Cryptanalysis of a (Somewhat) Additively Homomorphic Encryption Scheme Used in PIR

Tancrède Lepoint[1] and Mehdi Tibouchi[2]

[1] CryptoExperts, tancrede.lepoint@cryptoexperts.com
[2] NTT Secure Platform Laboratories, tibouchi.mehdi@lab.ntt.co.jp

Abstract. Private Information Retrieval (PIR) protects users' privacy in outsourced storage applications and can be achieved using additively homomorphic encryption schemes. Several PIR schemes with a "real world" level of practicality, both in terms of computational and communication complexity, have been recently studied and implemented. One of the possible building block is a conceptually simple and computationally efficient protocol proposed by Trostle and Parrish at ISC 2010, that relies on an underlying secret-key (somewhat) additively homomorphic encryption scheme, and has been reused in numerous subsequent works in the PIR community (PETS 2012, FC 2013, NDSS 2014, etc.).

In this paper, we show that this encryption scheme is not one-way: we present an attack that decrypts arbitrary ciphertext without the secret key, and is quite efficient: it amounts to applying the LLL algorithm twice on small matrices. Used against *existing practical instantiations* of PIR protocols, it allows the server to recover the users' access pattern in a matter of seconds.

1 Introduction

Cloud computing has gained widespread importance and adoption in recent years. One of the main concerns of cloud security is user privacy. Encryption of data at rest is a first step towards the protection of user data in such a setting. In combination with fully homomorphic encryption [Gen09], cloud servers can continue to provide services to users while only manipulating encrypted data. However, encryption of users' data is only a partial solution to cloud security. Private Information Retrieval (PIR), introduced by Chor, Goldreich, Kushilevitz and Sudan [CKGS98], allows a user to retrieve its data in a manner that prevents the server from knowing which data was retrieved. In a PIR protocol with a single server, the only way to information theoretically hide the users' access pattern is to send the entire data back at each query. (By considering several servers with a copy of the data, secure information theoretic PIR protocols with smaller communication complexity can be achieved.)

In [KO97], Kushilevitz and Ostrovsky presented the first (single database) *computational* PIR (cPIR) where security is achieved against a computationally bounded server. More generally, they present a construction of a cPIR from an additively homomorphic encryption scheme, i.e. from an encryption scheme that allows to publicly compute an encryption of the sum of the plaintexts from the ciphertexts (see, e.g., [DSH14] for one example). Since this result, numerous protocols of cPIR have been proposed.

In this paper, we focus on the cPIR protocol proposed by Trostle and Parrish at ISC 2010, and more precisely on its underlying (somewhat) additively homomorphic encryption scheme—the TP scheme. Due to its conceptual simplicity and computational efficiency, this scheme was used as a building block of other PIR protocols [BPMÖ12,MBC13,MBC14,EÖM14], and to a private spectrum availability information retrieval protocol [GZL$^+$13]. In particular, it was implemented in Java by Mayberry et al. [MBC13,MBC14] to demonstrate the practicality of PIR in a "real world" setting.

Our Contributions. In this paper we focus on the TP scheme. We present a concrete attack showing that the scheme is not one-way: one can in fact recover the plaintext of any given ciphertext. Our attack is based on the notion of orthogonal lattice introduced by Nguyen and Stern at Crypto '97 [NS97], and it is very efficient: it amounts to applying LLL reduction twice on lattices of small dimension. We implemented it and carried out the attack on the parameters suggested in [TP10] as well as those used in existing implementations of the TP scheme [MBC13,MBC14]. Every time, it succeeded in a matter of seconds on a desktop computer. Our technique can be described as follows.

The secret key in the TP scheme consists of a pair (b, m) where m is a large secret prime and $b \in \mathbb{Z}_m$ is a secret odd multiplicative mask. A bit μ is encrypted as $c \in \mathbb{Z}_m$ given by

$$c = b \cdot (2r + \mu) \bmod m = b \cdot e + m \cdot k\,,$$

where r is some random small noise value, $e = 2r + \mu$, and k is quotient in the Euclidean division of c by m.

