \mathbb{G} \to \mathcal{Y}$ be a one-way function. Suppose that we have an imperfect oracle $\mathcal{U}_k$ that takes as input a short Weierstrass equation $W \in \mathcal{W}(E)$ and the value $f(R) \in \mathcal{Y}$ and predicts $B_k((R_W)_x)$ with non-negligible advantage over the majority value $\delta_p(k)$ (here, the advantage is taken over a random short Weierstrass equations $W \in \mathcal{W}(E)$ and over a random point $R \in \mathbb{G}$). Then, we show that we can efficiently invert f.

Before we state precisely the theorem, we rigorously define the advantage of the bit-prediction oracle $\mathcal{U}_k$:

Definition 6 (Advantage). *We say that* $\mathcal{U}_k$ *has advantage* ϵ *in predicting the predicate* B_k *of the* x*-coordinate of the input* $R \in \mathbb{G}$ *of the one way function* f *if*

$$\mathrm{Adv}_f^{x,k}(\mathcal{U}_k) \coloneqq \left| \Pr_{\substack{W \in_U \mathcal{W}(E) \\ R \in_U \mathbb{G}, z}} [\mathcal{U}_k(W, f(R); z) = B_k((R_W)_x)] - \delta_p(k) \right| > \epsilon ,$$

where z is a random variable corresponding to the random coins used by the oracle $\mathcal{U}_k$. Similarly, we define the advantage $\mathrm{Adv}_f^{y,k}(\mathcal{U}_k)$ of predicting the kth bit of the y-coordinate of the input point R to f.

We are now ready to state the main theorem:

Theorem 1. *Let $k \geq 0$ be an integer and let $\epsilon \in (0,1)$. Let E and $\mathbb{G}$ be as above (i.e., $\mathbb{G}$ is a subgroup of prime order $n = \Theta(p)$). Let $\mathcal{Y}$ be any set and let $f : \mathbb{G} \to \mathcal{Y}$ be an elliptic curve based one-way function on E. Let $\mathcal{U}_k(W, v; z)$ be an algorithm that takes as input $W \in \mathcal{W}(E)$, $v \in \mathcal{Y}$ and outputs an element of $\{\pm 1\}$ in time T. Assume that $\mathrm{Adv}_f^{x,k}(\mathcal{U}_k) > \epsilon$. Then, there exists an algorithm $\mathcal{A}$ that inverts $f \colon \mathbb{G} \to \mathcal{Y}$ in time $T \cdot \mathrm{poly}(\log p, \frac{1}{\epsilon})$ for some polynomial that is independent of p, E, $\mathbb{G}$, and ϵ.*

Remark 5. One can see from the above theorem that if $1/\epsilon = \mathrm{poly}(\log p)$ and if $T = \mathrm{poly}(\log p)$ then one can invert the function f efficiently (in time polynomial in $\log p$). This means that either the kth bit of the input to f is hard-to-compute or that the function is invertible.

Remark 6. Note that our result is on average as our argument exploits in an essential way the freedom to change the short Weierstrass equation. This is why we assume that algorithm $\mathcal{U}_k$ works on a non-negligible fraction of all the short Weierstrass equations W. Ideally, one wishes to fix a short Weierstrass equation W and prove similar hardness result only on W. This last question appears to be very difficult and out of reach with the current techniques that one has so far for showing hardness of bits (see also [6] for a similar remark regarding hardness of computing the least significant bit for Diffie–Hellman secrets).

3 Hard–to–compute Predicates via List-decoding

3.1 Fourier Transforms

In order to describe the general method of Akavia–Goldwasser–Safra, we briefly recall some basic notions related to Fourier transforms.

Let G be a finite abelian group. If $f, g \colon G \to \mathbb{C}$ are functions then their *inner product* is defined as $\langle f, g \rangle \coloneqq 1/|G| \sum_{x \in G} \overline{f(x)} g(x)$. The ℓ_2-norm on the space $\mathbb{C}(G)$ of all complex valued functions $f \colon G \to \mathbb{C}$ is then $\|f\|_2 \coloneqq \sqrt{\langle f, f \rangle}$. A *character* of G is a homomorphism $\chi \colon G \to \mathbb{C}^\times$, i.e., $\chi(x+y) = \chi(x)\chi(y)$ for all $x, y \in G$. The set of all characters of G forms a group $\widehat{G}$, the character group. The elements of $\widehat{G}$ form an orthonormal basis for the space $\mathbb{C}(G)$ (the *Fourier basis*). One can then describe a function $f \in \mathbb{C}(G)$ via its *Fourier expansion* $\sum_{\chi \in \widehat{G}} \langle f, \chi \rangle \chi$. Equivalently, one can define the Fourier transform $\widehat{f} \colon \widehat{G} \to \mathbb{C}$ of f by $\widehat{f}(\chi) = \langle f, \chi \rangle$. The coefficients $\widehat{f}(\chi)$ in the Fourier basis $\{\chi\}_{\chi \in \widehat{G}}$ are the *Fourier coefficients* of f. When $G = \mathbb{Z}/n\mathbb{Z}$, the characters of G are defined by $\chi_\alpha(x) \coloneqq \omega_n^{\alpha x}$, for $\alpha \in \mathbb{Z}/n\mathbb{Z}$ and $\omega_n \coloneqq \exp(2\pi i/n)$. The *weight* of a Fourier coefficient $\widehat{f}(\chi)$ is $|\widehat{f}(\chi)|^2$. Using these definition, we can define *heavy characters* with respect to a function f:

Definition 7 (Heavy characters). *Given a function $f\colon G \to \mathbb{C}$ and a threshold τ, we denote by* $\mathrm{Heavy}_\tau(f)$ *the set of characters for which the weight of the corresponding Fourier coefficient of f is at least τ. In other words,*

$$\mathrm{Heavy}_\tau(f) \coloneqq \{\chi \in \widehat{G} : |\widehat{f}(\chi)|^2 \geq \tau\} \ .$$

We will frequently approximate a function $f \in \mathbb{C}(G)$ using subsets $\Gamma \subset \widehat{G}$ of characters via its *restriction*: $f_{|\Gamma} \coloneqq \sum_{\chi\in\Gamma} \widehat{f}(\chi)\chi$.

