Hardness of Computing Individual Bits for Pairing-based One-way Functions

Alexandre Duc and Dimitar Jetchev

Ecole Polytechnique Fédérale de Lausanne, 1015 Lausanne, Switzerland

Abstract. We prove 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 for 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 pairing-based one-way function are hard–to–compute, assuming that CDH is hard. Our argument uses a list-decoding technique via discrete Fourier transforms due to Akavia–Goldwasser–Safra.

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

1 Introduction

One-way functions 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 the security of the individual bits of the input to pairing-based one-way functions. More precisely, if $\widehat{e}\colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a cryptographic pairing where $\mathbb{G}_T$ is the target group, one can consider the function $f_Q(R) \coloneqq \widehat{e}(R, Q)$ for a fixed $Q \in \mathbb{G}$ (here, $R \in \mathbb{G}$). 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 needed in Joux's three-party one-round key agreement protocol [14] and for the identity-based signature scheme by Hess [12] to be unforgeable. This problem and potential approaches to solve it is studied in [7]. 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 consider cryptographic pairings $\widehat{e}$ on a subgroup $\mathbb{G}$ of the points of an elliptic curve and show that if there exists an algorithm that takes as input $f_Q(R)$ for some hidden element $R \in \mathbb{G}$ and predicts (with non-negligible advantage over a random guess) the kth bit of the x-coordinate of R, then we can invert f_Q (here, the advantage is measured over a random point R in the group as well as a random short Weierstrass equation representing the elliptic curve). Our proof uses methods developed by Akavia et al. [2] based on list-decoding via discrete Fourier transforms. We introduce a special 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.

1.1 Previous Work

The first hard–to–compute predicate in 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 in numerous papers. For instance, Håstad and Näslung showed that every bit in the RSA [19] one-way function is hard–to–compute [10] 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 [11] following the work of Schrift and Shamir [21]. By changing the way the bits are represented, Schnorr showed that almost all of the bits in the DL function are hard–to–compute [20]. A similar hardness result (independent of the bit representation) was proven in [10]. 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 is studied as well [6,13].

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 sets of one-way functions is highly desirable. The first such predicate was the Goldreich–Levin predicate [9]. Given a OWF $f\colon \{0,1\}^n \to \mathcal{R}$, Goldreich and Levin define another OWF $f'\colon \{0,1\}^n \times \{0,1\}^n \to \mathcal{R} \times \{0,1\}^n$ by $f'(x,y) \coloneqq (f(x),y)$. For this OWF, the predicate $P(x,y) \coloneqq \sum_{i=1}^{n} x_i y_i$ is hard–to–compute.

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. Using their methodology, security results can be proven for entire classes of one-way functions. Furthermore, it is elegant to use and hides the cumbersome bit manipulations that appeared in the previous proofs. Their 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 $f\colon \mathcal{D} \to \mathcal{R}$ and a predicate $P(x)$ for $x \in \mathcal{D}$, we construct a code $\mathcal{C}^P$ that associates to $x \in \mathcal{D}$ a codeword $\mathcal{C}_x^P \in \mathcal{C}^P$. If the code verifies two properties, namely if the code is *recoverable* and *Fourier-concentrated*, then the code is *list decodable*. Intuitively, a code is list-decodable when, given access to a corrupted codeword w, there is a PPT algorithm that lists all the codewords that are close to w with some non-negligible probability. This means that given a corrupted version of a codeword $\mathcal{C}_x^P$, one can recover x in polynomial time.

The method is used to prove the security of certain bits in RSA, in the Rabin cryptosystem [18], 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 those functions, where n is the size of the group corresponding to the domain of the one-way function.

In 2009, Morillo and Ràfols [16] 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 $\mathbb{Z}/n\mathbb{Z}$ to the value of the kth bit of its corresponding representative in $[0, n-1]$. They also extended the result to the Paillier [17] trapdoor permutation.

The remaining of the paper is organized as follows: in Section 2 we introduce basic definitions and present our main theorem and its implications. In Section 3, we describe the framework used to prove our theorem. In Section 4 we prove our main theorem. We conclude in Section 5.

2 Main theorem

2.1 Preliminaries

Let p be a prime and let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$. In order to discuss the individual bits of a point on $\mathbb{E}$, we first need to 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 $\mathbb{E}$. Let $\mathcal{W}(\mathbb{E})$ 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 $\mathbb{E}$ 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 $\mathcal{W}(\mathbb{E})$ is in bijection with $\mathbb{F}_p^\times$. For a point $R \in \mathbb{E}(\mathbb{F}_p)$ and $W \in \mathcal{W}(\mathbb{E})$, 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.

Next, we call a function $\nu\colon \mathbb{N} \to \mathbb{R}$ *negligible* if for every constant c, there exists $k_0 \in \mathbb{N}$ such that $\nu(k) < k^{-c}$ for all $k > k_0$. A function $\rho\colon \mathbb{N} \to \mathbb{R}$ will be called *non-negligible*[1] if $1/\rho(n) \geq \mathrm{poly}(\log n)$ for a polynomial independent of n.