Now consider a vector $\boldsymbol{c} \in \mathbb{Z}^t$ of ciphertexts associated with the plaintext vector $\boldsymbol{\mu}$, and let $\boldsymbol{e}, \boldsymbol{k}$ be the corresponding vectors of "noisy plaintexts" and Euclidean quotients:

$$\boldsymbol{c} = b \cdot \boldsymbol{e} + m \cdot \boldsymbol{k}.$$

The first step of our attack is similar to [NS98,CNT10]: by applying lattice reduction on the lattice of vectors orthogonal to $\boldsymbol{c}$ in $\mathbb{Z}^t$, we can obtain a short basis $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-2}\}$ of the lattice orthogonal to $L = \mathbb{Z}\boldsymbol{e} \oplus \mathbb{Z}\boldsymbol{k}$, and taking the orthogonal again, we get a short basis $\{\boldsymbol{x}, \boldsymbol{y}\}$ of L. In particular, $\boldsymbol{e} = u\boldsymbol{x} + v\boldsymbol{y}$ for some integers u, v. As a result, $\boldsymbol{\mu} = \boldsymbol{e} \bmod 2$ is equal modulo 2 to one of $\boldsymbol{0}$, $\boldsymbol{x}$, $\boldsymbol{y}$ or $\boldsymbol{x} + \boldsymbol{y}$, and can thus be recovered with probability at least 1/4.

We stress that our attack differs from [NS98,CNT10] in at least two respects: on the one hand, the modulus m is secret, which is the main reason why Trostle and Parrish believed their scheme to be secure; and on the other hand, the vectors $\boldsymbol{x}, \boldsymbol{y}$ are actually *too short* to allow us to recover $\boldsymbol{r}$ and $\boldsymbol{k}$ (or the integers u, v above) directly, due to an exponentially large search space. To the best of our knowledge, this last point is an unheard of situation in the realm of orthogonal lattice techniques so far, and makes this particular attack quite interesting from a theoretical cryptanalytic viewpoint as well.

Outline. In Section 2, we recall Trostle and Parrish's scheme, some of its applications to PIR and provide some background on orthogonal lattices. In Section 3, we describe our attack against the scheme. And finally, we assess its practicality in Section 4. (For the reader's convenience, the source code of the attack is also made available in Appendix A).

2 Preliminaries

For any integer $n \in \mathbb{Z}$, we denote by $[n]$ the set $\{1, \ldots, n\}$. Vectors are denoted in bold characters. For any vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{Z}^t$, $\|\boldsymbol{x}\|$ denotes the Euclidean norm of $\boldsymbol{x}$, $[\boldsymbol{x}]_n = (x_i \bmod n)_{i \in [t]}$ denotes the componentwise reduction of the coefficients of $\boldsymbol{x}$ modulo n, and $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ denotes the scalar product of $\boldsymbol{x}$ and $\boldsymbol{y}$.

2.1 Trostle and Parrish's SHE Scheme

In this section, we present the secret-key encryption scheme of Trostle and Parrish [TP10], a key ingredient of their PIR protocol.

Let λ be the security parameter, η the bit-length of the secret modulus m and ρ the bit-length of the noise in a fresh ciphertext (both η and ρ are functions of λ).

$\mathsf{KeyGen}(1^\lambda)$**.** On input the security parameter λ, generate a η-bit secret modulus m, and an odd secret random invertible mask $b \in \mathbb{Z}_m$. Output $\mathsf{sk} = \{m, b\}$.
$\mathsf{Encrypt}(\mathsf{sk}, \mu \in \{0, 1\})$. On input the secret key $\mathsf{sk} = \{m, b\}$ and a message μ, sample $r \overset{u}{\leftarrow} [0, 2^{\rho-1})$ and output $c = b \cdot (2 \cdot r + \mu) \bmod m$.
$\mathsf{Decrypt}(\mathsf{sk}, c)$. On input the secret key $\mathsf{sk} = \{m, b\}$, a ciphertext c, output $\mu = (b^{-1} \cdot c \bmod m) \bmod 2$.

This scheme is (somewhat) additively homomorphic, i.e. can be used to compute the sum of (a bounded number of) values only manipulating encrypted values. More precisely consider two ciphertexts $c_1 = b \cdot (2r_1 + \mu_1) \bmod m$ and $c_2 = b \cdot (2r_2 + \mu_2) \bmod m$ where r_1 (resp. r_2) is a ρ_1-bit (resp. ρ_2-bit) integer. One can homomorphically add the ciphertexts: the ciphertext $c_1 + c_2$ is an encryption of $\mu_1 + \mu_2 \bmod 2$ under a $(\max(\rho_1, \rho_2) + 1)$-bit noise. Note that the ciphertext noise must remain smaller than m to maintain correctness.

Remark 1. In [TP10], the scheme is also described with message space $\mathbb{Z}_N$ for any $N \geqslant 2$. An encryption of $\mu \in \mathbb{Z}_N$ is an integer c such that $c = b \cdot (N \cdot r + \mu) \bmod m$ with $r \overset{u}{\leftarrow} [0, 2^\rho/N)$, and we recover μ from c by $\mu = (b^{-1} \cdot c \bmod m) \bmod N$. We discuss extension of our attack to this setting in Section 3.3.

