The Security of Individual Bit for XTR

Kewei Lv(✉), Si-wei Ren, and Wenjie Qin

Institute of Information Engineering
Data Assurance and Communication Security Research Center,
Chinese Academy of Sciences, Beijing 100093, People's Republic of China
kwlu@ucas.ac.cn

Abstract. We consider bit security of public key cryptosystem XTR, presented by Lenstra and Verheul in 2000. Using the list-decoding method, we prove finding one of its pre-image of XTR if single bit of its plaintext is predicted with a non-negligible advantage. That is, every single bit of plaintext of XTR is a hardcore predicate if XTR is one-way.

Keywords: List-decoding · XTR · One-way function · Hardcore predicate

1 Introduction

One-way functions (OWFs) are functions that are easy-to-evaluate but hard-to-invert. It has numerous cryptographic applications. However, its definition does not say much about the security of a particular predicate over its pre-image. For instance, how about the most significant bit of the pre-image of one-way function? If one can guess this bit with a non-negligible advantage beyond 1/2, one might be able to obtain (partial) secret information that is hidden by this one-way function. So to prove that some bit is hard to be predicted is of primary interest. The kind of bit is called hard-core predicate of one-way function and we also say it is hard.

There are three main methods to study a hardcore predicate in our view: The first one is the traditional reduction technique, which is based on the multiplicative or additive homomorphism property of some one-way functions. Specifically, if there exists an oracle with a non-negligible advantage to predict one bit of the pre-image from ciphertext, then one could construct other ciphertexts and invoke the oracle to predict the bit of pre-images of the fresh ciphertexts. Thus we could transform the advantage of oracle into the probability of correctly inverting one-way function. This method only applies to some one-way functions with homomorphism property, such as DL, RSA, Rabin, ECL, Paillier, etc., [1,3,7,8,21], and that $O(\log n)$ bits are simultaneous hard [1,3,22]. All the subsequent works make efforts to prove the simultaneous or individual security of $O(n)$ bits for these candidate one-way functions (see [5,17]). The second method is hidden number problem (HNP) method. If we are given an oracle to predict partial

This work is partially supported by NSF of China No. 61272039.

K.-Y. Lam et al. (Eds.): ICICS 2016, LNCS 9977, pp. 87–98, 2016.
DOI: 10.1007/978-3-319-50011-9_7

relevant information about the secret called hidden number, then we try to find out the hidden number. In this method, we choose a series of samples uniformly and randomly to query oracle. The oracle answers partial information about the hidden number. Then we use these samples and answers of oracle to construct a lattice. Using the lattice reduction algorithm and bounds of exponential sums, we could recover the hidden number in probabilistic polynomial time. This method is uniform but it only shows us there exists a hardcore predicate in a section of a bit string (see [2,14,19,20] and therein). [11] proved that for every one-way function there is a predicate that is hard to be predicated, given the value of any one-way function. The techniques they used indeed is a application of sub-linear time list-decoding Hadamard code. Following this idea, Akavia et al. [4] proposed the third method, a uniform elegant method called list-decoding method, to prove that a predicate is hard-to-compute for some one-way functions, which avoids the cumbersome bit manipulations in 2003. Using it, bit security can be studied for entire classes of functions. The method relies on the construction of a code that encodes the pre-images of one-way function we try to invert. That is, given a one-way function $f : X \to Y$ and a predicate $P(x)$ for $x \in X$, we construct a code C^P that associates $x \in X$ with a codeword C_x^P. If we could have access to a corrupted codeword w (which we can get by an oracle on predicting the bit), there is a PPT algorithm that computes a list of all $x \in X$ such that C_x^P is close to w (usually using Hamming distance). So we can find exact x by exhausting the list, which show the predicate P is hard-to-compute for one-way function f. This method has a strong point, that is, since code C^P associates $x \in X$ with a codeword C_x^P and each codeword is bijective to one pre-image x, the final list must contain all x corresponding to codeword close to w whether or not f is an injective function. The method can be used widely to study bit security of one-way functions, such as RSA, Rabin, EXP, ECDL and so on (see [4,5,10], etc.). [14,15] studied bit security of LUC function (see [6]) over RSA modulo and over an extension field of degree 2 respectively.

As a generalization of LUC to an extension field of degree 6, XTR is presented in [16], which takes advantage of traces to calculate and represent powers of elements of a subgroup of a finite field. Its idea is to gain a secure cryptosystem basing on discrete logarithms problem in $\mathbb{F}_{p^6}$ while the messages exchanged and actual computation are performed over $\mathbb{F}_{p^2}$. It contributes to substantially savings both in computational and communication cost without compromising security when being applied in cryptographic protocols. It has been proved that the security of XTR is computationally equivalent to solving discrete logarithms in $\mathbb{F}_{p^6}$ (see [9,16]). In this paper, we study the bit security of XTR. We use list-decoding method based on list-decoding via discrete Fourier transforms and construct the XTR multiplication code as [15]. We show, if given a probabilistic polynomial time (PPT) algorithm with a non-negligible advantage to predict the k-th bit of pre-image x accessing a noisy codeword that can be list-decoded, we could recover its pre-image of XTR by constructing proper access algorithm with witness, which results in inverting XTR.

Related Works: The first hardcore predicate was found by Blum and Micali [8] for the discrete logarithm problem (DL) over a prime field $\mathbb{F}_p$. Subsequently, the question of finding hardcore predicates of one-way functions was studied extensively. For example, [12] showed that every bit of RSA plaintext is hard-to-compute. Similarly, for exponent function modulo a Blum composite, [13] showed that all the bits are hard-to-compute. By changing representation of the bits, [18] showed that almost all of the bits in the DL function modulo a prime are hard-to-compute. A similar result but independent of the bit representation was proven in [12]. Each proof of these results need cumbersome bit manipulations and algebraic techniques, which only applies to a specific one-way function and have to be significantly modified to be used on another OWF (or even most cannot be used at all). Thus, finding generic method to study hardcore predicates that apply to most general collections of one-way functions is highly desirable.