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

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

- *Given $x \in X$, one can compute $f(x)$ in polynomial time in* $\log|X|$*,*
- *For every probabilistic polynomial time (PPT) (in* $\log|X|$*) algorithm $\mathcal{A}$, we have* $\Pr[f(z) = y | y = f(x), z = \mathcal{A}(y)] < \nu_{\mathcal{A}}(\log|X|)$ *, where the probability is taken over $x \in X$ chosen uniformly at random and where $\nu_{\mathcal{A}}$ is a negligible function. In other words, the advantage to invert f for every PPT (in* $\log|X|$*) algorithm $\mathcal{A}$ is negligible.*

Remark 1. When we are dealing with complexities, we see domains in their bit representation as would a computer do. For instance, in the above definition, we can see the function f as $f\colon \{0,1\}^n \to \{0,1\}^m$ for $m, n \in \mathbb{N}$. Then f is one-way if one can compute

[1] Note that a function being non-negligible is a stronger requirement than a function not being negligible.

$f(x)$ in poly(n) time and if there is no PPT algorithm that can fin a preimage in poly(n) time.

Definition 2 (maj_g**).** *Given a boolean function* $g\colon X \to \{a_1, a_2\}$*, we define*

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

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.*

Remark 2. 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 $\delta_p(k) := \mathrm{maj}_{B_k}$ for the probability of occurrence of the majority value.

Definition 3 (Hard–to–compute predicate). *A boolean predicate* $P\colon X \to \{0,1\}$ *is* hard–to–compute *with respect to a one-way function* $f\colon X \to \mathcal{R}$ *if there is no PPT (in* $\log|X|$*) algorithm* $\mathcal{A}$ *that can compute* $P(x)$ *from* $f(x)$ *with a non-negligible advantage over the majority value, i.e., such that*

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

for some polynomial that is independent of $|X|$*.*

Intuitively, such a predicate is one bit of information derived from the preimage of f that is not efficiently predictable with non-negligible advantage over a random guess.

Remark 3. Note that very often, the term *hard–core* predicate is misused for *hard–to–compute* predicate. In this paper, we will 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). This was also pointed out in [1, Defn.2.5]

2.2 Pairing-based One-way Functions

We define now a pairing-based function. For an integer n, let $E[n]$ be the subgroup of points of $\mathbb{E}$ of prime order n (the points in $E[n]$ are defined over the 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 $\mathbb{G} := \langle S \rangle$ for an $S \in \mathbb{E}(\mathbb{F}_p)$. To avoid having $e(P,Q) = 1$ for all $P, Q \in \mathbb{G}$, we need to suitably twist e and define what we refer to as a cryptographic pairing:

Definition 4 (Cryptographic pairing). *Let* $\xi\colon \mathbb{E} \to \mathbb{E}$ *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 $P \in \mathbb{E}(\mathbb{F}_p)$ 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 5 (Pairing-based one-way function). *Let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$ with Weierstrass equation $y^2 = x^3 + ax^2 + b$ and let $\mathbb{G} \subset E[n]$ 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 $f_Q(R) := \widehat{e}(R, Q)$, 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 [7,14,5].

2.3 Main Result on Hard–to–compute Bits

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 $\mathbb{E}$ is an elliptic curve over $\mathbb{F}_p$ and $\mathbb{G} \subset \mathbb{E}(\mathbb{F}_p)$ is a cyclic subgroup of order n of cryptographically meaningful size (i.e., $n = \Theta(p)$).

Let Q be a generator of $\mathbb{G}$ and let $R \in \mathbb{G}$ be a hidden point. Suppose that we have an imperfect oracle that takes as input a short Weierstrass equation $W \in \mathcal{W}(\mathbb{E})$ and the value $f_Q(R) \in \mu_n$ 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 $W \in \mathcal{W}(\mathbb{E})$ and over a random point $R \in \mathbb{G}$). We then show that we can efficiently invert f_Q.

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

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

$$\mathrm{Adv}_Q^{x,k}(\mathcal{B}) := \left| \Pr_{\substack{W \in_U \mathcal{W}(\mathbb{E}) \\ R \in_U \mathbb{G}, z}} [\mathcal{B}(W, Q, f_Q(R); z) = B_k((R_W)_x)] - \delta_p(k) \right| > \epsilon \, ,$$

where z is a random variable corresponding to the random coins used by the oracle $\mathcal{B}$. Similarly, we define the advantage of predicting the kth bit of the y-coordinate of the input point R to f_Q.