3.2 Codes, Fourier Concentration and Recoverability

When working on an abelian group G, we consider binary codewords of length $|G|$. Every codeword corresponding to an element $x \in G$ will be represented by a function $C_x\colon G \to \{\pm 1\}$. If $G = \mathbb{Z}/n\mathbb{Z}$ then C_x is represented by $(C_x(0), C_x(1), \ldots, C_x(n-1))$. We define now two properties of codes we will use in our proof.

Definition 8 (Concentration). *Let $\epsilon > 0$ be a real number. A function $f\colon G \to \{\pm 1\}$ is called Fourier ϵ-concentrated if there exists a set of characters $\Gamma \subseteq \widehat{G}$ of size* $\mathrm{poly}(\log|G|, 1/\epsilon)$ *(for a polynomial that does not depend on $|G|$, ϵ or the function f) such that $\|f - f_{|\Gamma}\|_2 \leq \epsilon$.*

A code $\mathcal{C} = \{C_x\colon G \to \{\pm 1\}\}$ is ϵ-concentrated *if each of its codewords C_x is* Fourier ϵ-concentrated*. In other words, we can approximate with an error at most ϵ every codeword using a polynomial number (in $\log|G|$ and $1/\epsilon$) of characters $\chi \in \widehat{G}$. A function is called Fourier concentrated if it is ϵ-concentrated for every $\epsilon > 0$. A code is called* Fourier concentrated *if all of its codewords are Fourier concentrated.*

Definition 9 (Recoverable code). *A code $\mathcal{C} = \{C_x\colon G \to \{\pm 1\}\}$ is* recoverable *if there exists an algorithm that takes as input a character $\chi \in \widehat{G}$ and a threshold τ and outputs (in time polynomial in $\log|G|$ and $1/\tau$) the list $\{x \in G\colon \chi \in \mathrm{Heavy}_\tau(C_x)\}$ of all codewords having χ as a τ-heavy coefficient.*

Using the orthogonality of the characters $\chi \in \widehat{G}$, one shows [2, Lem.1] that if a code $\mathcal{C}$ is concentrated, then a word $w_x\colon G \to \mathbb{C}$ and a close codeword C_x have at least one heavy Fourier coefficient in common. We show here a slight modification of this lemma.

Lemma 1 ([2, Lem.1]). *Let $f\colon \mathbb{Z}/n\mathbb{Z} \to \{\pm 1\}$ be a Fourier concentrated function and let $g\colon \mathbb{Z}/n\mathbb{Z} \to \{\pm 1\}$ such that*

$$\Pr_{x\in\mathbb{Z}/n\mathbb{Z}}[f(x) = g(x)] \geq \mathrm{maj}_f + \epsilon \ , \tag{1}$$

for some $\epsilon > 0$. Then there exists a threshold τ such that $1/\tau$ is polynomial in $1/\epsilon$ and $\log n$, and $\exists\, \chi \neq 0, \chi \in \mathrm{Heavy}_\tau(f) \cap \mathrm{Heavy}_\tau(g)$.

We omit the proof since it is straightforward from the proof in [2].

In Section 4, we will apply this lemma in the following way: every preimage $x \in G$ of the one-way function we try to invert corresponds to a codeword C_x.

First, we recover a noisy version w_x of C_x by using the prediction oracle. If the code is concentrated, the words w_x and C_x share at least one heavy coefficient. Thus, if we can compute this heavy coefficient in polynomial time and if the code is recoverable, then we can recover x in polynomial time.

One recovers the heavy coefficient using the following theorem:

Theorem 2 ([2, Thm.6]). *There exists a randomized learning algorithm over $\mathbb{Z}/n\mathbb{Z}$ that, given a function $w\colon \mathbb{Z}/n\mathbb{Z} \to \{\pm 1\}$, $0 < \tau$ and $0 < \delta < 1$, returns a list of $\mathcal{O}(1/\tau)$ characters containing $\mathrm{Heavy}_\tau(w)$ with probability at least $1-\delta$ (here, the probability is taken over the random coins of the algorithm) and that has running time* [3]

$$\widetilde{\mathcal{O}}\left(\log(n)\ln^2\frac{(1/\delta)}{\tau^{5.5}}\right).$$

For completeness, we recall the algorithm in the full version of the paper [10].

Remark 7. In the language of Akavia et al. [2, §2.3], $\mathbb{Z}/n\mathbb{Z}$ is a *learnable domain*. It turns out that any finite abelian group G is a learnable domain [1].

4 Proof of Theorem 1

We will reduce the proof of Theorem 1 to a list-decoding problem that will be solved using Akavia et al.'s method. The first step is to properly define a code that reflects our input recovery problem. We explain in Section 4.1that the straightforward definition of such a code does not quite work since the Fourier transforms of the codewords are difficult to analyze from the point of view of concentration and recoverability. In order to overcome this difficulty, we use an idea motivated by the work of Boneh and Shparlinski on the Hidden Number Problem that modifies the prediction oracle via extra randomization while still keeping the non-negligible advantage. This leads us to the definition of the Elliptic Curve Multiplication Code (ECMC) (Definition 10).