[4] presented a uniform elegant method to prove that a predicate is hard-to-compute for some one-way functions. This method avoids the cumbersome bit manipulations. Using it, bit security can be studied for entire classes of functions. The method relies on the construction of a code that encodes the pre-images of one-way function we try to invert and can be used to study bit security RSA, Rabin, EXP and ECDL. Indeed, security of the $O(\log n)$ least and most significant bits of these functions are proved, where n is the size of pre-image of one-way function. [17] proved the security of all bits in RSA, Rabin and Paillier function for RSA moduli using a specific analysis of the Fourier coefficients that maps an element of $\mathbb{Z}_N$ to the value of the k-th bit of its corresponding representative in $[0, N-1]$. Bit security of the argument for one-way function based on elliptic curve also is proved using this method in [3]. [10] defined a very natural variation of Diffie-Hellman problem over $\mathbb{F}_{p^2}$ and proved the unpredictability of every single bit of one of the coordinates of the secret DH value is hardcore.

Our Works: It is believed that breaking XTR is computationally equivalent to solving discrete logarithms in $\mathbb{F}_{p^6}$. Using hidden number problem method and tool of lattice, [14] proved that the $\log^{1/2} p$ most significant bits of Diffie-Hellman type variation of XTR are secure, but specific hardcore predicates could not be shown. Furthermore, [14] showed that XTR is not a injective function, so it could not be studied as that of LUC. So far, bit security of XTR should be more studied. Here, we study the bit security of XTR using the list-decoding method, and show the k-th bit of x of XTR is a hardcore predicate. But, using list-decoding method and properties of XTR, we could invert XTR.

Given a PPT algorithm with a non-negligible advantage to predict the k-th bit of x, we first construct a new multiplication code (XTRMC) such that it is list-decodable and accessible. Then we use discrete Fourier transforms on abelian groups to study its Fourier concentration and recoverability, and, based on the learning algorithm of [4], prove that XTRMC is list-decodable and accessible. Finally, we give an inverting algorithm to find pre-image of XTR, which results in inverting XTR. Although XTR is not an injective function, that is, for one value of XTR, there exists three pre-images x, xp^2 and xp^4, we can construct

an access algorithm with witness using Theorem 1 such that its output values contain a witness $S_j(Tr(g^x)) = (Tr(g^{(j-1)x}), Tr(g^{jx}), Tr(g^{(j+1)x}))$. For any $j' \neq j$, $S_j(Tr(g^x)) \neq S_{j'}(Tr(g^x))$ by Sect. 3.2, which assure that access algorithm can not bring another pre-image into list. Thus, each j is bijective to unique accessed value. By learning algorithm, a list of characters is output, which contains heavy characters of corrupted codeword with a high probability. So Inversing algorithm can use recovery algorithm to find a list containing pre-image x such that x is uniquely determined.

Notations: Let $\mathbb{N}$ be the set of natural number and $\mathbb{R}$ be the set of real number. Given an element $x \in \mathbb{F}_q$, define $[x]$ as the representative of the class of x in $[0, q-1]$ and $abs_q(x) = min\{[x], q-[x]\}$. Let A be a set, then $x \in_R A$ denotes that x is chosen randomly, uniformly and independently in A.

2 Organization

The paper is organized as follows: Sect. 3 gives some preliminaries. In Sect. 3, we introduce some basic notions, XTR cryptosystem and properties of discrete Fourier transforms on abelian groups and also present the learning algorithm due to Akavia et al. In Sect. 4, we present our main theorem. In Sect. 5, we summarize our contribution and some extensions are discussed.

3 Preliminaries

3.1 Basic Concepts

Definition 1. *A function $\nu : \mathbb{N} \to \mathbb{R}$ is called negligible if for every constant $c \in \mathbb{R}$ and $c > 0$, there exists a $k_0 \in \mathbb{N}$ such that $|\nu(k)| < k^{-c}$ for all $k > k_0$. A function $\rho : \mathbb{N} \to \mathbb{R}$ is non-negligible if there exists a constant $c \in \mathbb{R}$, $c > 0$ and a $k_0 \in \mathbb{N}$ such that $|\rho(k)| > k^{-c}$ for infinite number of $k > k_0$.*

Definition 2. *A function $f : X \to Y$ is called one-way if it satisfies that: (1) Given $x \in X$, one can compute $f(x)$ in polynomial time in $\log|X|$; (2) For every probabilistic polynomial time in $\log|X|$ algorithm $\mathcal{A}$, there exists a negligible function $\nu_{\mathcal{A}}$ such that $Pr[f(z) = y : y = f(x), z = \mathcal{A}(y)] < \nu_{\mathcal{A}}(\log|X|)$, where the probability is taken over random coin tossing of $\mathcal{A}$ and choice of $x \in X$ uniform and random. That is, for every PPT in $\log|X|$ algorithm $\mathcal{A}$, its advantage of inverting f is negligible.*

Definition 3. *A Boolean function $P : D \to \{\pm 1\}$ is called a predicate for a function f if both share a common domain. In order to do with biased predicates, let $maj_P = \max_{b \in \{\pm 1\}} \Pr_{x \in_R D}[P(x) = b]$ and $minor_P = \min_{b \in \{\pm 1\}} \Pr_{x \in_R D}[P(x) = b]$. Obviously, $maj_P = 1 - minor_P$.*