2.2 Applications to PIR

Due to its simplicity and computational efficiency (the homomorphic addition being a simple addition of integers), the TP scheme is used as building block in several PIR protocols [TP10,GZL$^+$13,MBC13,MBC14,EÖM14]. Below, we briefly describe the original PIR protocol, and the protocols *implemented* by Mayberry et al. [MBC13,MBC14]. In the following, assume a user want to recover a file among t files of bitsize s from a server.

In their initial paper, Trostle and Parrish described the following protocol (we present the variant in which a user wants to recover one row of a database that is a square bit array). The database D is a $t \times s$ matrix of bits, and a user send $\boldsymbol{c} = (c_1, \ldots, c_t)$ to the server, where $c_i \leftarrow \mathsf{Encrypt}(1)$ if the user requests the i-th row of D and $c_j \leftarrow \mathsf{Encrypt}(0)$ for $j \neq i$. The server then multiplies the j-th row by c_j for all j, adds all the rows and sends the result to the user. Since the TP scheme is additively homomorphic, the users recovers a vector $\boldsymbol{c}'$ such that c'_k encrypts the k-th coefficient of the i-th row of D.

In [MBC13], Mayberry et al. considered the TP scheme with plaintext space $\mathbb{Z}_N$ with $N = 2^\ell$, and use the fact that if $c \leftarrow \mathsf{Encrypt}(1)$ and $\mu \in \mathbb{Z}_N$, then $\mu \cdot c$ encrypts μ (this is a special property of the TP scheme – and could be obtained from any somewhat homomorphic encryption scheme [Gen09] by "encrypting" μ). The database D is a table of $t \times (s/\ell)$ ℓ-bit integers. The rest of the protocol is as above.

Finally, in [MBC14], Mayberry et al. combined the previous approach with an Oblivious RAM protocol to obtain an ORAM-like protocol in which the communication complexity is significantly improved compared to previous ORAM protocols, at the cost of some computational complexity on the server side (coming from the PIR protocol).[3]

Note that in all three protocols, a user seeking to recover the i-th row of the database will send a vector of ciphertexts

$$\boldsymbol{c} = (c_1, \ldots, c_t)\,,$$

where c_i encrypts 1 and the c_j's for $j \neq i$ encrypt 0. Without loss of generality (see Remark 3 page 7), we assume that $N = 2$ and we describe an attack which allows to recover the index of the queried row efficiently.

2.3 The Orthogonal Lattice

In this section, we recall some useful facts about the notion of orthogonal lattice and LLL [NS97,NS01,LLL82].

[3] The TP scheme was *one* possible building block of this protocol; therefore the latter might still be secure when instantiated with a different homomorphic encryption scheme.

Let t be an integer. For any vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{Z}^t$, we say that $\boldsymbol{u}$ and $\boldsymbol{v}$ are orthogonal if $\langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0$, and we denote it $\boldsymbol{u} \perp \boldsymbol{v}$. For any vector $\boldsymbol{u} \in \mathbb{Z}^t$, we denote $\boldsymbol{u}^\perp$ the set of vectors in $\mathbb{Z}^t$ orthogonal to $\boldsymbol{u}$. More generally, if L is a lattice in $\mathbb{Z}^t$, its orthogonal lattice $L^\perp$ is defined as the set of vectors in $\mathbb{Z}^t$ orthogonal to the points in L, i.e.

$$L^\perp = \left\{ \boldsymbol{v} \in \mathbb{Z}^t \mid \forall \boldsymbol{u} \in L, \langle \boldsymbol{u}, \boldsymbol{v} \rangle = 0 \right\}.$$

We have the following theorems [NS97]:

Theorem 1. *If L is a lattice in $\mathbb{Z}^t$, then* $\dim(L) + \dim(L^\perp) = t$.

Theorem 2. *There exists an algorithm which, given any basis $\{\boldsymbol{b}_1, \ldots, \boldsymbol{b}_d\}$ of a lattice L in $\mathbb{Z}^t$ of dimension d, outputs an LLL-reduced basis of the orthogonal lattice $L^\perp$, and whose running time is polynomial with respect to t, d and any upper bound on the bit-length of the $\|\boldsymbol{b}_j\|$'s.*

Most of the vectors of a reduced basis of $L^\perp$ are quite shorts, with norm around $\det(L^\perp)^{1/(t-\dim(L))}$. In practice, a very simple algorithm for Theorem 2 consists in a single call to LLL [LLL82]; we refer the reader to [NS97] for details, and will use that algorithm in Section 4.