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 $\mathbb{E}$, $\mathbb{G}$ 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 $\mathcal{B} = \mathcal{B}(W, Q, u; z)$ be an algorithm that takes as input $W \in \mathcal{W}(\mathbb{E})$, $Q \in \mathbb{G}$, $u \in \mu_n$ and outputs an element of $\{\pm 1\}$ in time t. Assume that $\mathrm{Adv}_Q^{x,k}(\mathcal{B}) > \epsilon$. Then there exists an algorithm $\mathcal{A}$ that inverts $f_Q \colon \mathbb{G} \to \mu_n$ in time $t \cdot T(\log p, \frac{1}{\epsilon})$ for some polynomial T that is independent of p, $\mathbb{E}$, $\mathbb{G}$, ϵ and Q.*

Remark 4. One can see from the above theorem that if $\epsilon = \mathrm{poly}(\log p)$ and if $t = \mathrm{poly}(\log p)$ then one can invert the function f_Q efficiently (in time polynomial in $\log p$). This means that either the kth bit of the input to f_Q is hard–to–compute or the function is invertible.

Remark 5. Note that the definition of the function f_Q does not depend on the choice of a short Weierstrass equation $W \in \mathcal{W}(\mathbb{E})$, but only on the curve $\mathbb{E}$ itself. Yet, in order to talk about the individual bits of a point R, one needs to specify a short Weierstrass equation in order to obtain the representation of R. 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 $\mathcal{B}$ 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.

2.4 Consequences of our Result

Our main result 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 [7,14].

Definition 7 (BDH). *Let $\widehat{e} \colon \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear pairing, let P 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 $\langle P, aP, bP, cP \rangle$, $a, b, c \in \mathbb{Z}/n\mathbb{Z}$, compute $\widehat{e}(P,P)^{abc}$.*

The following relations holds. 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 abP given $\langle aP, bP, P \rangle$. 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 implication 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, one can forge signatures [7] in Hess' identity-based signature scheme [12]. if FAPI-2 is easy. In this scheme, let s be the master key and P, $Q \coloneqq sP$ be parameters and h, H hash functions. A signature of a message m consists in a pair (u, v) where $v \coloneqq h(m, r)$, $r \coloneqq \widehat{e}(R, P)^k$ for a random k, a random R and where $u \coloneqq vS_{ID} + kR$, with $S_{ID} = sH(ID)$. The signature is verified if $r = \widehat{e}(u, P) \cdot \widehat{e}(H(ID), -Q)^v$. To forge a signature, an adversary selects a random r and selects $v = h(m, r)$. Then, using the algorithm for f_P he inverts $r\widehat{e}(H(ID), Q)^v = \widehat{e}(u, P)$.

Joux's Tripartite Protocol. In this scheme [14], three parties, A, B and C, pick two elements $P, Q \in \mathbb{G}$ such that $\widehat{e}(P, Q) \neq 1$ and broadcast respectively (aP, aQ), (bP, bQ) and (cP, cQ) in one round after which every party can compute the shared secret key $\widehat{e}(P, Q)^{abc}$ (here, a, b and c are random secrets selected by A, B and C, respectively). Using an algorithm for f_Q on $\widehat{e}(aP, bQ)$, one can recover abP. The shared secret key is then $\widehat{e}(abP, cQ)$.

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 $\langle f, g \rangle \coloneqq 1/|G| \sum_{x \in G} g(x) \cdot \overline{h(x)}$.The ℓ_2-norm on the space $\mathcal{C}(G)$ of all complex valued functions $h \colon G \to \mathbb{C}$ 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 $\mathcal{C}(G)$ (the *Fourier basis*). One can then describe a function $f \in \mathcal{C}(G)$ via its *Fourier expansion* $\sum_{\chi \in \widehat{G}} \langle f, \chi \rangle \chi$. Equivalently, one can define the Fourier transform $\widehat{f} \colon \widehat{G} \to \mathcal{C}$ 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 8 (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 $f \in \mathcal{C}(G)$ using subsets $\mathit{\Gamma} \subset \widehat{G}$ of characters via its *restriction*: $f_{|\mathit{\Gamma}} \coloneqq \sum_{\chi \in \mathit{\Gamma}} \widehat{f}(\chi)\chi$.

3.2 Codes, Fourier Concentration and Recoverability

Throughout, we consider bits that take values in $\{\pm 1\}$ instead of $\{0,1\}$ where we substitute -1 for 0. 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 $\mathcal{C}_x \colon G \to \{\pm 1\}$. 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 now state the formal definition of concentration:

Definition 9 (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.*

We can also state the definition of a recoverable code:

Definition 10 (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* $\exists\, \chi \neq 0, \chi \in \mathrm{Heavy}_\tau(f) \cap \mathrm{Heavy}_\tau(g)$.