Definition 4. *We say an PPT algorithm $\mathcal{B}$ efficiently predicts predicate P for f if there exists a non-negligible function ρ, s.t.* $\Pr[\mathcal{B}(f(x)) = P(x)] \geqslant maj_P + \rho(\log|D|)$*, where the probability is taken over random coin tossing of $\mathcal{B}$ and choices of $x \in D$. We say predicate P is hardcore for a one-way function f if it could not be predicted efficiently.*

3.2 XTR

Let $F(c, X) = X^3 - cX^2 + c^pX - 1 \in \mathbb{F}_{p^2}[X]$ be an irreducible polynomial for prime p, then the roots of $F(c, X)$ take the form h, h^{p^2}, h^{p^4} for some $h \in \mathbb{F}_{p^6}$ of order dividing $p^2 - p + 1$ and larger than 3. For $n \in \mathbb{Z}$, we set $c_1 = c$, $c_n = h^n + h^{np^2} + h^{np^4}$. Thus $c_n = Tr(h^n)$, where the trace $Tr(h^n)$ over $\mathbb{F}_{p^2}$ is $\mathbb{F}_{p^2}$-linear, and $c_{-n} = c_p^n$. For any $g \in \mathbb{F}_{p^6}$ which have order q for a prime $q > 3$ and $q|p^2 - p + 1$, its minimal polynomial is $F(Tr(g), X)$. Furthermore, $Tr(g^n) \in \mathbb{F}_{p^2}$ and $F(Tr(g^n), g^n) = 0$ for all n. It is shown that, for such g, the trace value fully specifies g's minimal polynomial, and thus its conjugates, which gives the fundamental idea of XTR. As shown in [16], if $p \equiv 2 \mod 3$, then c_n can be computed efficiently given $c = c_1$ using a recurrence relation, and c_{n-1} and c_{n+1} are obtained at no extra cost as a side result. It is almost three times faster than computing g^n from g using traditional exponentiation methods. Thus, in XTR we replace powers of g by their traces, thereby saving a factor of three both in storage and in computing time. Note that an actual representation of g is not required, and that it suffices to have its trace $Tr(g)$.

Given $Tr(g)$ and the order of g, the subgroup $\langle g \rangle$ generated by g (unknown) is called the XTR group, and function $f : \mathbb{F}_q^* \to \mathbb{F}_{p^2}$ with $f(x) = Tr(g^x)$ is called XTR one-way function. XTR parameters consists of primes p and q as the prior, where $p \equiv 3 \mod 4$, and the trace $Tr(g)$ of a generator of the XTR group. The primes p and q of appropriate sizes can be found using either of the two methods given in [16]. To find a proper $Tr(g)$, it suffices to find $c \in \mathbb{F}_{p^2} \backslash \mathbb{F}_p$ such that $F(c, X) \in \mathbb{F}_{p^2}[X]$ is irreducible, and $c_{(p^2-p+1)/q} = 3$, and set $Tr(g) = c_{(p^2-p+1)/q}$. Since the probability that $c_{(p^2-p+1)/q} \not\equiv 3$ if $F(c, X)$ is irreducible is only $1/q$, usually the irreducible $F(c, X)$ works.

Theorem 1 [14]**.** *Let $S_n(c) = (c_{n-1}, c_n, c_{n+1})$. Given the sum of c of the roots of $F(c, X)$, there exists an algorithm computing the sum c_n of the n-th powers of the roots which takes $8 \log n$ multiplications in $\mathbb{F}_p$.*

3.3 Fourier Transforms

Let G be a finite abelian group and $C(G)$ be the space of all complex valued functions $f : G \to \mathbb{R}$. For any $f, g \in C(G)$, their inner product is defined as $\langle f, g \rangle = \frac{1}{|G|} \sum_{x \in G} f(x)\overline{g(x)}$. The ℓ_2-norm of function f is $\|f\|_2 = \sqrt{\langle f, f \rangle}$. A character of G is a homomorphism $\chi : G \to \mathbb{R}$ satisfying $\chi(x + y) = \chi(x)\chi(y)$ for all $x, y \in G$. The set of all characters of G forms a group $\hat{G}$ called character group. Elements of $\hat{G}$ form a normal orthogonal base of $C(G)$ (i.e. Fourier basis).

Then a function $f \in C(G)$ can be described by its Fourier expansion $f(x) = \sum_{x\in G}\langle f,\chi\rangle\chi$. So its Fourier transform $\hat{f} : \hat{G} \to \mathbb{R}$ is defined by $\hat{f}(\chi) = \langle f,\chi\rangle$. The coefficients $\hat{f}(\chi)$ in the Fourier basis $\{\chi\}_{x\in\hat{G}}$ are called Fourier coefficients of f. We can approximate a function $f \in C(G)$ using subsets $\Gamma \subset \hat{G}$ of characters via its restriction $f_\Gamma = \sum_{x\in\Gamma}\hat{f}(\chi)\chi$. When $G = \mathbb{Z}/n\mathbb{Z}$, characters of G are defined by $\chi(\alpha) = \omega_n^{\alpha x}$ for $\alpha \in \mathbb{Z}_n$ and $\omega_n = e^{\frac{-2\pi i}{n}}$. Weight of a Fourier coefficient $\hat{f}(\chi)$ is $\|\hat{f}(\chi)\|_2^2$. So we define heavy characters of a function f.

Definition 5 (Heavy character). *Given a function $f : G \to \mathbb{R}$ and a threshold τ, $Heavy_\tau(f)$ denotes a set of characters for which weight of the corresponding Fourier coefficient of f is at least τ. That is, $Heavy_\tau(f) = \{x \in \hat{G} | \|\hat{f}(\chi)\|^2 \geqslant \tau\}$.*

Definition 6 (Fourier Concentration). *We say a function $f : \mathbb{Z}_N \to \mathbb{R}$ is Fourier concentrated if, for every $\epsilon > 0$, there exists a set Γ consisting of $poly(\log N/\epsilon)$ characters, so that $\|f - f_\Gamma\|_2^2 = \sum_{\alpha\notin\Gamma}\|\hat{f}(\alpha)\|^2 \leqslant \epsilon$. For simplicity, f is called to be ϵ-concentrated on set Γ.*

The heavy character of f is any character for which the projection of f on it has a large norm. So, given $\tau > 0$ and f, we set $Heavy_\tau(f) = \{\chi_\alpha | \|\hat{f}(\alpha)\|^2 \geqslant \tau\}$.

