Controlling Data Disclosure in Computational PIR Protocols

Ning Shang
Purdue University
West Lafayette, IN, USA
nshang.cs.purdue@gmail.com

Gabriel Ghinita
Purdue University
West Lafayette, IN, USA
gghinita@cs.purdue.edu

Yongbin Zhou[*]
Purdue University
West Lafayette, IN, USA
yongbin@purdue.edu

Elisa Bertino
Purdue University
West Lafayette, IN, USA
bertino@cs.purdue.edu

ABSTRACT

Private Information Retrieval (PIR) protocols allow users to learn data items stored at a server which is not fully trusted, without disclosing to the server the particular data element retrieved. Several PIR protocols have been proposed, which provide strong guarantees on user privacy. Nevertheless, in many application scenarios it is important to protect the database as well. In this paper, we investigate the amount of data disclosed by the the most prominent PIR protocols during a single run. We show that a malicious user can stage attacks that allow an excessive amount of data to be retrieved from the server. Furthermore, this vulnerability can be exploited even if the client follows the legitimate steps of the PIR protocol, hence the malicious request can not be detected and rejected by the server. We devise mechanisms that limit the PIR disclosure to a single data item.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection*

General Terms

Design, security

Keywords

Data disclosure, private information retrieval, oblivious transfer

1. INTRODUCTION

Consider the case of a database server that stores information about diseases and their associated symptoms and publishes the associated indices for search. Alice, who is experiencing health problems, wishes to query the server in order to determine a candidate diagnosis. However, Alice wants to keep her health status private, so she does not want the server, or some malicious eavesdropper, to learn the contents of her query. Therefore, Alice should be able to find a possible diagnosis, without the server learning any of Alice's symptoms.

Private Information Retrieval (PIR) addresses such application scenarios. The PIR problem was first formulated [4] in the context of binary data, represented as a bit string $X = [x_1, \ldots, x_n]$. The client holds an index i and wishes to privately retrieve the value of bit x_i.

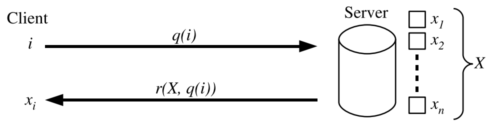

Figure 1: Overview of Computational PIR

PIR has been studied in an information-theoretic setting in which the adversary possesses unbounded computational power [4, 2], and in a computational setting where the adversary's computing capability is polynomially bounded [6, 3]. While offering stronger privacy guarantees, information-theoretic PIR protocols are computationally more expensive and need stronger assumptions, which make them less practical for real-world applications. Computational PIR (cPIR), on the other hand, is more efficient. cPIR is outlined in Fig. 1: the client, who wants to find the value of the i^{th} bit of X (i.e., x_i), sends an encrypted request $q(i)$ to the server. The server responds with a value $r(X, q(i))$, which allows the client to determine the value x_i. It is computationally intractable for an adversary to find the value of i, given $q(i)$. Furthermore, the client can easily determine the value of x_i based on the server's response $r(X, q(i))$, using a trapdoor function.

The primary goal of the PIR protocols is to protect the privacy of the client. However, the database represents an asset for the server. In traditional database applications, clients are typically billed in proportion to the amount of transferred data. Designing an effective billing mechanism is an important concern with PIR applications, since the server does not learn which particular data item has been retrieved by the client. Nevertheless, the protocol design should control the amount of data (e.g., the number of elements) retrieved during a single PIR request. Otherwise, a malicious client could abuse the system. Given practical concerns as such, in this paper, we identify two important classes of attacks that allow clients to

[*]The author is also affiliated with Institute of Software, Chinese Academy of Sciences, Beijing, China.

retrieve excessive information with a single PIR request: *redundancy attacks (RA)* and *subliminal-channel attacks (SCA)*. The former class is relevant to protocols that disclose redundant data items due to their design. The latter class of attacks is more subtle, and applies to most cPIR protocols that we are aware of.

In the rest of the paper, we analyze the unintended data disclosures during the operation of the two most prominent cPIR protocols ([6, 3]), and propose modifications that protect server privacy against both RA and SCA attacks. Specifically, we focus on securing the protocol in [6], which has been shown previously [5] to obtain reasonable performance in practice.