Proof. Let $\mathrm{maj}_f =: 1/2 + \beta$, for some $\beta > 0$. Equation (1) implies that $\langle f, g \rangle \geq 2\beta + 2\epsilon$. We also have $\left|\widehat{f}(0)\right| = 2\,\mathrm{maj}_f - 1 = 2\beta$. Since f is Fourier concentrated, there is a set of characters Γ with $|\Gamma| \leq \mathrm{poly}(\log n, 1/\epsilon)$ such that $\|f - f_\Gamma\|_2 =: \epsilon$. Since a Fourier basis is orthonormal, we have by Cauchy–Schwarz

$$\begin{aligned}\sum_{\alpha : \chi_\alpha \in \Gamma} \widehat{f}(\alpha)\widehat{g}(\alpha) = \langle f_{|\Gamma}, g_{|\Gamma} \rangle &\geq \langle f, g \rangle - \langle f - f_{|\Gamma}, g - g_{|\Gamma} \rangle \\ &\geq 2\beta + 2\epsilon - \left| \|f - f_\Gamma\|_2 \cdot \|g - g_\Gamma\|_2 \right| \geq \left|\widehat{f}(0)\right| + 2\epsilon - \epsilon \cdot 1 \\ &\geq \left|\widehat{f}(0)\widehat{g}(0)\right| + \epsilon. \end{aligned} \tag{2}$$

The last inequality holds since $0 \leq \widehat{f}(0) \leq 1$ and $\|g - g_\Gamma\| \leq 1$. Hence, there exists an $\alpha \neq 0$ such that $\chi_\alpha \in \Gamma$ and $\left|\widehat{f}(\alpha)\widehat{g}(\alpha)\right| \geq \frac{\epsilon}{|\Gamma|}$. Note that the term $\left|\widehat{f}(0)\widehat{g}(0)\right|$ in (2) is to avoid $\alpha = 0$. Since f and g are binary functions we have $\left|\widehat{f}(\alpha)\right|, \left|\widehat{g}(\alpha)\right| \geq \frac{\epsilon}{|\Gamma|} =: \tau$. It is now clear that $1/\tau$ is polynomial in ϵ and $1/n$. □

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 via the following theorem in the case $G = \mathbb{Z}/n\mathbb{Z}$:

Theorem 2 ([2, Thm.6]). *There exists a 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 $1 - \delta$ and that has running time* [2]

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

Remark 6. In the language of Akavia et al. [2, §2.3], the group G is a *learnable domain*. It turns out that any finite abelian group G is a learnable domain [1]. A more efficient learning algorithm that improves upon the above theorem is presented in [8].

4 Proof of Theorem 1

We will reduce the proof of Theorem 1 to a list-decoding problem that will be solved using the methods summarized in Section 3. The first step is to properly define a code that reflects our input recovery problem. We explain in Section 4.1 and Appendix A that 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 11).

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 1 and -1 otherwise. A natural way to associate a code to this predicate is to fix a (base) short Weierstrass equation $W \in \mathcal{W}(\mathbb{E})$ and a hidden point R and define the codewords

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

[2] 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.

The above definition is natural since the isomorphism class $\mathcal{W}(\mathbb{E})$ 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 $\mathcal{W}(\mathbb{E})$. In order to study how concentrated these codes are, one 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 Appendix A).

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 (see Appendix B). 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 $\mathcal{B}$ 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 $\mathrm{Adv}_Q^{x,k}(\mathcal{B}) > \epsilon$ for any generator $Q \in \mathbb{G}$.

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 $\mathcal{B}'$ using $\mathcal{B}$ as follows:

$$\mathcal{B}'(W_\lambda, f_Q(R); z) = \begin{cases} \mathcal{B}(W_{r(\lambda)}, f_Q(R); z) & \text{if } \lambda \in \mathbb{F}_p^\times \text{ is a square in } \mathbb{F}_p^\times \\ \mu \in_{\mathcal{D}_k} \{\pm 1\} & \text{otherwise,} \end{cases}$$

where $\mathcal{D}_k$ denotes the distribution for the predicate B_k. We now associate a code to the modified oracle $\mathcal{B}'$ rather than to the original oracle $\mathcal{B}$ 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 11 (Elliptic curve multiplication code (ECMC)). *Let $\mathbb{E}$ be an elliptic curve over $\mathbb{F}_p$ and let $P\colon \mathbb{F}_p \to \{\pm 1\}$ be a binary predicate. Let $\mathbb{G} \subset \mathbb{E}(\mathbb{F}_p)$ be a cyclic subgroup. Given a (base) short Weierstrass equation $W\colon y^2 = x^3 + ax + b$ representing $\mathbb{E}$, the* elliptic curve multiplication code *is the code $\mathcal{C}^{P,W} = \{C_R^{P,W} : \mathbb{F}_p \to \{\pm 1\}\}_{R \in \mathbb{G}}$ 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 7. Reducing the quadratic term λ^2 with λ is a big advantage since (as we will show in Section 4.2 and Appendix B), the Fourier transform of $B_k(\lambda)$ is simpler than the Fourier transform of $B_k(\lambda^2)$, so it is 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 $W \in \mathcal{W}(\mathbb{E})$ be a fixed (base) short Weierstrass equation and let $\mathcal{B}$ be the prediction algorithm from the statement of Theorem 1. There exists a set S of points $R \in \mathbb{G}$ with*

$$|S| \geq \frac{\epsilon}{4\left(1 - \delta_p(k) - \frac{\epsilon}{4}\right)} |\mathbb{G}| \tag{3}$$