3.4 Code and List-Decoding Method

To encode elements of $\mathbb{Z}_N$, we will only consider codewords of length N. Thus, a binary code is a subset $C \subset \{\pm 1\}^N$, and each of codeword C_x is a function $C_x : \mathbb{Z}_N \to \{\pm 1\}$ expressed as $(C_x(0), C_x(1), \cdots, C_x(N-1))$.

Definition 7 (Hamming distance). *The normalized Hamming distance between two functions $g, h : \mathbb{Z}_N \to \{\pm 1\}$ is $\Delta(g,h) = \Pr_{x\in\mathbb{Z}_N}[g(x) \neq f(x)]$.*

Definition 8 (List-decodable code). *A code $C = \{C_x : \mathbb{Z}_N \to \{\pm 1\}\}$ is list-decodable if there exists a PPT algorithm which, given access to a corrupted codeword w and on input a threshold δ, ϵ, and 1^N, returns a list $L \supseteq \{x | \Delta(w, C_x) < minor_{C_x} - \epsilon\}$ with a probability $1 - \delta$.*

Definition 9 (Concentration). *We say a code C is concentrated if each of its codewords $C_x \in C$ is Fourier Concentrated.*

Definition 10 (Accessibility). *For each $n \in \mathbb{N}$, assume $I_n \subseteq \{0,1\}^n$ be a countable set and $I = (I_n)_{n\in\mathbb{N}}$. Let $P = (P_i)_{i\in I}$ be a collection of predicates and $\mathcal{F} = \{f_i | D_i \to \{\pm 1\}^*\}_{i\in I}$ be a family of one-way functions. We say that P is accessible with respect to $\mathcal{F}$ if there exists a PPT access algorithm $\mathcal{A}$ such that for all $i \in I_n$, C^{P_i} is accessible to f_i, namely*

1. *Code access: $\forall x, j \in D_i$, $\mathcal{A}(i, f_i(x), j)$ returns $f_i(x')$ such that $C_x^{P_i}(j) = P_i(x')$;*
2. *Well spread: For uniformly distributed $C_x^{P_i} \in C^{P_i}$ and $j \in D_i$, the distribution of x' satisfying $f_i(x') = \mathcal{A}(i, f_i(x), j)$ is statistically close to uniform distribution on D_i;*

3. Bias preserving: For a non-negligible fraction of codeword $C_x^{P_i}$, $|Pr[C_x^{P_i}(j) = 1|j \in D_i] - Pr[P_i(z) = 1|z \in D_i]| \leqslant \nu(n)$, *where* ν *is a negligible function.*

Now we give a sufficient conditions that a code is list-decodable and its detailed explanation can be found in [4].

Theorem 2 (List-decoding method). *Let* $C = \{C_x | C_x : \mathbb{Z}_N \to \{\pm 1\}\}$ *be a concentrated and recoverable code, then* C *is list-decodable.*

3.5 The Learning Algorithm

[4] extends the algorithm of learning heavy Fourier coefficients of a function $f : \{0,1\}^k \to \{0,1\}$ to the function $f : \mathbb{Z}_N^k \to \mathbb{R}$. Specifically, they devise an efficient search procedure to find fewer relevant characters.

Theorem 3 [4]. *There is an algorithm* $\mathcal{A}$ *that, given query access to* $g : \mathbb{Z}_N \to \{\pm 1\}$, $\tau > 0$ *and* $\delta \in (0,1)$, *outputs a list* L *of* $O(1/\tau)$ *characters (each can be encoded in* $\log N$ *bits), that contains* $Heavy_\tau(g)$ *with a probability at least* $1 - \delta$*; and its running time is* $\tilde{O}(\log N) \cdot \ln^2(1/\delta)/\tau^{5.5}$.

Remark. $\tilde{O}(\cdot)$ indicates that terms of complexity which is a polynomial in $\log(1/\tau)$, $\log N$ or $\ln\ln(1/\delta)$ have been omitted. The theorem implies that if we could access a function defined on an abelian group, then it is computationally feasible to obtain a list of all the Fourier coefficients. It is helpful for us to construct the recovering algorithm for XTRMC (see Subsect. 4.1).

4 Main Theorem

Throughout, we set bits values to be $\{\pm 1\}$ instead of $\{0,1\}$. That is, we take values $(-1)^b$ for $b \in \{0,1\}$. For $\mathbb{F}_p$, let $P : \mathbb{F}_p \to \{\pm 1\}$ be the predicate defined by $P(x) = B_i(x)$, where $B_i(x)$ denotes the i-th bit of an element x. We show it is a hardcore predicate for XTR one-way function $f : \mathbb{F}_q^* \to \mathbb{F}_{p^2}$ with $f(x) = Tr(g^x)$.

Definition 11. *Let* p, q *be two primes selected by XTR cryptosystem,* $g \in \mathbb{F}_{p^6}$ *have order* q *dividing* $p^2 - p + 1$ *and larger than* 3. *We say that* $\mathcal{A}$ *has a advantage* $\rho \in (0,1)$ *of predicting the predicate* P *of the argument of XTR one-way function* $f : \mathbb{F}_q^* \to \mathbb{F}_{p^2}$ *with* $f(x) = Tr(g^x)$ *if* $|\Pr[\mathcal{A}(f(x), z) = P(x)] - maj_P| > \rho$. *The probability is taken over* $x \in F_q^*$ *chosen uniformly and randomly, and random coins* z *of* $\mathcal{A}$. *When* ρ *is a non-negligible function, let* $1/\rho = poly(\log q)$.