2. REVIEW OF EXISTING CPIR PROTOCOLS

We review the two most prominent solutions for computational PIR. Section 2.1 presents the KO protocol [6], which is based on the *Quadratic Residuosity Assumption (QRA)*, whereas Section 2.2 reviews the CMS protocol [3] which relies on the *ϕ-hiding assumption (ϕHA)*.

2.1 QRA-based PIR

Consider a large k-bit integer N which is a product of two $\frac{k}{2}$-bit primes q_1 and q_2, where k is a security parameter. Let $\mathbb{Z}_N^* = \{x \in \mathbb{Z}_N \,|\, \gcd(N, x) = 1\}$ be the set of positive numbers in $\mathbb{Z}_N$ which are relatively prime to N (gcd is the greatest common divisor). Then, the set of *quadratic residues (QR)* modulo N is defined as:

$$QR = \{y \in \mathbb{Z}_N^* | \exists x \in \mathbb{Z}_N^* : y = x^2 \pmod N\}.$$

The complement of QR in $\mathbb{Z}_N^*$ represents the set of *quadratic non-residues (QNR)*. Let $\mathbb{Z}_N^{+1} = \{y \in \mathbb{Z}_N^* | \left(\frac{y}{N}\right) = 1\}$, where $\left(\frac{y}{N}\right)$ denotes the Jacobi symbol [1]. Then, exactly half of the numbers in $\mathbb{Z}_N^{+1}$ are in QR, while the other half are in QNR.

The Quadratic Residuosity Assumption (QRA) states that it is computationally hard to determine whether a given random number is a quadratic residue (QR) or a quadratic non-residue (QNR) modulo N, provided that the factorization of N is not known.

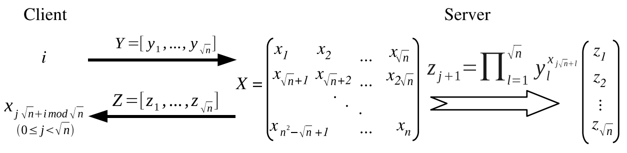
Client
Server
i
$Y=[y_1,\ldots,y_{\sqrt{n}}]$
$Z=[z_1,\ldots,z_{\sqrt{n}}]$
$x_{j\sqrt{n}+i \bmod \sqrt{n}}$
$(0 \le j < \sqrt{n})$
$X = \begin{pmatrix} x_1 & x_2 & \ldots & x_{\sqrt{n}} \\ x_{\sqrt{n}+1} & x_{\sqrt{n}+2} & \ldots & x_{2\sqrt{n}} \\ & & \ddots & \\ x_{n^2-\sqrt{n}+1} & & \ldots & x_n \end{pmatrix}$
$z_{j+1} = \prod_{l=1}^{\sqrt{n}} y_l^{x_{j\sqrt{n}+l}}$
$\begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_{\sqrt{n}} \end{pmatrix}$

Figure 2: Overview of PIR protocol from [6]

As illustrated in Fig. 2, the KO protocol organizes database X as a $t \times t$ square[1] matrix M, where $t = \lceil \sqrt{n} \rceil$, such that $M_{i,j} = x_{(i-1)\cdot t+j}$ (for ease of presentation, we assume that n is a perfect square).

When the client wishes to retrieve the data item at index i, it first generates two random $\frac{k}{2}$-bit primes q_1 and q_2 and computes the modulus $N = q_1 \cdot q_2$. Next, the client determines the row $a = \lceil i/t \rceil$ and the column $b = i \pmod t$ corresponding to x_i in matrix M, and assembles the query $q(i)$ which is a $1 \times t$ array

$$Y = [y_1, y_2, \ldots, y_t]. \tag{1}$$

Y consists of t random k-bit numbers such that $Q_N(y_b) = 1$, (i.e., y_b is in QNR), whereas all other elements $y_j, j \neq b$ are in QR. The query Y is sent to the server together with modulus N (note that, the factorization of N is kept secret).