4.1 The Elliptic Curve Multiplication Code (ECMC)

Let $B_k\colon \mathbb{F}_p \to \{\pm 1\}$ be the binary predicate that returns 1 if the kth least significant bit of the argument is 0 and -1 otherwise. A natural way to associate a code to this predicate is to fix a (base) short Weierstrass equation $W \in \mathcal{W}(E)$ and a hidden point R and define the codewords

$$C_R^{B_k,W}\colon \mathbb{F}_p \to \{\pm 1\}\,, \qquad C_R^{B_k,W}(\lambda) = B_k(\lambda^2\cdot(R_W)_x) = B_k((R_{W_\lambda})_x)\,.$$

The above definition is natural since the isomorphism class $\mathcal{W}(E)$ of short Weierstrass equations consists precisely of the equations W_λ where $\lambda \in \mathbb{F}_p^\times$. So, each codeword encodes the kth bit of all representations of the point $R \in \mathbb{G}$ on the equations from $\mathcal{W}(E)$. In order to study how concentrated these codes are, one

[3] A function is $\widetilde{\mathcal{O}}(f(n))$ if it is $\mathcal{O}\left(f(n)\cdot\log(f(n))^k\right)$ for some $k \in \mathbb{N}$.

needs precise estimates of the Fourier coefficients of these functions. Yet, the only tool we are aware of that gives such estimates are standard estimates from analytic number theory on Gauss sums (see full version of the paper).

Unfortunately, these are not sufficient to get any information about how concentrated the code is. If one is able to replace the square term λ^2 with a linear term in λ, one could obtain a much better control on the code.As mentioned above, we use an idea of Boneh and Shparlinski [6, §5] that modifies the prediction oracle via further randomization while keeping the advantage non-negligible.

The idea works as follows: suppose that $\mathcal{U}_k$ is the prediction oracle from the statement of Theorem 1. Recall that given a hidden point $R \in \mathbb{G}$, the oracle returns an element of $\{\pm 1\}$ in such a way that $\mathrm{Adv}_f^{x,k}(\mathcal{U}_k) > \epsilon$.

If $\mathbb{F}_p^2 \subset \mathbb{F}_p$ is the set of squares in $\mathbb{F}_p$, let $r\colon \mathbb{F}_p^2 \to \mathbb{F}_p$ be a function satisfying $r(\lambda)^2 = \lambda$ that is chosen uniformly at random among all such functions. The observation of Boneh and Shparlinski is that one can define an auxiliary prediction oracle $\mathcal{U}_k'$ using $\mathcal{U}_k$ as follows:

$$\mathcal{U}_k'(W_\lambda, f(R); z) = \begin{cases} \mathcal{U}_k(W_{r(\lambda)}, f(R); z) & \text{if } \lambda \in \mathbb{F}_p^\times \text{ is a square in } \mathbb{F}_p^\times \\ \underset{b\in\{\pm 1\}}{\arg\max} \Pr_{x\in_U \mathbb{F}_p}[B_k(x) = b] & \text{otherwise} \end{cases}.$$

Hence, if λ is not a square, the oracle returns the most common value which is the best random strategy to guess B_k. We now associate a code to the modified oracle $\mathcal{U}_k'$ rather than to the original oracle $\mathcal{U}_k$ and thus, arrive at the following definition (we include the more general case of binary predicates that are not necessarily the predicates B_k):

Definition 10 (Elliptic curve multiplication code (ECMC)). *Let E be an elliptic curve over $\mathbb{F}_p$ and let $P\colon \mathbb{F}_p \to \{\pm 1\}$ be a binary predicate. Let $\mathbb{G} \subset E(\mathbb{F}_p)$ be a cyclic subgroup. Given a (base) short Weierstrass equation $W\colon y^2 = x^3 + ax + b$ representing E, the* elliptic curve multiplication code *is the code $\mathcal{C}^{P,W} = \{C_R^{P,W}\colon \mathbb{F}_p \to \{\pm 1\}\}_{R\in\mathbb{G}}$ defined by*

$$C_R^{P,W}(\lambda) = P(\lambda \cdot (R_W)_x)\ ,$$

where R_W denotes the tuple (x, y) representing the point R on W.

Remark 8. Reducing the quadratic term λ^2 with λ is a big advantage since (as we will show in Section 4.2), the Fourier transform of $B_k(\lambda)$ is simpler to analyze for the purpose of studying heavy coefficients than the Fourier transform of $B_k(\lambda^2)$. This makes it easier to show that the code $\mathcal{C}^{B_k,W}$ is Fourier concentrated and recoverable and, thus, apply the techniques of Akavia et al. to obtain a list-decoding algorithm.

Lemma 2. *Let $W \in \mathcal{W}(E)$ be a fixed (base) short Weierstrass equation and let $\mathcal{U}_k$ be the prediction algorithm from the statement of Theorem 1. There exists a set S of points $R \in \mathbb{G}$ satisfying*

$$|S| \geq \frac{\epsilon}{4\left(1 - \delta_p(k) - \frac{\epsilon}{4}\right)}|\mathbb{G}| \tag{2}$$

such that for every $R \in S$, given $f_Q(R)$, we have access to a corrupted codeword $w_{R,W}$ such that

$$\Pr_{\lambda \in_U \mathbb{F}_p}[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] \geq \delta_p(k) + \frac{\epsilon}{4}\,, \qquad \forall R \in S\,. \tag{3}$$

Proof. Recall that our prediction algorithm $\mathcal{U}_k$ satisfies:

$$\Pr_{W_i,R;z}[\mathcal{U}_k(W_i, f(R); z) = B_k((R_{W_i})_x)] > \delta_p(k) + \epsilon\,. \tag{4}$$