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

$$\Pr_{\lambda \in_U \mathbb{F}_p}[w_{R,W}(\lambda) \neq C_R^{B_k,W}(\lambda)] \leq (1-\delta_p(k)) - \frac{\epsilon}{2}, \qquad \forall R \in S \ . \tag{4}$$

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

$$\Pr_{W,R;z}[\mathcal{B}(W, f_Q(R); z) = B_k((R_W)_x)] > \delta_p(k) + \epsilon \ . \tag{5}$$

The latter is equivalent to

$$\Pr_{\lambda,R;z}\left[\mathcal{B}(W_\lambda, f_Q(R); z) = B_k(\lambda^2 \cdot (R_{W_0})_x)\right] > \delta_p(k) + \epsilon \ . \tag{6}$$

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

$$w_{R,W}(\lambda) = \begin{cases} \mathcal{B}(W_{r(\lambda)}, f_Q(R); z) & \text{if } \lambda \text{ is a square} \\ \mu \in_{\mathcal{D}_k} \{\pm 1\} & \text{otherwise,} \end{cases}$$

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 $\mathcal{B}$. Using the randomness of r, we estimate

$$\begin{aligned}
&\Pr_{\lambda,R;z}\left[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)\right] = \\
&\quad = \frac{1}{2}\Pr_{\substack{\lambda \in_U \mathbb{F}_p^2,\\ R;z}}\left[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)\right] + \frac{1}{2}\Pr_{\substack{\lambda \notin \mathbb{F}_p^2,\\ R;z}}\left[w_{R,W}(\lambda) = C_R^{B_k,W}(\lambda)\right] \\
&\quad = \frac{1}{2}\Pr_{\substack{\lambda' \in_U \mathbb{F}_p,\\ R;z}}\left[\mathcal{B}(W_{\lambda'}, f_Q(R); z) = B_k(\lambda'^2 \cdot (R_W))\right] + \frac{1}{2}\delta_p(k) \\
&\quad > \frac{1}{2}\left(\delta_p(k) + \epsilon\right) + \frac{1}{2}\delta_p(k) = \delta_p(k) + \frac{\epsilon}{2} \ .
\end{aligned} \tag{7}$$

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 (4). We now show that the set S satisfies (3). Using (7), we arrive at

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

Since $\delta_p(k) \neq 1$, we obtain (3). $\square$

Remark 8. 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 [16].

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 clever 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 $\delta_{\alpha,k} := 2^k\alpha - (p-1)/2 \bmod p$ 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 $2^k\alpha = -(p+1)/2 + \delta_{\alpha,k} + p\lambda_{\alpha,k}$.

In both 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 Appendix B for the details) 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 $W \in \mathcal{W}(\mathbb{E})$. According to Lemma 2, there exists a subset $S \subset \mathbb{G}$ of size determined by (3) and the property that for any $R' \in S$, we have access to a corrupted codeword $w_{R',W}$ satisfying (4). The problem is that our hidden point $R \in \mathbb{G}$ need not be in S. In order to fix this, we will repeat the following procedure: we pick a random multiple $s \in [1, n-1]$ and set $R' = sR$. It is clear that R' is again a hidden point and (at least if s is invertible mod$n$), 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 ϵ 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 $w_{W,R'}$ is Fourier concentrated, we will 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'.

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

The method used in our proof is very close to the list-decoding method of Akavia et al. [2, Lem.5] and was successfully used by Morillo and Ràfols [16, §6]. The reason it works is that the codeword $\mathcal{C}_R^{B_k,W}$ is τ-concentrated in $\Gamma_{W,R} = \{\chi_\beta : \beta \equiv \alpha \cdot (R_W)_x \bmod p, \chi_\alpha \in \Gamma\}$ where Γ is the set of additive characters $\chi_\alpha : \mathbb{F}_p \to \mathbb{C}^\times$ 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., $\Gamma = \{\chi_\alpha : \lambda_{\alpha,k} = \mathcal{O}(1/\tau), \mu_{\alpha,k} = \mathcal{O}(1/\tau)\}$. Here, we will take τ such that $1/\tau = \mathrm{poly}(\log p)$.

Algorithm 1 The recovery algorithm

```
Input: An additive character $\chi_\beta$ of $\mathbb{F}_p$, a threshold parameter $\tau$ with $1/\tau \in \mathrm{poly}(\log p)$ and
    $z \in \mu_n$ such that $z = f_Q(R)$ for a hidden point $R$.
Output: The hidden point $R \in \mathbb{G}$ such that $f_Q(R) = z$.
1: Calculate $\Gamma \leftarrow \{\alpha \in \mathbb{F}_p : \lambda_{\alpha,k} = \mathcal{O}(1/\tau), \mu_{\alpha,k} = \mathcal{O}(1/\tau)\}$.
2: for $\alpha \in \Gamma \setminus \{0\}$ do
3:    Compute $x \leftarrow \beta\alpha^{-1} \bmod p$
4:    if $y \in \mathbb{F}_p$ exists so that $R = (x, y) \in W(\mathbb{F}_p)$ and $f_Q(R) = z$ then
5:       return $R$
6:    end if
7: end for
8: return false.
```