We state the main theorem:

Theorem 4. *Let* $\rho \in (0,1)$ *be a non-negligible function, both* p *and* q *be primes as above. Let* $f : \mathbb{F}_q^* \to \mathbb{F}_{p^2}$ *with* $f(x) = Tr(g^x)$ *be a XTR one-way function. If there exists an algorithm* $\mathcal{A}$ *to predict* P *with a non-negligible advantage* ρ *in time* $poly(\log q)$, *where* $\rho(\log q) > 0$. *Then there exists an algorithm* INV *that inverts* $f(x)$ *in time* $poly(\log |q|, 1/\rho)$ *for at least* $\frac{\rho}{2}|\mathbb{F}_q^*|$ *of* x.

4.1 Proof of Main Theorem

Before we prove the main theorem, we first construct multiplication code of XTR function (XTRMC).

Definition 12 (XTRMC). *Let p, q, g and $B_i(x)$ be defined as above. We define multiplication code $C^P = \{C_x^P : \mathbb{F}_q^* \to \{\pm 1\}\}_{x\in\mathbb{F}_q^*}$, where $C_x^P(j) = P(j \cdot x \bmod q)$, x is the argument of XTR one-way function f. We denote the code $C = C^P = \{C_x^P\}$.*

Lemma 1. *Let $P : \mathbb{F}_q^* \to \{\pm 1\}$ be a predicate and C^P be accessible to f. If there exists a PPT algorithm $\mathcal{A}_k$ that predicts P from f with advantage ρ', then there exists a set S and $|S| \geqslant \frac{\rho'}{2}|C^P|$ such that $\forall C_x^P \in S$, given $f(x)$, we have query access to a corrupted codeword w_x satisfying $\Delta(w_x, C_x^P) \leqslant minor_{C_x^P} - \rho(k)$, where ρ is a non-negligible function and $k = \log q$.*

Proof. Since C^P is accessible with regard to f, there exists an access algorithm $\mathcal{D}$ satisfying $\mathcal{D}(f(x), j) = f(x')$. Let $w_x(j) = \mathcal{A}_k(\mathcal{D}(f(x), j)))$ and set $\alpha_{x,j} \in \mathbb{F}_q^*$ such that $f(\alpha_{x,j}) = \mathcal{D}(f(x), j)$. By the construction of $\mathcal{D}$, there is only j here. Since the code is well spread and $\mathcal{A}_k$ has an advantage $\rho'(k)$ to predict P, $Pr[\mathcal{A}_k(f(\alpha_{x,j})) = P(\alpha_{x,j})] \geqslant maj_P + \rho'(k)$, where the probability is taken over random coin tosses of $\mathcal{A}_k$ and random choice of $C_x^P \in C^P$ and $j \in \mathbb{F}_q^*$.

Let S be a set satisfying $Pr[\mathcal{A}_k(f(\alpha_{x,j})) = P(\alpha_{x,j})] \geqslant maj_P + \frac{\rho'(k)}{2}$ for all $C_x^P \in S$. Then $|S| \geqslant \frac{\rho'(k)}{2}|C^P|$, s.t. $\forall C_x^P \in S$, $Pr[\mathcal{A}_k(f(\alpha_{x,j})) = P(\alpha_{x,j})] \geqslant maj_P + \frac{\rho'(k)}{2}$. Note that the code is bias preserving, $|maj_{C_x^P} - maj_P| \leqslant \nu'(k)$, where ν' is a negligible function. So $\mathcal{A}_k$ has a non-negligible function $\rho(k) = \frac{\rho'(k)}{2} - \nu'(k)$ s.t. $\forall C_x^P \in S$, $Pr[\mathcal{A}_k(f(\alpha_{x,j})) = P(\alpha_{x,j})] \geqslant maj_P + \rho(k)$. Namely, $\forall C_x^P \in S$, $\Delta(w_x, C_x^P) \leqslant minor_{C_x^P} - \rho(k)$. This completes the proof.

Fourier Concentration of XTRMC. In order to bound the size of the fourier coefficients $\hat{P}(\alpha)$ and sieve the heavy ones, we could use the method of [17] to obtain a careful analysis of function $P(x)$ and find out the concentrated set of XTRMC accurately.

Let $q = r2^{i+1} \pm m$ for $m \in (0, 2^i)$. For $\alpha \in [-\frac{q-1}{2}, \frac{q-1}{2}]$ and function $g(x) = \frac{P(x+2^i)+P(x)}{2}$, its Fourier transform coefficient is $\hat{g}(\alpha) = \frac{w_q^{2^i\alpha}+1}{2}\hat{P}(\alpha)$, where $w_p = e^{\frac{2\pi i}{q}}$. For both $x \in [(r-1)2^{i+1} + 2^i - m, (r-1)2^{i+1} + 2^i - 1]$ and $x \in [2^{i+1}r, 2^{i+1}r + m - 1]$, we compute $\hat{g}(\alpha)$ respectively and obtain in both cases $|\hat{P}(\alpha)|^2 = \frac{1}{q^2} \cdot \frac{\sin^2(\frac{m\alpha x}{q})}{\sin^2(\frac{\alpha x}{q})\sin^2(\frac{2^i\alpha x}{q})}$. So $|\hat{P}(\alpha)|^2 \leqslant \frac{1}{\pi^2(1-\pi^2/12)^2} \cdot \frac{abs_q^2(m\alpha)}{abs_q^2(\alpha)abs_q^2(2^i\alpha - q/2)}$.