3 Breaking the One-Wayness of the Scheme

In this section, we show that the scheme described in Section 2.1 is not one-way.

3.1 Overview

Let $\mathsf{sk} = \{m, b\} \leftarrow \mathsf{KeyGen}(1^\lambda)$ be a secret key, and $\boldsymbol{c} = (c_i)_{i \in [t]} \in \mathbb{Z}^t$ be a vector of ciphertexts such that $c_i \leftarrow \mathsf{Encrypt}(\mathsf{sk}, \mu_i)$ where $\boldsymbol{\mu} = (\mu_i)_{i \in [t]} \in \{0,1\}^t$. We can write, for each $i \in [t]$:

$$c_i = b \cdot (2r_i + \mu_i) \bmod m = b \cdot e_i + m \cdot k_i$$

with $e_i = 2r_i + \mu_i$ and k_i the quotient in the Euclidean division of $b \cdot e_i$ by m. Thus, if we let $\boldsymbol{e} = (e_i)_{i \in [t]}$ and $\boldsymbol{k} = (k_i)_{i \in [t]}$ we have:

$$\boldsymbol{c} = b \cdot \boldsymbol{e} + m \cdot \boldsymbol{k}. \tag{1}$$

Now a rough sketch of the attack is as follows. Consider short vectors $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-2} \in \mathbb{Z}^t$ orthogonal to $\boldsymbol{c}$. For all $j \in [t-2]$, we get that

$$0 = \langle \boldsymbol{u}_j, \boldsymbol{c} \rangle = b \cdot \langle \boldsymbol{u}_j, \boldsymbol{e} \rangle + m \cdot \langle \boldsymbol{u}_j, \boldsymbol{k} \rangle.$$

If the $\|\boldsymbol{u}_j\|$'s are sufficiently short, the fact that $\boldsymbol{e}$ and $\boldsymbol{k}$ are also short yields:

$$\langle \boldsymbol{u}_j, \boldsymbol{e} \rangle = 0 \quad \text{and} \quad \langle \boldsymbol{u}_j, \boldsymbol{k} \rangle = 0$$

for all j, and hence $\boldsymbol{e}$ and $\boldsymbol{k}$ belong to the orthogonal $L^\perp$ of the lattice L spanned by $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-2}$. Then, if $\{\boldsymbol{x}, \boldsymbol{y}\}$ is any basis of $L^\perp$ (easy to find from the $\boldsymbol{u}_j$'s), there are only three possible non-zero linear combinations of $\boldsymbol{x}$ and $\boldsymbol{y}$ modulo 2 (namely $\boldsymbol{x}, \boldsymbol{y}$ and $\boldsymbol{x} + \boldsymbol{y}$), and we know that the vector of plaintexts $\boldsymbol{\mu} = \boldsymbol{e} \bmod 2$ is either one of them or equal to $\boldsymbol{0}$. The encryption scheme is therefore not one-way.

3.2 Applying Orthogonal Lattice Techniques

The first steps of our attack resemble the attack of Nguyen and Stern [NS98] against the Itoh-Okamoto-Mambo cryptosystem [IOM97], and similar attacks such that the one of Coron et al. on EMV signatures [CNT10]. See also [NT12] for a relevant theoretical discussion. In particular, a simple observation common with those previous attacks is that a vector orthogonal to $\boldsymbol{c}$ is either large, or orthogonal to both $\boldsymbol{e}$ and $\boldsymbol{k}$.

Lemma 1. *Let $\boldsymbol{u} \in \mathbb{Z}^t$. If $\boldsymbol{u} \perp \boldsymbol{c}$, then ($\boldsymbol{u} \perp \boldsymbol{e}$ and $\boldsymbol{u} \perp \boldsymbol{k}$), or $\|\boldsymbol{u}\| \geqslant m/(t^{1/2} \cdot 2^{\rho+1})$.*

Proof. Let $\boldsymbol{u} \in \mathbb{Z}^t$ such that $\|\boldsymbol{u}\| < m/(t^{1/2} \cdot 2^{\rho+1})$ and $\boldsymbol{u} \perp \boldsymbol{c}$. We have that $|\langle \boldsymbol{u}, \boldsymbol{e} \rangle| \leqslant \|\boldsymbol{u}\| \cdot \|\boldsymbol{e}\| < m$. Now,

$$0 = \langle \boldsymbol{u}, \boldsymbol{c} \rangle = b \cdot \langle \boldsymbol{u}, \boldsymbol{e} \rangle + m \cdot \langle \boldsymbol{u}, \boldsymbol{k} \rangle \,,$$

and since $\gcd(b, m) = 1$, this yields that $\langle \boldsymbol{u}, \boldsymbol{e} \rangle = 0$, and then that $\langle \boldsymbol{u}, \boldsymbol{k} \rangle = 0$. □