The latter is equivalent to

$$\Pr_{\lambda,R;z}\left[\mathcal{U}_k(W_\lambda, f(R); z) = B_k(\lambda^2 \cdot (R_W)_x)\right] > \delta_p(k) + \epsilon\,. \tag{5}$$

Given a hidden point $R \in \mathbb{G}$, define $w_{R,W}$ as follows:

$$w_{R,W}(\lambda) = \begin{cases} \mathcal{U}_k(W_{r(\lambda)}, f(R); z) & \text{if } \lambda \text{ is a square} \\ \arg\max\limits_{b \in \{\pm 1\}} \Pr\limits_{x \in_U \mathbb{F}_p}[B_k(x) = b] & \text{otherwise,} \end{cases}$$

where $r\colon \mathbb{F}_p^2 \to \mathbb{F}_p$ is chosen uniformly at random among all function $r\colon \mathbb{F}_p^2 \to \mathbb{F}_p$ satisfying $r(\lambda)^2 = \lambda$ and where z is the random coin used by $\mathcal{U}_k$. Using the randomness of r, we estimate

$$\begin{aligned}
&\Pr_{\lambda,R;z}\left[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)\right] = \\
&\quad = \frac{1}{2}\Pr_{\substack{\lambda \in_U \mathbb{F}_p^2, \\ R;z}}\left[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)\right] + \frac{1}{2}\Pr_{\substack{\lambda \notin \mathbb{F}_p^2, \\ R;z}}\left[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)\right] \\
&\quad = \frac{1}{2}\Pr_{\substack{\lambda' \in_U \mathbb{F}_p, \\ R;z}}\left[\mathcal{U}_k(W_{\lambda'}, f(R); z) = B_k(\lambda'^2 \cdot (R_W))\right] + \frac{1}{2}\delta_p(k) \\
&\quad > \frac{1}{2}\left(\delta_p(k) + \epsilon\right) + \frac{1}{2}\delta_p(k) = \delta_p(k) + \frac{\epsilon}{2}\,.
\end{aligned} \tag{6}$$

Next, let $S \subseteq \mathbb{G}$ be the subset of all points $R \in \mathbb{G}$ that satisfy

$$\Pr_{\lambda;z}[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)] > \delta_p(k) + \frac{\epsilon}{4}\,.$$

Points in this set satisfy (3). We now show that the set S satisfies (2). Using (6), we arrive at

$$\begin{aligned}
\delta_p(k) + \frac{\epsilon}{2} &< \frac{1}{|\mathbb{G}|}\sum_{R \in \mathbb{G}}\Pr_{\lambda;z}\left[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)\right] \\
&= \frac{1}{|\mathbb{G}|}\left(\sum_{R \in S}\Pr_{\lambda;z}\left[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)\right] + \sum_{R \in \mathbb{G}\setminus S}\Pr_{\lambda;z}\left[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)\right]\right) \\
&< \frac{1}{|\mathbb{G}|}\left(|S| + |\mathbb{G}\setminus S|\left(\delta_p(k) + \frac{\epsilon}{4}\right)\right) = \frac{|S|}{|\mathbb{G}|}\left(1 - \delta_p(k) - \frac{\epsilon}{4}\right) + \left(\delta_p(k) + \frac{\epsilon}{4}\right)\,.
\end{aligned}$$

Since $\delta_p(k) \neq 1$, we obtain (2). □

Remark 9. If $1/\epsilon = \mathrm{poly}(\log p)$ then the above lemma tells us that the kth bit is predictable with non-negligible advantage over a random guess for a polynomial fraction of all the points $R \in \mathbb{G}$.

In the next section, we explain in more detail the two major properties of the ECMC associated to the kth bit predicates, namely, Fourier concentration and recoverability. This is done via the methods developed in [27].

4.2 Fourier Concentration of ECMC

In order to gain more control on the size of the Fourier coefficients $\widehat{B_k}(\alpha)$, and thus, be able to pick the heavy ones, we use another idea of Morillo and Ràfols: since p is odd, we can assume that $\alpha \in \left[-\frac{p-1}{2}, \frac{p-1}{2}\right]$. Consider the following two cases for α:

- When $\alpha \geq 0$, we consider $\delta_{\alpha,k} \coloneqq 2^k\alpha - (p-1)/2 \bmod p$ and let $\lambda_{\alpha,k} \in [0, 2^{k-1}-1]$ be the unique integer for which $2^k\alpha = (p-1)/2 + \delta_{\alpha,k} + p\lambda_{\alpha,k}$.
- When $\alpha < 0$, we consider $\delta_{\alpha,k} = 2^k\alpha + (p+1)/2 \bmod p$ and let $\lambda_{\alpha,k} \in [0, 2^{k-1}-1]$ be the unique integer for which $2^k\alpha = -(p+1)/2 + \delta_{\alpha,k} + p\lambda_{\alpha,k}$.

For both of the cases, there are unique integers $\mu_{\alpha,k} \in [0, r]$ and $r_{\alpha,k} \in [0, 2^k - 1]$ such that $a_p\left(\alpha 2^k - (p-1)/2\right) = \mu_{\alpha,k}2^k + r_{\alpha,k}$, where $a_p(x) = \min(x \bmod p, p - x \bmod p)$ for $y \bmod p$ being taken in $[0, p-1]$. From here, one characterizes (see full version of the paper)the asymptotic behavior of $|\widehat{B_k}(\alpha)|$ by $|\widehat{B_k}(\alpha)|^2 < \mathcal{O}\left(1/(\lambda_{\alpha,k}^2\mu_{\alpha,k}^2)\right)$.

The idea of having the above representation $(\lambda_{\alpha,k}, \mu_{\alpha,k})$ is that it is very convenient for picking the heavy Fourier coefficients: one simply has to pick the coefficients α for which $(\lambda_{\alpha,k}, \mu_{\alpha,k})$ is in a box $[0, 1/\tau] \times [0, 1/\tau]$ for $\tau = \mathrm{poly}(\log p)$.