To be asymptotic $|\hat{P}(\alpha)|^2$ closer, we set $2^i\alpha = \frac{q-1}{2} + \delta_\alpha + q\lambda_\alpha$ such that $\delta_\alpha = 2^i\alpha - \frac{q-1}{2} \mod q$ and $\lambda_\alpha \in [0, 2^{i-1} - 1]$ for $\alpha \in [0, \frac{q-1}{2}]$; and $\delta_\alpha = 2^i\alpha + \frac{q-1}{2} \mod q$ and $\lambda_\alpha \in [0, 2^{i-1} - 1]$ for $\alpha \in [-\frac{q-1}{2}, 0]$, where λ_α is integer.

Proposition 1. *For all $\alpha \in F_p^*$, we have $abs_q(\alpha) = (2\lambda_\alpha + 1) \pm \mu_\alpha$, where λ_α is define as above and $\mu_\alpha \in [0, r]$ is a integer. Furthermore, $|\hat{P}(\alpha)|^2 < O(\frac{1}{\lambda_\alpha^2\mu_\alpha^2})$.*

Proof. $\forall \alpha \in F_q^*$, $abs_q(\alpha) = k_r r \pm \mu_r$, where $\mu_r \in [-r/2, r/2]$. If $k_r = 2k+1$, then $abs_q(\alpha) = (2k+1)r \pm \mu_r$. So we can set $\lambda_\alpha = k$ and $\mu_\alpha = \mu_r$. Else, if $k_r = 2k$, then $abs_q(\alpha) = (2k+1)r - (r - \mu_r)$ for $\mu_r > 0$ and $abs_q(\alpha) = (2k-1)r + r - \mu_r = (2(k+1)-1)r - (r - \mu_r)$ for $\mu_r < 0$. So λ_α and μ_α can be set. Furthermore, since $abs_q^2(\alpha) abs_q^2(2^i\alpha - \frac{q-1}{2}) \geqslant \lambda_\alpha^2 \cdot \mu_\alpha^2 \cdot r^2 \cdot 2^{2i+2} \cdot 1/4$, $|\hat{P}(\alpha)|^2 < O(\frac{1}{\lambda_\alpha^2 \mu_\alpha^2})$.

Lemma 2. *Let P be a predicate defined as above. Then P is τ-concentrated on $\Gamma = \{\chi_\alpha | \lambda_\alpha < O(1/\tau), \mu_\alpha < O(1/\tau)\}$.*

Proof. The proof is almost identical to Theorem 7 in [17], we present it here for completeness. At first, we give an injective map

$$\begin{array}{ccc} \pi : \left[-\frac{q-1}{2}, \frac{q-1}{2}\right] & \to & \left[0, 2^{i-1}-1\right] \times [0, r] \times \{\pm 1\} \times \{\pm 1\} \\ \alpha & \to & (\lambda_\alpha, \quad \mu_\alpha, \quad s_\alpha, \quad s_\delta) \end{array}$$

where $s_\delta = sgn(\delta)$, $s_\alpha = sgn(\alpha)$ for sign function $sgn(\cdot)$.

All characters of Z_N consists of $\Gamma \cup \Gamma_0 \cup \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$, where $\Gamma = \{\chi_\alpha | \lambda_\alpha \leqslant O(1/\tau), \mu_\alpha \leqslant O(1/\tau)\}$, $\Gamma_0 = \{\chi_\alpha | \lambda_\alpha = 0, \mu_\alpha \geqslant O(1/\tau)\}$, $\Gamma_1 = \{\chi_\alpha | \lambda_\alpha \geqslant O(1/\tau), \mu_\alpha = 0\}$, $\Gamma_2 = \{\chi_\alpha | \lambda_\alpha \geqslant 1, 1 \leqslant \mu_\alpha \leqslant O(1/\tau)\}$, $\Gamma_3 = \{\chi_\alpha | \mu_\alpha \geqslant 1, 1 \leqslant \lambda_\alpha \leqslant O(1/\tau)\}$, $\Gamma_4 = \{\chi_\alpha | \lambda_\alpha \geqslant O(1/\tau), \mu_\alpha \geqslant O(1/\tau)\}$. We bound the sum of $|\hat{P}(\alpha)|^2$:

$$\sum_{\chi_\alpha \in \Gamma_0} |\hat{P}(\alpha)|^2 \leqslant O(m^2) \sum_{\chi_\alpha \in \Gamma_0} \frac{1}{abs_N^2(2^i\alpha - \frac{N-1}{2})} < O(m^2) \sum_{\chi_\alpha \in \Gamma_0} \frac{1}{(2^i \mu_\alpha)^2} < O(\tau),$$

$$\sum_{\chi_\alpha \in \Gamma_1} |\hat{P}(\alpha)|^2 \leqslant O(r^2) \sum_{\chi_\alpha \in \Gamma_1} \frac{1}{abs_N^2(\alpha)} < O(r^2) \sum_{\chi_\alpha \in \Gamma_1} \frac{1}{\lambda_\alpha^2} < O(\tau) \quad \text{and}$$

$$\begin{aligned} &\textstyle\sum_{\chi_\alpha \in \Gamma_2} |\hat{P}(\alpha)|^2 + \sum_{\chi_\alpha \in \Gamma_3} |\hat{P}(\alpha)|^2 + \sum_{\chi_\alpha \in \Gamma_4} |\hat{P}(\alpha)|^2 \\ &\leqslant \sum_{1 \leqslant \mu_\alpha \leqslant k} \frac{1}{\mu_\alpha^2} \Big(\sum_{\lambda_\alpha > k} \frac{1}{\lambda_\alpha^2} \Big) + \sum_{1 \leqslant \lambda_\alpha \leqslant k} \frac{1}{\lambda_\alpha^2} \Big(\sum_{\mu_\alpha \geqslant k} \frac{1}{\mu_\alpha^2} \Big) + \sum_{\mu_\alpha \geqslant k} \frac{1}{\mu_\alpha^2} \Big(\sum_{\lambda_\alpha > k} \frac{1}{\lambda_\alpha^2} \Big) \leqslant O(\tau) \end{aligned}$$

So the predicate is τ-concentrated on $\Gamma = \{\chi_\alpha | \lambda_\alpha < O(1/\tau), \mu_\alpha < O(1/\tau)\}$.