From Theorem 2, it is possible to compute a reduced basis $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-1}\}$ of $\boldsymbol{c}^\perp \subset \mathbb{Z}^t$ of vectors orthogonal to $\boldsymbol{c}$ in $\mathbb{Z}^t$. From Lemma 1, we get that for all $j \in [t-1]$, there are two possibilities:

(1) $\boldsymbol{u}_j \perp \boldsymbol{e}$ and $\boldsymbol{u}_j \perp \boldsymbol{k}$, in which case $\boldsymbol{u}_j$ belongs to the lattice $\{\boldsymbol{e}, \boldsymbol{k}\}^\perp$ of vectors in $\mathbb{Z}^t$ orthogonal to both $\boldsymbol{e}$ and $\boldsymbol{k}$;
(2) $\|\boldsymbol{u}_j\| \geqslant m/(t^{1/2} \cdot 2^{\rho+1})$.

Since $\boldsymbol{e}$ and $\boldsymbol{k}$ are linearly independent, the first possibility cannot hold for all $j \in [t-1]$ (for reasons of dimensions) and the largest $\boldsymbol{u}_j$, say $\boldsymbol{u}_{t-1}$, must satisfy $\|\boldsymbol{u}_{t-1}\| \geqslant m/(t \cdot 2^{\rho+1})$. Now the other vectors form a lattice $L = \mathbb{Z}\boldsymbol{u}_1 \oplus \cdots \oplus \mathbb{Z}\boldsymbol{u}_{t-2}$ of rank $t-2$ and of volume

$$V = \mathrm{vol}(L) \approx \frac{\mathrm{vol}(\boldsymbol{c}^\perp)}{\|\boldsymbol{u}_{t-1}\|} = \frac{\|\boldsymbol{c}\|}{\|\boldsymbol{u}_{t-1}\|} \leqslant t \cdot 2^{\rho+1} \,,$$

which can heuristically be expected to behave like a random lattice. In particular, assuming the Gaussian heuristic, we should have

$$\|\boldsymbol{u}_j\| = \mathcal{O}(\sqrt{t-2} \cdot V^{1/(t-2)}) = \mathcal{O}(t^{1/2} \cdot V^{1/(t-2)}) \qquad \text{for } j \in [t-2].$$

Thus, the condition for $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{t-2}$ all being orthogonal to $\boldsymbol{e}, \boldsymbol{k}$ becomes:

$$\left(t \cdot 2^{\rho+1}\right)^{1+\frac{1}{t-2}} \ll m.$$

Taking logarithms and ignoring logarithmic factors, this means:

$$t \gtrsim 2 + \frac{\rho+1}{\eta-\rho-1} = \frac{2-\alpha}{1-\alpha} \quad \text{where} \quad \alpha = \frac{\rho+1}{\eta}. \tag{2}$$

Assuming this condition (2) is satisfied, the vectors $\boldsymbol{e}$ and $\boldsymbol{k}$ belong to $L^\perp$. Denote $\{\boldsymbol{x}, \boldsymbol{y}\}$ an arbitrary basis of that lattice. Since $\boldsymbol{e} \in L^\perp$, there exist integers $u, v \in \mathbb{Z}$ such that $\boldsymbol{e} = u\boldsymbol{x} + v\boldsymbol{y}$. This yields

$$\boldsymbol{\mu} = [\boldsymbol{e}]_2 \in \{\boldsymbol{0}, [\boldsymbol{x}]_2, [\boldsymbol{y}]_2, [\boldsymbol{x}+\boldsymbol{y}]_2\} \,,$$

which breaks the one-wayness of the scheme (and in applications to e.g. PIR, the case $\boldsymbol{\mu} = \boldsymbol{0}$ is excluded, so we really find $\boldsymbol{\mu}$ as one of three possible bit vectors).