4.3 Recoverability of ECMC and End of Proof

Fix a short Weierstrass equation $W \in \mathcal{W}(E)$. According to Lemma 2, there exists a subset $S \subset \mathbb{G}$ of size determined by (2) and the property that for any $R' \in S$, we have access to a corrupted codeword $w_{R',W}$ satisfying (3). The problem is that our hidden point $R \in \mathbb{G}$ need not be in S. In order to remedy this, we repeat the following procedure: we pick a random multiple $s \in [1, n-1]$ and set $R' = sR$. Note that when s is invertible modulo n, knowing R' is equivalent to knowing R. Thus, if s is chosen uniformly at random, we have $1/\mathrm{poly}(\log p)$-chance of obtaining R' in the set S.

Suppose for the moment that R' happens to be in S. One can then use Lemma 1 to deduce that there exists $0 < \tau < 1$ for which $1/\tau$ is polynomial in $\log p$ and $1/\epsilon$ such that the noisy codeword $w_{W,R'}$ and the actual codeword $C_{R'}^{B_k,W}$ share a τ-heavy Fourier coefficient. Then, we apply the learning algorithm of Akavia et al. (Theorem 2) to efficiently compute all τ-heavy Fourier

characters χ_β for the noisy codeword $w_{R',W}$. We then run the recovery algorithm (Algorithm 1) for each of these τ-heavy Fourier coefficients to decode the hidden R' and thus, obtain the possible R's by computing $s^{-1}R'$. Assuming that $w_{R',W}$ is Fourier concentrated, we only have to run this algorithm $\mathrm{poly}(\log p)$ times, so we get a polynomial time (in $\log p$ and $1/\epsilon$) recovery procedure for R'.

Notice that we have no way of knowing whether $R' \in S$ unless we try to recover the point via the above recovery procedure. Yet, by using a random choice of $s \in [1, n-1]$ and repeating the procedure $\mathrm{poly}(\log p)$ times, weobtain (with high probability) a point R' in the set S guaranteed by Lemma 2 and thus, prove Theorem 1.

The method used in our proof is close to the list-decoding method of Akavia et al. [2, Lem. 5] and was successfully used by Morillo and Råfols [27, §6]. The reason it works is that the codeword $\mathcal{C}_R^{B_k,W}$ is τ-concentrated in $\Gamma_{R,W} = \{\chi_\beta \colon \beta \equiv \alpha \cdot (R_W)_x \bmod p, \chi_\alpha \in \Gamma\}$ where Γ is the set of additive characters $\chi_\alpha \colon \mathbb{F}_p \to \mathbb{C}^\times$ where $(\lambda_{\alpha,k}, \mu_{\alpha,k})$ is in a small square of size $\mathcal{O}(1/\tau)$ and lower-right corner at $(0,0)$, i.e., $\Gamma = \{\chi_\alpha \colon \lambda_{\alpha,k} = \mathcal{O}(1/\tau), \mu_{\alpha,k} = \mathcal{O}(1/\tau)\}$. Here, we will take τ such that $1/\tau = \mathrm{poly}(\log p)$.

Algorithm 1 The recovery algorithm

Input: An additive character χ_β of $\mathbb{F}_p$, a threshold parameter τ with $1/\tau \in \mathrm{poly}(\log p)$ and $z \in \mathcal{Y}$ such that $z = f(R)$ for a hidden point R.
Output: The hidden point $R \in \mathbb{G}$ such that $f(R) = z$.

```
Calculate Γ ← {α ∈ F_p : λ_{α,k} = O(1/τ), μ_{α,k} = O(1/τ)}.
for α ∈ Γ\{0} do
    Compute x ← βα^{-1} mod p
    if y ∈ F_p exists so that R = (x, y) ∈ W(F_p) and f(R) = z then
        return R
    end if
end for
return false.
```

The above algorithm works in time polynomial in $\log p$ because 1) the algorithm from Theorem 2 works in polynomial time (in $\log p$); 2) the set S from Lemma 2 is a polynomial fraction of all points in $\mathbb{G}$ and hence, a randomly chosen multiple will be recoverable with probability $1/\mathrm{poly}(\log p)$ (so, we need on average $\mathrm{poly}(\log p)$ trials to exit the **repeat** loop). This completes the proof of Theorem 1.

5 Application To Pairing-based One-way Functions

5.1 Pairing-based One-way Functions

We define now a pairing-based one-way function. For an prime n, let $E[n]$ be the subgroup of points of E of order n (the points in $E[n]$ are defined over the

Algorithm 2 Elliptic curve-based OWF inversion algorithm

Input: An elliptic curve $E/\mathbb{F}_p$, a subgroup $\mathbb{G} \subset E$ of prime order $n = \Theta(p)$, an element $z = f(R) \in \mathcal{Y}$ for a hidden point $R \in \mathbb{G}$ and access to a (noisy) prediction oracle $\mathcal{U}_k$ for $B_k((R_W)_x)$ for $W \in \mathcal{W}(E)$.
Output: An input point $R \in \mathbb{G}$ such that $f(R) = z$.