Algorithm 2 Pairing-based OWF inversion algorithm

```
Input: An elliptic curve $\mathbb{E}/\mathbb{F}_p$, a cyclic subgroup $\mathbb{G} \subset \mathbb{E}$ of prime order $n = \Theta(p)$, an element
    $z = f_Q(R) \in \mu_n$ for a hidden point $R \in \mathbb{G}$ and access to a (noisy) prediction oracle $\mathcal{B}$ for
    $B_k((R_W)_x)$ for $W \in \mathcal{W}(\mathbb{E})$.
Output: The input point $R \in \mathbb{G}$ such that $f_Q(R) = z$.
1: Fix a (base) short Weierstrass equation $W \in \mathcal{W}(\mathbb{E})$.
2: Choose $\tau$ such that $1/\tau = \mathrm{poly}(\log p)$
3: Choose a random function $r : \mathbb{F}_p^2 \to \mathbb{F}_p$.
4: repeat
5:    Choose a random $s \in [1, n-1]$ and set $R' := sR$ (still a hidden point)
6:    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 $\mathcal{B}$.
7:    for $\chi_\beta \in \mathrm{Heavy}_\tau(w_{R',W})$ do
8:       Run Algorithm 1 with $z' := z^s$ to try to recover $R'$
9:       if Algorithm 1 does not fail then
10:         break
11:      end if
12:   end for
13: until $R'$ is recovered
14: return $R \leftarrow s^{-1}R'$
```

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 $1/\operatorname{poly}(\log p)$ (so, we need on average $\operatorname{poly}(\log p)$ trials to exit the **repeat** loop). This completes the proof of Theorem 1.

5 Conclusions

In conclusion, 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 predicates* and not only to some particular bits. Intuitively, a segment predicate over $\mathbb{Z}/n\mathbb{Z}$ is a predicate that splits $\mathbb{Z}/n\mathbb{Z}$ in $\operatorname{poly}(\log n)$ segments, or a multiplicative shift of it. Our work can easily be extended to prove the hardness of these predicates using the same ECMC code.

Acknowledgments. We are grateful to Dan Boneh, David Freeman, Rosario Gennaro, Florian Hess, 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.P.: 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.K.: Identity-Based Encryption from the Weil Pairing. In: Kilian [15], pp. 213–229
6. Boneh, D., Shparlinski, I.: On the Unpredictability of Bits of the Elliptic Curve Diffie–Hellman Scheme. In: Kilian [15], pp. 201–212
7. Galbraith, S.D., Hess, F., Vercauteren, F.: Aspects of Pairing Inversion. IEEE Transactions on Information Theory 54(12), 5719–5728 (2008)
8. Gilbert, A., Muthukrishnan, S., Strauss, M.: Improved time bounds for near-optimal sparse Fourier representations. In: Proceedings of SPIE. vol. 5914, p. 59141A. Citeseer (2005)
9. Goldreich, O., Levin, L.A.: A Hard-Core Predicate for all One-Way Functions. In: STOC. pp. 25–32. ACM (1989)
10. Håstad, J., Näslund, M.: The Security of Individual RSA Bits. In: FOCS. pp. 510–521 (1998)
11. Håstad, J., Schrift, A.W., Shamir, A.: The Discrete Logarithm Modulo a Composite Hides $O(n)$ Bits. J. Comput. Syst. Sci. 47(3), 376–404 (1993)
12. Hess, F.: Efficient Identity Based Signature Schemes Based on Pairings. In: Nyberg, K., Heys, H.M. (eds.) Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 2595, pp. 310–324. Springer (2002)

13. 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)
14. 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)
15. 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)
16. 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)
17. Paillier, P.: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. In: EUROCRYPT. pp. 223–238 (1999)
18. Rabin, M.: Digitalized signatures and public-key functions as intractable as factorization (1979)
19. 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)
20. Schnorr, C.P.: Security of Allmost ALL Discrete Log Bits. Electronic Colloquium on Computational Complexity (ECCC) 5(33) (1998)
21. Schrift, A.W., Shamir, A.: The Discrete Log is Very Discreet. In: STOC. pp. 405–415. ACM (1990)
22. Shparlinski, I.: (2010), Private communication

A The Fourier transform of LSB(λ^2 mod p)

Let $P = \mathrm{LSB}$ be the predicate associated to the list significant bit (only for this section, P will take values in $\{0, 1\}$ coinciding with the values of LSB). We explain how one could analyze directly the code

$$C(\lambda) = \mathrm{LSB}((R_{W_\lambda})_x) = \mathrm{LSB}(\lambda^2 \cdot (R_{W_\lambda})_x)$$

via Fourier transforms and why the concentration of this code is more difficult to establish than the ECMC (analyzed in Appendix B). The argument below was suggested to us by Shparlinski [22].