When the server receives the query, it computes for each row r of M the product

$$z_r = \prod_{j=1}^{t} y_j^{M_{r,j}} \pmod N \tag{2}$$

and sends the resulting array

$$Z = [z_1 \; z_2 \; \ldots \; z_t] \tag{3}$$

back to the client. Note that, each z_r value is a k-bit number obtained by multiplying those y_j values for which the corresponding data bit $M_{a,b} = 1$ (i.e., a masked product). Therefore

$$M_{a,b} = \begin{cases} 1, & z_a \in \text{QNR} \\ 0, & z_a \in \text{QR} \end{cases}.$$

Knowing the factorization of N, the client can efficiently check whether or not $z_a \in$ QR, and learn the value of $M_{a,b}$. Note that, in addition to the requested item $M_{a,b}$, the client can use the remaining Z values in order to recover all elements in column b of M, i.e., a total of $\sqrt{n}$ data bits.

In [8], a modification to the KO protocol is proposed in order to reduce the amount of data disclosed to the client. However, this approach is flawed [9], and therefore the claim of the authors in [8] that redundancy attacks are completely prevented does not hold.

2.2 ϕ-hiding-based PIR

The more recent work in [3] proposes the CMS protocol which improves on the asymptotic communication cost of KO. Specifically, the $O(n^\epsilon)$ bound is reduced to $O(\log^\beta n)$ where $\beta > 1$ is a system parameter.[2] CMS relies on the *ϕ-hiding assumption (ϕHA)*, which states that given large composite prime N, it is difficult to decide (without knowing the factorization of N) whether a small prime divides $\phi(N)$, where $\phi(N)$ is the *Euler phi function* [1] at N. A prime number p is said to be *ϕ-hidden* by N if $p | \phi(N)$.

In the protocol, the client possesses a pseudo-random prime-sequence generator $\mathcal{PG}$ that computes a deterministic sequence of k-bit primes $p_1 \ldots p_n$, one for each item in the database. The client also computes an integer N that hides the prime p_i, corresponding to the item x_i that the client wishes to retrieve. As shown in [3], the probability of N hiding any other prime $p_j, j \neq i$ is negligible. The client also generates a random $g \in \mathbb{Z}_N^*$ and sends to the server g, the generator $\mathcal{PG}$ and N.

The server computes recursively the following:

$$v_0 = g, \quad v_j = v_{j-1}^{p_j^{x_j}} \pmod N), 1 \le j \le n \tag{4}$$

and returns to the client v_n. Note that each v_j is obtained by raising the previous v_{j-1} to a power equal to the prime $p_j^{x_j}$. The client receives from the server the value v_n, and determines that

$$x_i = \begin{cases} 1, & \text{if } v_n \text{ has a } p_i^{\text{th}} \text{ root modulo } N \\ 0, & \text{otherwise} \end{cases}. \tag{5}$$

3. EXCESSIVE DISCLOSURE OF DATA IN PIR PROTOCOLS

Real-world database applications of the PIR protocol should enable the server to control the number of disclosed items, even if

[1]For brevity of presentation, we only consider square matrices. Nevertheless, the KO protocol, as well as the vulnerability we will present in Section 3.1, also apply to general rectangular (i.e., non-square) matrices.

[2]According to the authors, a value of $\beta = 8$ is suitable in practice.

the server does not learn which exact items were retrieved. Ideally, only one single item (the object of the client request) should be revealed by the protocol. Next, we show how a malicious client can exploit the KO and CMS protocols to gain knowledge on a number of data items considerably larger than intended by the PIR protocols' design. In this section, we focus on SCA attacks only. The RA attack[3] for KO is discussed in detail in [9].

3.1 Exploiting KO

As mentioned in Section 2.1, the KO protocol is designed to disclose no more than $\sqrt{n}$ data items in a single request. This is already a large fraction of the database. Still, a malicious client can stage an attack that can increase the data disclosure even further, by injecting a subliminal channel within the query message.