Fix a (base) short Weierstrass equation $W \in \mathcal{W}(E)$.
Choose τ such that $1/\tau = \mathrm{poly}(\log p)$
Choose a random function $r\colon \mathbb{F}_p^2 \to \mathbb{F}_p$.
repeat
  Choose a random $s \in [1, n-1]$ and set $R' \leftarrow sR$
  Apply the algorithm of Theorem 2 to compute $\mathrm{Heavy}_\tau(w_{R',W})$ for the function (noisy codeword) $w_{R',W}$ from Lemma 2 defined via r and $\mathcal{U}_k$.
  for $\chi_\beta \in \mathrm{Heavy}_\tau(w_{R',W})$ **do**
    Run Algorithm 1 with $z' \leftarrow z^s$ to try to recover R'
    if Algorithm 1 does not fail **then**
      break
    end if
  end for
until R' is recovered
return $R \leftarrow s^{-1}R'$

algebraic closure $\overline{\mathbb{F}}_p$ of $\mathbb{F}_p$). Let k be the smallest integer for which $n \mid p^k - 1$ (also known as the *embedding degree*) and let μ_n be the subgroup of order n of $\mathbb{F}_{p^k}^\times$. Let $e\colon E[n] \times E[n] \to \mu_n$ be a bilinear pairing, e.g., the Tate or the Weil pairing. Let $\mathbb{G} \coloneqq \langle S \rangle$ for an $S \in E(\mathbb{F}_p)$. To avoid having $e(R, Q) = 1$ for all $R, Q \in \mathbb{G}$, we need to suitably twist e and define what we refer to as a *cryptographic pairing*:

Definition 11 (Cryptographic pairing). *Let $\xi\colon E \to E$ be a non-trivial endomorphism defined over an extension field of $\mathbb{F}_p$ (ξ is often referred to as a distortion map). We define the* cryptographic pairing $\widehat{e}\colon \mathbb{G} \times \mathbb{G} \to \mu_n$ *as*

$$\widehat{e}(R, Q) = e(R, \xi(Q)) \,, \qquad R, Q \in \mathbb{G} \,.$$

Here, if $\mathbb{G}$ is a cyclic subgroup and if $R, Q \in \mathbb{G}$ then $e(R, Q)$ will be trivial since e is bilinear and alternating. The role of the endomorphism ξ is to *distort* Q in such a way that $e(R, \xi(Q)) \neq 1$.

A typical example of a cryptographic pairing (see [5]) is a twisted version of the Weil pairing. More precisely, let $p \equiv 2 \bmod 3$ and $q > 3$ be two primes such that q divides $p-1$ and let E be the elliptic curve over $\mathbb{F}_p$ defined by $y^2 = x^3 + 1$. Let $\mathbb{G}$ be the cyclic group generated by a random $S \in E(\mathbb{F}_p)$ of order q. The distortion map is defined as $\xi(Q_x, Q_y) = (\zeta Q_x, Q_y)$, for $\zeta \in \mathbb{F}_{p^2}$, $\zeta \notin \mathbb{F}_p$ such that $\zeta^2 + \zeta + 1 = 0$ (such a ζ exists as long as $X^2 + X + 1$ has a zero in $\mathbb{F}_p[X]$ which is equivalent to $p \equiv 2 \bmod 3$). One could think of ζ as distorting one of the points so that it is mapped to a point that is outside of the group $\mathbb{G}$ and that is defined over a non-trivial extension of $\mathbb{F}_p$.

Definition 12 (Pairing-based one-way function). *Let E be an elliptic curve over $\mathbb{F}_p$ with Weierstrass equation $y^2 = x^3 + ax^2 + b$ and let $\mathbb{G} \subseteq E[n]$ be a*

cyclic subgroup. Let $Q \in \mathbb{G}$ be a fixed generator and let $\widehat{e}\colon \mathbb{G} \times \mathbb{G} \to \mu_n$ be a cryptographic bilinear pairing. We define a function $f_Q\colon \mathbb{G} \to \mu_n$ by $f_Q(R) \coloneqq \widehat{e}(R, Q)$, for $Q \in \mathbb{G}$. The preimage R will often be referred to as a hidden point*.*

The function $f_Q(R)$ is believed to be one-way [11,23,5].

Obviously, we can apply Theorem 1 to f_Q:

Corollary 1. *Let $k \geq 0$ be an integer and let $\epsilon \in (0, 1)$. Let E, $\mathbb{G}$ and Q be as above (i.e., $\mathbb{G}$ is cyclic of order $n = \Theta(p)$ and $Q \in \mathbb{G}$ is a generator). Let $\mathcal{U}_k = \mathcal{U}_k(W, v; z)$ be an algorithm that takes as input $W \in \mathcal{W}(E)$, $v \in \mu_n$ and outputs an element of $\{\pm 1\}$ in time T. Assume that $\mathrm{Adv}_{f_Q}^{x,k}(\mathcal{U}_k) > \epsilon$. Then there exists an algorithm $\mathcal{A}$ that inverts $f_Q\colon \mathbb{G} \to \mu_n$ in time $T \cdot \mathrm{poly}(\log p, \frac{1}{\epsilon})$ for some polynomial that is independent of p, E, $\mathbb{G}$, ϵ and Q.*

5.2 Consequences of our Result

Corollary 1 implies that either every bit of the input of f_Q is hard-to-compute or that f_Q can be inverted efficiently, i.e., FAPI-2 is easy. The hardness of FAPI-2 has been related to various problems [11,24,23].

Definition 13 (BDH). *Let $\widehat{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear pairing, let S be a generator of $\mathbb{G}$ and let n be the order of $\mathbb{G}$. The* Bilinear Diffie–Hellman problem *(BDH) is the following problem: given $\langle S, aS, bS, cS\rangle$, $a, b, c \in \mathbb{Z}/n\mathbb{Z}$, compute $\widehat{e}(S, S)^{abc}$.*

The following relations hold. The hardness of BDH implies the hardness of the computational Diffie-Hellman problem (CDH) in both $\mathbb{G}$ and $\mathbb{G}_T$, which imply the hardness of FAPI-2. Recall that CDH in $\mathbb{G}$ consists in computing abS given $\langle aS, bS, S\rangle$. The hardness of FAPI-2 implies also the hardness of the discrete logarithm in $\mathbb{G}_T$. Hence, our result implies that if we assume that CDH is hard in both groups, every bit of the input of f_Q is hard-to-compute. Many cryptographic schemes relies on the hardness of BDH or FAPI-2. We show what implications an easy FAPI-2 would have.