Remark 2. It is interesting to note that we can find a (short) basis such that

$$\|\boldsymbol{x}\|, \|\boldsymbol{y}\| = \mathcal{O}(\sqrt{2} \cdot V^{1/2}) = \mathcal{O}(t^{1/2} \cdot 2^{\rho/2})\,.$$

Quite surprisingly these vectors $\boldsymbol{x}, \boldsymbol{y}$ (of the “doubly orthogonal” lattice) are actually *too short* to provide a direct break, in the sense that the coefficients (u, v) of $\boldsymbol{e}$ in the basis $\{\boldsymbol{x}, \boldsymbol{y}\}$ of $L^{\perp}$ are actually exponentially large (of $\approx \rho/2$ bits), so that we cannot hope to recover the vector $\boldsymbol{e}$ itself from this data.

In fact, $\boldsymbol{e}$ and $\boldsymbol{k}$ are in some sense hidden, since for any pair (u', v') of coprime integers of the same size as (u, v), we can complete the “fake” vector $\boldsymbol{e}' = u'\boldsymbol{x} + v'\boldsymbol{y}$ into a basis $\{\boldsymbol{e}', \boldsymbol{k}'\}$ of $L^{\perp}$ of the correct size, and deduce a “fake” secret key (m', b') also of the correct size such that $\boldsymbol{c} = b' \cdot \boldsymbol{e}' + m' \cdot \boldsymbol{k}'$. This is, to the best of our knowledge, an unheard of situation for orthogonal lattice attacks!

But again, our attack does not need to recover $\boldsymbol{e}$ completely to break the one-wayness of the scheme. Since the scheme encrypts bits, we only need to recover $[\boldsymbol{e}]_2$, and that is easy.

3.3 Larger Message Space

As mentioned in Remark 1, instead of $\mathbb{Z}_2$, the message space could be $\mathbb{Z}_N$ for $N \geqslant 2$. Let N_0 be the smallest prime factor of N (if N is prime, $N_0 = N$).

Using the notation of previous section, our attack recovers a basis $\{\boldsymbol{x}, \boldsymbol{y}\}$ of $L^{\perp}$. Since $\boldsymbol{e} \in L^{\perp}$, there exists $u, v \in \mathbb{Z}$ such that $\boldsymbol{e} = u\boldsymbol{x} + v\boldsymbol{y}$. Now there are at most N^2 pairs (u mod N, v mod N). Therefore, we can recover the plaintext by a random guess with probability at least N^{-2}, and the scheme is therefore not one-way provided that $N = \text{poly}(\lambda)$.

Similarly, if $N_0 = \text{poly}(\lambda)$, the same attack shows that the scheme is not IND-CPA-secure, because for every component μ_i of $\boldsymbol{\mu}$ divisible by N_0, the corresponding components x_i, y_i of $\boldsymbol{x}, \boldsymbol{y}$ are both divisible by N_0 with significant probability $1/N_0^2$, whereas this cannot happen if μ_i is not divisible by N_0.

Finally, for a superpolynomial choice of N_0, our attack allows to recover small messages $\boldsymbol{\mu}$. Denote $\boldsymbol{e} = N \cdot \boldsymbol{r} + \boldsymbol{\mu}$. If the $\|\boldsymbol{u}_j\|$'s and $\|\boldsymbol{\mu}\|$ are sufficiently small (e.g. such that $\langle \boldsymbol{u}_j, \boldsymbol{\mu} \rangle < N$ for all j), then $\boldsymbol{u}_j \perp \boldsymbol{e}$ yields $\boldsymbol{u}_j \perp \boldsymbol{r}$ and $\boldsymbol{u}_j \perp \boldsymbol{\mu}$ for all j. Therefore $\boldsymbol{\mu}$ is likely to be the shortest vector of $L^{\perp}$ and can be efficiently recovered by lattice reduction. Our attack seems only ineffective against a superpolynomial choice of N_0 when encrypting large messages.

Remark 3. In the PIR protocols of [MBC13,MBC14], the message space is chosen to be $N = 2^{\ell}$ for $\ell \geqslant 1$. From the discussion above, it follows that one can recover the queried index file to the server.[4]

[4] Note that taking selecting N as a superpolynomial prime does not thwart the attack since the users sends encryption of *bits*.

4 Implementation of the Attack

Since the attack is heuristic, one needs to assess its behavior in practice. We implemented the attack described in Section 3 using SAGE [S$^+$14]; the source code is provided in Appendix A.

4.1 Attack Summary

Assume that, for t bits $\mu_1, \dots, \mu_t$, we know the ciphertexts $c_1, \dots, c_t$. Then we can heuristically recover $\boldsymbol{\mu} = (\mu_i)_{i \in [t]}$ as follows.

(1) Define $\boldsymbol{c} = (c_1, \dots, c_t) \in \mathbb{Z}^t$.