Let $p = 2m + 1$, let $\omega_p(x) := \exp(2\pi i x/p)$ and let $u = (R_W)_x \in \mathbb{F}_p$. Observe that for any $0 \le k \le m$

$$\sum_{a=0}^{p-1} \omega_p(a(2k + 1 - \lambda^2)) = \begin{cases} p & \text{if } \lambda^2 \equiv 2k + 1 \bmod p \\ 0 & \text{otherwise} . \end{cases}$$

We thus have

$$C(\lambda) = \frac{1}{p} \sum_{k=0}^{m-1} \sum_{a=0}^{p-1} \omega_p(a(2k + 1 - \lambda^2)) . \tag{8}$$

The Fourier transform is then $\widehat{C}(\alpha) = \sum_{\lambda=0}^{p-1} \omega_p(\lambda\alpha) C(\lambda)$, so by (8) we have

$$\widehat{C}(\alpha) = \frac{1}{p} \sum_{a=0}^{p-1} \sum_{\lambda=0}^{p-1} \omega_p(\lambda\alpha - \lambda^2 a u) \sum_{k=0}^{m-1} \omega_p(a(2k + 1)) .$$

Since $\left|\sum_{\lambda=0}^{p-1} \omega_p(\lambda\alpha - \lambda^2 au)\right| = p^{1/2}$ (the latter is a Gauss sum), we have

$$|\widehat{C}(\alpha)| \leq p^{-1/2} \sum_{a=0}^{p-1} \left| \sum_{k=0}^{m-1} \omega_p(a(2k+1)) \right| .$$

The double sum is known to be $\mathcal{O}(p \log p)$, so we have $|\widehat{C}(\alpha)| = \mathcal{O}\left(p^{1/2} \log p\right)$. Unfortunately, the latter is only an upper bound and is not enough to prove concentration and moreover, to find an efficient way to detect the heavy coefficients (as we did for the linearized version in Appendix B).

B The Fourier transform of $B_k(\lambda u \bmod p)$

Let $\omega_p = \exp(2\pi i/p)$ and let $B_k \colon \mathbb{F}_p \to \{\pm 1\}$ be the predicate determined from the kth least significant bit (i.e., $B_k(x) = 1$ if the kth bit of x considered as an element of $[0, p-1]$ is 1 and $B_k(x) = -1$ otherwise). Without loss of generality, we assume that $(R_W)_x \neq 0$.

B.1 Fourier Concentration of $C_R^{P,W}$ and of P

Recall that if P is an arbitrary predicate (in our case, we restrict to $P = B_k$ for some k), then the elliptic curve multiplication code is $C_R^{P,W}(\lambda) = P(\lambda \cdot (R_W)_x)$ for any $\lambda \in \mathbb{F}_p^\times$. We extend the function $C_R^{P,W}$ to $\mathbb{F}_p$ by $C_R^{P,W}(0) = -1$. Hence, we will do Fourier analysis on the additive group $\mathbb{F}_p$ of order p.

Note that if the function $C_R^{P,W}$ is ϵ-concentrated in $\Gamma = \{\chi_\alpha\}$ where the α's are elements of $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$ then P is ϵ-concentrated in the set $\Gamma_{W,R} = \{\chi_\beta \colon \beta \equiv \alpha \cdot (R_W)_x \bmod p\}$. Thus, the question of Fourier-concentration for the ECMC corresponding to a predicate P reduces to the question of the Fourier-concentration of the predicate itself. We thus need to analyze the Fourier coefficients of $B_k \colon \mathbb{F}_p \to \{\pm 1\}$.

B_k is Fourier Concentrated. One way to analyze B_k is to note that it is *block-alternating*, that is, it looks like

$$\begin{array}{ll} k=0: & 1\ {-1}\ 1\ {-1}\ 1\ {-1}\ 1\ {-1} \ldots \\ k=1: & 1\ 1\ {-1}\ {-1}\ 1\ 1\ {-1}\ {-1} \ldots \\ k=2: & 1\ 1\ 1\ 1\ {-1}\ {-1}\ {-1}\ {-1} \ldots \\ & \qquad \vdots \end{array}$$

One can then try to compute and estimate the Fourier coefficients of the function represented by each row and then try to analyze the large coefficients.