Boneh–Franklin's Identity-based Encryption Scheme. The security of this well-known scheme [5] relies on the hardness of BDH. If FAPI-2 is easy, then an adversary can recover the secret key of any user of the system. Recall that in IBE, the secret key is computed as $d_{ID} \coloneqq sQ_{ID}$, where Q_{ID} is a point dependent on the identity of the owner of the key and s is the master key. Two points are also public parameters of the scheme: P, which is a generator of $\mathbb{G}$ and $P_{pub} \coloneqq sP$. Hence, $\widehat{e}(P_{pub}, Q_{ID}) = \widehat{e}(sQ_{ID}, P) = \widehat{e}(d_{ID}, P)$ and using an inversion algorithm to invert f_P, one can recover the secret key of the user associated with ID. Note that if the algorithm is imperfect, one can easily add some randomness by trying to invert $\widehat{e}(P_{pub}, Q_{ID})^r$ for a random r instead.

Hess' Identity-based Signature Scheme. In a similar fashion, it was shown in [11] that one can forge signatures in Hess' identity-based signature scheme [19] if FAPI-2 is easy. In this scheme, let s be the master key, U, $V \coloneqq sU$ be parameters and h, H hash functions. A signature of a message m consists in a pair (u, v) where $v \coloneqq h(m, r)$, $r \coloneqq \widehat{e}(R, U)^k$ for a random k, a random R and where $u \coloneqq vS_{ID} + kR$, with $S_{ID} = sH(ID)$. The signature is verified if $r = \widehat{e}(u, U) \cdot \widehat{e}(H(ID), -V)^v$. To forge a signature, an adversary selects a random r and selects $v = h(m, r)$. Then, using the algorithm for f_U he inverts $r\widehat{e}(H(ID), V)^v = \widehat{e}(u, U)$.

Joux's Tripartite Protocol. In this scheme [23], three parties, A, B and C, pick two elements $Q, R \in \mathbb{G}$ such that $\widehat{e}(Q, R) \neq 1$ and broadcast respectively (aQ, aR), (bQ, bR) and (cQ, cR) in one round after which every party can compute the shared secret key $\widehat{e}(Q, R)^{abc}$ (here, a, b and c are random secrets selected by A, B and C, respectively). Using an algorithm for f_R on $\widehat{e}(aQ, bR)$, one can recover abQ. The shared secret key is then $\widehat{e}(abQ, cR)$.

6 Conclusions

In conclusion, we proved that all the bits of elliptic curve based one-way functions are hard-to-compute. In particular, we proved that all the bits of the pairing-based one-way function are hard-to-compute assuming that CDH is hard. We proved our result for the x-coordinate of the point but the result can trivially be extended to the y-coordinate. In [2], the hardness result is proven for every *segment predicate* and not only for some particular bits. Intuitively, a segment predicate over $\mathbb{Z}/n\mathbb{Z}$ is a predicate that splits $\mathbb{Z}/n\mathbb{Z}$ in $\text{poly}(\log n)$ segments, or a multiplicative shift of it. Our work can be easily extended to prove the hardness of these predicates using the same ECMC code. There is another important aspect of bit security for the specific pairing-based one-way function to be studied: instead of considering a prediction oracle that works on an isomorphism class, we consider an imperfect oracle on an ordinary isogeny class of elliptic curves (i.e., elliptic curves with the same number of points) as was done in [22] for the least significant bits of the Diffie–Hellman secrets for elliptic curves. Note that using techniques based on isogeny graphs and rapid mixing of random walks [21], one can obtain a very strong conclusion for almost every isogeny class: namely, assuming that the oracle works with non-negligible advantage on a non-negligible fraction of all short Weierstrass equations in this class then one can solve FAPI-2 for *every* curve in this class. Proving such a result is the subject of a forthcoming paper.

Acknowledgments. We are grateful to Dan Boneh, David Freeman, Rosario Gennaro, Florian Hess, Eike Kiltz, Arjen Lenstra, Amin Shokrollahi, Igor Shparlinski, Martijn Stam, Serge Vaudenay and Ramarathnam Venkatesan for helpful discussions.