Recoverability of XTRMC. We have proved C^P is τ-concentrated on Γ. To prove C^P is list-decodable, we need C^P is recoverable. Namely, there exists a PPT recovery algorithm on input a character χ_β and a threshold parameter τ to output a list L containing $x \in F_q^*$ such that $\chi_\beta \in Heavy_\tau(C_x^P)$.

Lemma 3. *For any prime q, C^P is recoverable.*

Proof. By Lemma 2, C^P is τ-concentrated in $\Gamma' = \{\chi_\beta | \beta = \alpha \cdot x \mod q, \chi_\alpha \in \Gamma\}$, where $\Gamma = \{\chi_\alpha | \lambda_\alpha < O(1/\tau), \mu_\alpha < O(1/\tau)\}$. The recovery algorithm (Table 1) will output a list containing $x \in F_q^*$ such that $\chi_\beta \in Heavy_\tau(C_x^P)$.

As C_x^P is τ-concentrated in Γ', $\chi_\beta \in Heavy_\tau(C_x^P)$ implies $\chi_\beta \in \Gamma'$ and thus $\beta = \alpha \cdot x \mod q$ for $\lambda_\alpha < O(1/\tau)$ and $\mu_\alpha < O(1/\tau)$. The algorithm outputs list $L = \{x | x = \beta/\alpha \mod q, \chi_\alpha \in \Gamma\}$ containing all x such that $\chi_\beta \in Heavy_\tau(C_x^P)$. Since we can choose parameter $1/\tau \in poly(\log q)$, the length of list and running time of the recovery algorithm will be in $poly(\log q/\tau)$.

Combining Lemmas 2 and 3, we prove C^P is list-decodable for any q.

Table 1. The recovery algorithm

Input: A character χ_β, a threshold parameter τ with $1/\tau \in poly(\log q)$.
Output: A list L containing $x \in F_q^*$ such that $\chi_\beta \in Heavy_\tau(C_x^P)$.

1. $L = \emptyset$, $L_\alpha = \emptyset$.
2. Calculate $\Gamma = \{\chi_\alpha | \lambda_\alpha < O(1/\tau), \mu_\alpha < O(1/\tau)\}$.
3. For $\chi_\alpha \in \Gamma$ do.
4. $\quad x = \beta/\alpha \mod q$.
5. $\quad L_\alpha = \{x\}$.
6. $\quad L = L \cup L_\alpha$.
7. End for.
8. Return L.

Accessibility w.r. to XTR. Assuming discrete logarithm problem in $\mathbb{F}_{p^6}$ is intractable, we have XTR collection of one-way functions

$$\mathrm{XTR} = \{\mathrm{XTR}_{(p,q,g)}(x) = Tr(g^x)\}_{(p,q,g)\in I},$$

where $I = \{(p,q,g)|$ Both p, q are primes, $g \in \mathbb{F}_{p^6}$ of order q s.t $q|p^2 - p + 1\}$.

Lemma 4. *The code $C^P = \{C_x^P\}_{x\in\mathbb{F}_q^*}$ is accessible to XTR one-way function.*

Proof. We construct the access algorithm $\mathcal{D}$:

On input p, q, g, j and $XTR_{p,q,g}(x)$ For $j \in \mathbb{F}_q^*$, we can use Theorem 1 to compute $S_j(Tr(g^x)) = (Tr(g^{(j-1)x}), Tr(g^{jx}), Tr(g^{(j+1)x})) \in \mathbb{F}_{p^2}^3$ and return $Tr(g^{jx})$. Output $Tr(g^{jx})$ and $S_j(Tr(g^x))$ as its witness.

Fixed $x \in \mathbb{F}_q^*$ and j, for any $j' \in \{1, p^2, p^4\}$, both $Tr(g^{xj'}) = XTR_{p,q,g}(x') = Tr(g^{jx})$ and $S_j(Tr(g^x)) = S_{j'}(Tr(g^x))$ should hold. Since $S_j(Tr(g^x)) \neq S_{j'}(Tr(g^x))$ for $j \neq j'$, the other two choices is discarded. So the distribution of x' on $\mathbb{F}_q^*$ is close to uniform, and the code is well-spread and bias-preserving.

Continuing to Prove Theorem 4. Since C^P is list-decodable and there exists a non-negligible codewords w_x which is accessible, by Theorem 2, the predicate P is a hardcore for the XTR one-way function. Indeed, if there exists an oracle $\mathcal{A}$ which has a non-negligible advantage to predict $P(x) = B_i(x)$, then we could construct a PPT algorithm INV (see Table 2) which returns a list with a high probability containing at least one pre-image of XTR. Using $\mathcal{A}$, we can have access to C^P and there are at least $\frac{\rho}{2}\left|\mathbb{F}_q^*\right|$ of x by Lemma 1. Since the learning algorithm in step 3 runs in time $\tilde{O}(\log q) \cdot \ln^2(1/\delta)/\tau^{5.5}$ and the recovery algorithm in step 4 runs in time $poly(\log q/\tau)$, the INV algorithm runs in time $poly(\log q, 1/\rho)$. This completes the proof of Theorem 4.