(2) Compute an LLL-reduced [LLL82] basis $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_{t-1}\}$ of the lattice $\boldsymbol{c}^\perp \subset \mathbb{Z}^t$ of vectors in $\mathbb{Z}^t$ orthogonal to $\boldsymbol{c}$. This is done by applying LLL to the lattice in $\mathbb{Z}^{1+t}$ generated by the rows of the following matrix:

$$\begin{pmatrix} \gamma \cdot c_1 & 1 & & 0 \\ \vdots & & \ddots & \\ \gamma \cdot c_t & 0 & & 1 \end{pmatrix},$$

where γ is a large constant, and keeping only the t last coefficients of each resulting vector.

(3) Compute an LLL-reduced basis $\{\boldsymbol{x}, \boldsymbol{y}\}$ of the orthogonal $L^\perp$ to the lattice $L = \mathbb{Z}\boldsymbol{u}_1 \oplus \cdots \oplus \mathbb{Z}\boldsymbol{u}_{t-2} \subset \mathbb{Z}^t$ of rank $t-2$. Again, this amounts at applying LLL to the lattice in $\mathbb{Z}^{t-2+t}$ generated by the rows of

$$\begin{pmatrix} \gamma' \cdot u_{1,1} & \cdots & \gamma' \cdot u_{t-2,1} & 1 & & 0 \\ \vdots & & \vdots & & \ddots & \\ \gamma' \cdot u_{1,t} & \cdots & \gamma' \cdot u_{t-2,t} & 0 & & 1 \end{pmatrix},$$

where γ' is a large constant, and keeping only the t last coefficients of each resulting vector.

(4) Output $\boldsymbol{0}$, $[\boldsymbol{x}]_2$, $[\boldsymbol{y}]_2$ and $[\boldsymbol{x}+\boldsymbol{y}]_2$.

Heuristically, this attack allows us to guess $\boldsymbol{\mu}$ with probability at least $1/4$. Moreover, if we know $t-1$ coefficients of $\boldsymbol{\mu}$ and have to guess the last one (as in a security game, or in PIR protocols where only one bit is 1), the previous method is likely for large enough t's to make us guess it with probability 1.

4.2 Experimental Results

We ran our attack against the parameters suggested by Trostle and Parrish [TP10] and the parameters *used* in the proof-of-concept implementations in Java of Mayberry et al. [MBC13,MBC14] – we give these parameters in Table 1.

Table 2a gives the success probability of our attack in function of the parameters and the number of ciphertext t used. As expected when $(\log_2 m - \rho)$ becomes small, one will need more ciphertexts for the attack to be successful. Finally, our attack proves to be really efficient against parameters of Table 1, i.e. parameters used in "real world" implementations of PIR protocols [TP10,MBC13,MBC14] – cf. Table 2b.

Set of parameters	$\log_2(m)$	ρ
Set-Ia [TP10]	200	188
Set-Ib [TP10]	400	385
Set-IIa [MBC13]	4513	4113
Set-IIb [MBC13]	2195	1155
Set-IIIa [MBC14]	522	384
Set-IIIb [MBC14]	396	296

Table 1: Parameters sets.

# ciphertexts t	10	20	40
Set-Ia	0%	0%	100%
Set-Ib	0%	0%	100%
Set-IIa	0%	100%	100%
Set-IIb	100%	100%	100%
Set-IIIa	100%	100%	100%
Set-IIIb	100%	100%	100%

(a) Attack success probability

# ciphertexts t	10	20	40
Set-Ia	–	–	1.45s
Set-Ib	–	–	2.92s
Set-IIa	–	3.51s	38.1s
Set-IIb	88ms	919ms	10.0s
Set-IIIa	28ms	289ms	3.04s
Set-IIIb	23ms	220ms	2.33s

(b) Efficiency of the attack

Table 2: Attack success probability and efficiency for each parameter set, in function of the number t of ciphertexts used for the attack (average value over 500 experiments on a single 3.4Ghz Intel Core i7 CPU).

References

[BPMÖ12] Erik-Oliver Blass, Roberto Di Pietro, Refik Molva, and Melek Önen. PRISM - privacy-preserving search in mapreduce. In Simone Fischer-Hübner and Matthew Wright, editors, *PETS 2012*, pages 180–200. Springer, 2012.

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[CNT10] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Fault attacks against EMV signatures. In Josef Pieprzyk, editor, *CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 208–220. Springer, 2010.

[DSH14] Yarkın Doröz, Berk Sunar, and Ghaith Hammouri. Bandwidth efficient PIR from NTRU. In *WAHC 2014*, volume 8438. Springer, 2014.