References

1. Akavia, A.: Learning Noisy Characters, Multiplication Codes, and Cryptographic Hardcore Predicates. Ph.D. thesis, Massachusetts Institute of Technology (2008)
2. Akavia, A., Goldwasser, S., Safra, S.: Proving Hard-Core Predicates Using List Decoding. In: FOCS. pp. 146–. IEEE Computer Society (2003)
3. Alexi, W., Chor, B., Goldreich, O., Schnorr, C.: RSA and Rabin Functions: Certain Parts are as Hard as the Whole. SIAM J. Comput. 17(2), 194–209 (1988)
4. Blum, M., Micali, S.: How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. SIAM J. Comput. 13(4), 850–864 (1984)
5. Boneh, D., Franklin, M.: Identity-Based Encryption from the Weil Pairing. In: Kilian [25], pp. 213–229
6. Boneh, D., Shparlinski, I.: On the Unpredictability of Bits of the Elliptic Curve Diffie–Hellman Scheme. In: Kilian [25], pp. 201–212
7. Boneh, D., Venkatesan, R.: Hardness of Computing the Most Significant Bits of Secret Keys in Diffie–Hellman and Related Schemes. In: Koblitz, N. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 1109, pp. 129–142. Springer (1996)
8. Boneh, D., Halevi, S., Howgrave-Graham, N.: The Modular Inversion Hidden Number Problem. In: Boyd, C. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 2248, pp. 36–51. Springer (2001)
9. Boneh, D., Venkatesan, R.: Rounding in Lattices and its Cryptographic Applications. In: Saks, M.E. (ed.) SODA. pp. 675–681. ACM/SIAM (1997)
10. Duc, A., Jetchev, D.: Hardness of Computing Individual Bits for One-way Functions on Elliptic Curves (full version). Cryptology ePrint Archive, Report 2011/329 (2011), `http://eprint.iacr.org/`
11. Galbraith, S., Hess, F., Vercauteren, F.: Aspects of Pairing Inversion. IEEE Transactions on Information Theory 54(12), 5719–5728 (2008)
12. Galbraith, S.D., Hopkins, H.J., Shparlinski, I.: Secure Bilinear Diffie-Hellman Bits. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP. Lecture Notes in Computer Science, vol. 3108, pp. 370–378. Springer (2004)
13. Goldmann, M., Näslund, M., Russell, A.: Complexity Bounds on General Hard-Core Predicates
14. Goldreich, O., Levin, L.: A Hard-Core Predicate for all One-Way Functions. In: STOC. pp. 25–32. ACM (1989)
15. Gonzalez Vasco, M.I., Shparlinski, I.: On the Security of Diffie-Hellman Bits. Electronic Colloquium on Computational Complexity (ECCC) 7(45) (2000)
16. Gonzalez Vasco, M.I., Shparlinski, I.: Security of the most significant bits of the shamir message passing scheme. Math. Comput. 71(237), 333–342 (2002)
17. Håstad, J., Näslund, M.: The Security of Individual RSA Bits. In: FOCS. pp. 510–521 (1998)
18. Håstad, J., Schrift, A., Shamir, A.: The Discrete Logarithm Modulo a Composite Hides $O(n)$ Bits. J. Comput. Syst. Sci. 47(3), 376–404 (1993)
19. Hess, F.: Efficient Identity Based Signature Schemes Based on Pairings. In: Nyberg, K., Heys, H. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 2595, pp. 310–324. Springer (2002)
20. Howgrave-Graham, N., Nguyen, P.Q., Shparlinski, I.: Hidden number problem with hidden multipliers, timed-release crypto, and noisy exponentiation. Math. Comput. 72(243), 1473–1485 (2003)
21. Jao, D., Miller, S.D., Venkatesan, R.: Do All Elliptic Curves of the Same Order Have the Same Difficulty of Discrete Log? In: Roy, B.K. (ed.) ASIACRYPT. Lecture Notes in Computer Science, vol. 3788, pp. 21–40. Springer (2005)

22. Jetchev, D., Venkatesan, R.: Bits Security of the Elliptic Curve Diffie–Hellman Secret Keys. In: Wagner, D. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 5157, pp. 75–92. Springer (2008)
23. Joux, A.: A One Round Protocol for Tripartite Diffie–Hellman. In: Bosma, W. (ed.) ANTS. Lecture Notes in Computer Science, vol. 1838, pp. 385–394. Springer (2000)
24. Joux, A.: The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems. In: Fieker, C., Kohel, D. (eds.) ANTS. Lecture Notes in Computer Science, vol. 2369, pp. 20–32. Springer (2002)
25. Kilian, J. (ed.): Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings, Lecture Notes in Computer Science, vol. 2139. Springer (2001)
26. Li, W.C.W., Näslund, M., Shparlinski, I.: Hidden Number Problem with the Trace and Bit Security of XTR and LUC. In: Yung, M. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2442, pp. 433–448. Springer (2002)
27. Morillo, P., Ràfols, C.: The Security of All Bits Using List Decoding. In: Jarecki, S., Tsudik, G. (eds.) Public Key Cryptography. Lecture Notes in Computer Science, vol. 5443, pp. 15–33. Springer (2009)
28. Nguyen, P.Q., Shparlinski, I.: The Insecurity of the Digital Signature Algorithm with Partially Known Nonces. J. Cryptology 15(3), 151–176 (2002)
29. Nguyen, P.Q., Shparlinski, I.: The Insecurity of the Elliptic Curve Digital Signature Algorithm with Partially Known Nonces. Des. Codes Cryptography 30(2), 201–217 (2003)
30. Nguyen, P.: The dark side of the hidden number problem: Lattice attacks on DSA. In: Proc. Workshop on Cryptography and Computational Number Theory. pp. 321–330 (2001)
31. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: EUROCRYPT. pp. 223–238 (1999)
32. Rabin, M.: Digitalized signatures and public-key functions as intractable as factorization (1979)
33. Rivest, R., Shamir, A., Adleman, L.: A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM 21(2), 120–126 (1978)
34. Sakai, R., Ohgishi, K., Kasahara, M.: Cryptosystems based on pairings. Proceedings of SCIS '00, Okinawa, Japan (2000)
35. Schnorr, C.: Security of Allmost ALL Discrete Log Bits. Electronic Colloquium on Computational Complexity (ECCC) 5(33) (1998)
36. Schrift, A., Shamir, A.: The Discrete Log is Very Discreet. In: STOC. pp. 405–415. ACM (1990)
37. Shparlinski, I.: On the Generalised Hidden Number Problem and Bit Security of XTR. In: Boztas, S., Shparlinski, I. (eds.) AAECC. Lecture Notes in Computer Science, vol. 2227, pp. 268–277. Springer (2001)
38. Shparlinski, I., Winterhof, A.: A hidden number problem in small subgroups. Math. Comput. 74(252), 2073–2080 (2005)
39. Smart, N.P.: Identity-based authenticated key agreement protocol based on weil pairing. Electronics Letters 38(13), 630–632 (2002)