Table 2. XTR OWF inverse algorithm

Input: A query access to XTR function $f : \mathbb{F}_q^* \to \mathbb{F}_{p^2}$ with $f(x) = Tr(g^x)$ and its witness $S_{j_0}(Tr(g^x))$ for some $j_0 > 1$, where both p, q are primes, g is of order q s.t $q|p^2 - p + 1$; an oracle $\mathcal{A}$ has a non-negligible advantage to predict $P(x)$ given $y = f(x)$.
Output: $z \in \mathbb{F}_q^*$ such that $f(z) = y$.

1. Choose τ such that $1/\tau \in poly(\log q)$, $\delta \in (0, 1)$. Let $L = \emptyset$ initially.
2. Use the oracle $\mathcal{A}$ to obtain a corrupted codeword w_x.
3. Invoke the learning algorithm(theorem 3) with δ and τ. Output all the heavy coefficients of the w_x, i.e., $\Gamma = \{\beta | \chi_\beta \in Heavy_\tau(w_x)\}$.
4. For $\chi_\beta \in Heavy_\tau(w_x)$, run recovery algorithm(Table 1) and return all x such that $\chi_\beta \in Heavy_\tau(C_x)$ and set $L_\beta = \{x | \chi_\beta \in Heavy_\tau(C_x)\}$. Let $L = L \cup L_\beta$.
5. End for.
6. For $z \in L$.
7. Compare $f(z)$ with y and $S_{j_0}(Tr(g^z)) = S_{j_0}(Tr(g^x))$. If they all are equal, then return z.
8. End for.
9. Exit and failed.

5 Remark and Conclusion

In [14], DH-type XTR was only studied by HNP, but it is much rougher than list-decoding method. In this paper, we study the bit security of the XTR one-way function by the list-decoding method. Although XTR is not injective, using XTR inverse algorithm (Table 2), the pre-image z can be found such that $f(z) = y$. Indeed, the access algorithm we constructed have an output with a witness. It is the witness that assures that pre-images are bijective to codewords such that exact pre-image could be found when it is list-decoded correctly. Thus we prove that the individual bit is hardcore for XTR one-way function, which is also considered as a supplement to the work of the Akavia et al. For bit security of XTR variation of Diffie-Hellman problem, this method is also applied.

References

1. 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)
2. 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 1996. LNCS, vol. 1109, pp. 129–142. Springer, Heidelberg (1996). doi:10.1007/3-540-68697-5_11
3. Duc, A., Jetchev, D.: Hardness of computing individual bits for one-way functions on elliptic curves. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 832–849. Springer, Heidelberg (2012). doi:10.1007/978-3-642-32009-5_48
4. Akavia, A., Goldwasser, S., Safra, S.: Proving hard-core predicates using list decoding. In: FOCS, vol. 3, pp. 146–156, October 2003

5. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00457-5_28
6. Bleichenbacher, D., Bosma, W., Lenstra, A.K.: Some remarks on Lucas-based cryptosystems. In: Coppersmith, D. (ed.) CRYPTO 1995. LNCS, vol. 963, pp. 386–396. Springer, Heidelberg (1995). doi:10.1007/3-540-44750-4_31
7. Blum, L., Blum, M., Shub, M.: A simple secure pseudo-random number generator. SIAM J. Comput. **15**(2), 364–383 (1986)
8. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudorandom bits. SIAM J. Comput. **13**(4), 850–864 (1984)
9. Brouwer, A.E., Pellikaan, R., Verheul, E.R.: Doing more with fewer bits. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 321–332. Springer, Heidelberg (1999). doi:10.1007/978-3-540-48000-6_26
10. Fazio, N., Gennaro, R., Perera, I.M., Skeith lll, W.E.: Hard-core predicates for a Diffie-Hellman problem over finite fields. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8043, pp. 148–165. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40084-1_9
11. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: Proceedings of the 21st STOC, pp. 25–32 (1989)
12. Hastad, J., Naslund, M.: The security of individual RSA bits. In: Proceedings of 39th FOCS, pp. 510–519 (1998)
13. Hästad, 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)
14. Li, W.-C.W., Näslund, M., Shparlinski, I.E.: Hidden number problem with the trace and bit security of XTR and LUC. In: Yung, M. (ed.) CRYPTO 2002. LNCS, vol. 2442, pp. 433–448. Springer, Heidelberg (2002). doi:10.1007/3-540-45708-9_28
15. Lv, K., Ren, S.: Bit security for Lucas-based one-way function. In Proceedings of 2014 Ninth Asia Joint Conference on Information Security, pp. 111–118. IEEE (2014)
16. Lenstra, A.K., Verheul, E.R.: The XTR public key system. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 1–19. Springer, Heidelberg (2000). doi:10.1007/3-540-44598-6_1
17. Morillo, P., Ràfols, C.: The security of all bits using list decoding. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 15–33. Springer, Heidelberg (2009). doi:10.1007/978-3-642-00468-1_2
18. Schnorr, C.P.: Security of almost all discrete log bits. Electr. Colloq. Comput. Complex. Univ. Trier **TR98-033**, 1–13 (1998)
19. Shparlinski, I.E.: Playing "hide-and-seek" in finite fields: the hidden number problem and its applications. In: Proceedings of 7th Spanish Meeting on Cryptology and Information Security, vol. 1, pp. 49–72 (2002)
20. Su, D., Wang, K., Lv, K.: The bit security of two variants of Paillier trapdoor function. Chin. J. Comput. **33**(6), 1050–1059 (2010)
21. Su, D., Lv, K.: Paillier's trapdoor function hides $\Theta(n)$ bits. Sci. China Inf. Sci. **54**(9), 1827–1836 (2011)
22. Vazirani, U.V., Vazirani, V.V.: Efficient and secure pseudo-random number generation (extended abstract). In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 193–202. Springer, Heidelberg (1985). doi:10.1007/3-540-39568-7_17