Morillo and Ràfols [16] significantly simplify this computation by noticing that if the argument x was simple an integer (not an integer mod$p$) then $B_k(x) + B_k(x + 2^k)$ is identically zero and hence, is a constant function. This fails mod p, but it does not fail too much, so one still has good control over the coefficients. More precisely, let

$$g(x) = \frac{B_k(x) + B_k(x + 2^k)}{2} \ . \tag{9}$$

First, observe that the Fourier transform of $B_k(x)$ is easily related to the one of $g(x)$ by the simple identity

$$\widehat{g}(\alpha) = \frac{\omega_p^{2^k\alpha} + 1}{2}\widehat{B_k}(\alpha), \qquad \alpha \in \mathbb{Z}/p\mathbb{Z} \ . \tag{10}$$

Next, write $p = 2^{k+1}r \pm m$ for a unique $0 \le m < 2^k$ and $r \in \mathbb{Z}$. The Fourier transform $\widehat{g}$ is easy to compute by considering the following two cases:

Case 1: $p = 2^{k+1}r - m$. In this case,

$$g(x) = \begin{cases} 1 & \text{if } x \in [2^{k+1}(r-1) + 2^k - m, 2^{k+1}(r-1) + 2^k - 1], \\ 0 & \text{else,} \end{cases} \tag{11}$$

so one computes

$$\widehat{g}(\alpha) = \begin{cases} \frac{1}{p}\omega_p^{-\alpha(2^{k+1}(r-1)+2^k-m)} \frac{\omega_p^{-\alpha m}-1}{\omega_p^{-\alpha}-1} & \text{if } \alpha \neq 0, \\ \frac{m}{p} & \text{otherwise.} \end{cases} \tag{12}$$

Case 2: $p = 2^{k+1}r + m$. Here,

$$g(x) = \begin{cases} 1 & \text{if } x \in [2^{k+1}r, 2^{k+1}r + m - 1], \\ 0 & \text{else,} \end{cases} \tag{13}$$

and in this case,

$$\widehat{g}(\alpha) = \begin{cases} \frac{1}{p}\omega_p^{-\alpha(2^{k+1}r)} \frac{\omega_p^{-\alpha m}-1}{\omega_p^{-\alpha}-1} & \text{if } \alpha \neq 0, \\ \frac{m}{p} & \text{otherwise.} \end{cases} \tag{14}$$

In the two cases, one obtains (using (10))

$$|\widehat{B_k}(\alpha)|^2 = \frac{1}{p^2} \frac{\sin^2\left(\frac{m\alpha\pi}{p}\right)}{\sin^2\left(\frac{\alpha\pi}{p}\right)\cos^2\left(\frac{2^k\alpha\pi}{p}\right)} \ . \tag{15}$$

Remark 9. It is possible to arrive at this formula directly from analyzing the rows without introducing the function $g(x)$, but one has to be extra careful when calculating the Fourier transform — especially with the last incomplete block of 2^k digits.

Equation (15) allows Morillo and Ràfols to obtain a precise asymptotic bound for $|\widehat{B_k}(\alpha)|$. Recall from Section 4.2 that $a_p(x) = \min(x \bmod p, p - x \bmod p)$, where $y \bmod p$ is taken in $[0, p-1]$. Since for $x \in [-\pi, \pi]$ we have $x^2 - \dfrac{x^4}{3} \le \sin^2 x \le x^2$, one gets

$$\pi^2\left(1 - \frac{\pi^2}{12}\right) a_p(\beta)^2 \le p^2 \sin^2\left(\frac{\beta\pi}{p}\right) \le \pi^2 a_p(\beta)^2 \ . \tag{16}$$

It is now easy to deduce that

$$\left(\frac{1}{\pi^2}-\frac{1}{12}\right)\frac{a_p(m\alpha)^2}{a_p(\alpha)^2 a_p(2^k\alpha-\frac{p}{2})^2} \leq \left|\widehat{B_k}(\alpha)\right|^2 \leq \frac{1}{\pi^2(1-\frac{\pi^2}{12})^2}\frac{a_p(m\alpha)^2}{a_p(\alpha)^2 a_p(2^k\alpha-\frac{p}{2})^2} \ .$$

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 once, we use another idea of Morillo and Ràfols: since p is odd, we can assume that $\alpha \in \left[-\frac{p-1}{2}, \frac{p-1}{2}\right]$ and let $\lambda_{\alpha,k}$ and $\mu_{\alpha,k}$ be as in Section 4.2.

One can now give a lower bound for the desired denominator

$$a_p(\alpha)^2 a_p\left(2^k\alpha-\frac{p-1}{2}\right) \geq \lambda_{\alpha,k}^2\mu_{\alpha,k}^2 r^2 2^{2k+2}\frac{1}{4} \ . \tag{17}$$

From here, one characterizes the asymptotic behavior of $|\widehat{B_k}(\alpha)|$ by

$$|\widehat{B_k}(\alpha)|^2 < \mathcal{O}\left(\frac{1}{\lambda_{\alpha,k}^2\mu_{\alpha,k}^2}\right) . \tag{18}$$

The upshot of having the above representation is that it is very convenient for picking the heavy Fourier coefficients: one simply has to pick $(\lambda_{\alpha,k}, \mu_{\alpha,k})$ in the box $[0, 1/\tau] \times [0, 1/\tau]$ for some $0 < \tau < 1$ satisfying $1/\tau = \text{poly}(\log p)$ and then one can easily show that the function $f_u(\lambda) = B_k(\lambda u \bmod p)$ is τ-concentrated for every $u \in \mathbb{F}_p^\times$.