[EÖM14] Kaoutar Elkhiyaoui, Melek Önen, and Refik Molva. Privacy preserving delegated word search in the cloud. In Mohammad S. Obaidat, Andreas Holzinger, and Pierangela Samarati, editors, *SECRYPT 2014*, pages 137–150. SciTePress, 2014.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC 2009*, pages 169–178. ACM, 2009.

[GZL$^+$13] Zhaoyu Gao, Haojin Zhu, Yao Liu, Muyuan Li, and Zhenfu Cao. Location privacy in database-driven Cognitive Radio Networks: Attacks and countermeasures. In *INFOCOM 2013*, pages 2751–2759. IEEE, 2013.

[IOM97] Kouichi Itoh, Eiji Okamoto, and Masahiro Mambo. Proposal of a fast public key cryptosystem. In Carlisle Adams and Mike Just, editors, *SAC 1997*, pages 224–230, 1997.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *FOCS '97*, pages 364–373. IEEE Computer Society, 1997.

[LLL82] Arjen K. Lenstra, Hendrik W. Lenstra Jr., and László Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261(4):515–534, 1982.

[MBC13] Travis Mayberry, Erik-Oliver Blass, and Agnes Hui Chan. PIRMAP: efficient private information retrieval for MapReduce. In Ahmad-Reza Sadeghi, editor, *FC 2013*, pages 371–385. Springer, 2013.

[MBC14] Travis Mayberry, Erik-Oliver Blass, and Agnes Hui Chan. Efficient private file retrieval by combining ORAM and PIR. In *NDSS 2014*, 2014.

[NS97] Phong Q. Nguyen and Jacques Stern. Merkle-Hellman revisited: A cryptoanalysis of the Qu-Vanstone cryptosystem based on group factorizations. In Burton S. Kaliski Jr., editor, *CRYPTO '97*, pages 198–212. Springer, 1997.

[NS98] Phong Q. Nguyen and Jacques Stern. Cryptanalysis of a fast public key cryptosystem presented at SAC '97. In Stafford E. Tavares and Henk Meijer, editors, *SAC'98*, pages 213–218. Springer, 1998.

[NS01] Phong Q. Nguyen and Jacques Stern. The two faces of lattices in cryptology. In Joseph H. Silverman, editor, *CaLC 2001*, pages 146–180. Springer, 2001.

[NT12] Phong Q. Nguyen and Mehdi Tibouchi. Lattice-based fault attacks on signatures. In Marc Joye and Michael Tunstall, editors, *Fault Analysis in Cryptography*, Information Security and Cryptography, pages 201–220. Springer, 2012.

[S$^+$14] W. A. Stein et al. *Sage Mathematics Software (Version 6.2)*. The Sage Development Team, 2014. `http://www.sagemath.org`.

[TP10] Jonathan T. Trostle and Andy Parrish. Efficient computationally private information retrieval from anonymity or trapdoor groups. In Mike Burmester, Gene Tsudik, Spyros S. Magliveras, and Ivana Ilic, editors, *ISC 2010*, pages 114–128. Springer, 2010.

A Code in SAGE

```
def orthoListVec(li):
  "Returns a list of vectors that are orthogonal to vectors in li"

  nli=len(li)
  t=len(li[0])
  eta=int(N(log(max(li[0]),2)))+1

  M=matrix(ZZ,t,t+nli)
  A=2^(eta*nli)

  for i in range(nli):
    for j in range(t):
      M[j,i]=A*li[i][j]

  for i in range(t):
    M[i,i+nli]=1

  out=[]

  reducedM=M.LLL()
  for v in reducedM:
    if max([v[i] for i in range(nli)])==0:
      out.append(v[nli:])
  return out

def testPIR(eta=522,rho=384,t=5):
  m=random_prime(2^eta,proof=False)
  b=randint(1,m-1)

  e=vector(ZZ,[ZZ.random_element(2^(rho+1)) for i in range(t)])
  c=vector(ZZ,[b*e[i] % m for i in range(t)])

  timer=cputime()
  li = orthoListVec([c])
  out=orthoListVec(li[:t-2])
  x = out[0]
  y = out[1]

  if [e[i]%2 for i in range(t)] not in [[0 for _ in range(t)],
  [x[i]%2 for i in range(t)], [y[i]%2 for i in range(t)],
  [(x[i]+y[i])%2 for i in range(t)]]:
    print "Unsuccessful"
  else:
    print "Successful"
  print "CPU Time: ",cputime(timer)
```