Consider that the dishonest client rewrites the query array y (Eq. (1)) such that y_j is the j^{th} prime in the sequence of prime numbers starting with $y_1 = 2$. The server will multiply these numbers according to Eq. (2), and obtain the array z of composite numbers, one for each row of matrix M. Since the y_j values are relatively small numbers, the client can easily recover from z_i the value of $M_{i,j}$, $1 \leq j \leq t$, by factorizing z_i. Specifically,

$$M_{i,j} = \begin{cases} 1, & \text{if } y_j | z_i \\ 0, & \text{otherwise} \end{cases} .$$

EXAMPLE 1. *Suppose* $n = 16, t = 4$ *and*

$$M = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}$$

The client chooses $N = 1152921515344265237 = 1073741827 \times 1073741831$*, and* $Y = [2\ 3\ 5\ 7]$*, and sends them to the server. The server computes and returns to the client* $Z = [42\ 2\ 15\ 14]$.

By factoring z_1*, the client obtains* $42 = 2 \times 3 \times 7$*. Hence, the client learns that the first row of* M *must be* $[1, 1, 0, 1]$*. The same process is repeated for all other values* z_i. □

Since the Y values considered so far are small and in increasing order, the server may use a filter that rejects queries consisting of such y_j value sequences. However, the client can further obfuscate the server by changing the order of y_j above according to a permutation δ. The client can also *cover* each y_j value by multiplying it with a random number $\rho \in Z_N^*$. After receiving z from the server, the client divides each z_i by ρ^l, $1 \leq l \leq t$ (the client can perform division because it knows the factorization of N) until a relatively small number is obtained. By inspecting the factorization of this number, and knowing permutation δ, the client can reconstruct the values of all $M_{i,j}$.

One key observation is that although it is believed to be hard in general to factor large integers, it is easy to do so when the integer to be factored is composed of factors from a chosen factor base. In the original KO protocol [6], if a client can carefully choose the query message in such a way that all y_j are distinct small prime numbers, and the product of all y_j is noticeably smaller than the modulus N, then the entire database from the server can be retrieved by a malicious client. Certainly, in practice, t is much larger than that in the above example, and thus it is possible that the product of all the first t primes is larger than N. In this case, the client can choose the first $t' < t$ primes so that their product satisfies the desired property, assigns these t' primes to some of the y_j, and set the other y_j to be 1. In this way the client can retrieve t'/t of the database with a single query. For example, when $N \approx 2^{1024}$, t' can be chosen to be at least 100.

[3]Note that CMS is not vulnerable to RA attacks.

3.2 Exploiting CMS

As shown in Section 2.2, the client can reconstruct the value x_i of data item i by checking if the server response v_n has a root of order p_i. In the following, we show how an attacker can retrieve more than one data item with a single request.

The CMS protocol considers that a client generates an integer N that hides only a single prime p_i out of the prime sequence $p_1 \dots p_n$. However, the client could generate an N which hides a subset $p_{i_1} \cdots p_{i_\ell}$, and retrieve the values for all data items $i_1 \cdots i_\ell$. The challenging part of this attack consists of finding a value of N of suitable size which hides a larger fraction of primes p_i. Since revealing a large prime dividing $\phi(N)$ may compromise N's factorization, it is suggested in [3] that only small p_i be chosen. However, it turns out that in this case it is easy to find such an N in practice. Indeed, we have the following Algorithm 1.

Algorithm 1 Generating N that ϕ-hides $p_1, \dots, p_\ell$

```
Input: ℓ k-bit primes $p_1, \dots, p_\ell$, and a constant $f > 1$
Output: an integer $N$ that is a product of two $k^f$-bit primes and $\phi$-hides
    $p_i$ for $1 \leq i \leq \ell$.
1: Compute $q_1 \leftarrow \prod_{i=1}^{\ell_1} p_i, q_2 \leftarrow \prod_{i=\ell_1+1}^{\ell} p_i$, where $\ell_1 = \lceil \ell/2 \rceil$.
2: repeat
3:     Choose a random integer $h_1$ of approximately $(k^f - \ell_1 \cdot k)$ bits
       long, and compute a $k^f$-bit integer $r = q_1 h_1 + 1$.
4: until $r$ is prime.
5: repeat
6:     Choose a random integer $h_2$ of approximately $(k^f - (\ell - \ell_1) \cdot k)$
       bits long, and compute a $k^f$-bit integer $s = q_2 h_2 + 1$.
7: until $s$ is prime.
8: Set $N \leftarrow r \cdot s$.
```

As with the analysis in [3], under the Extended Riemann Hypothesis (cf., [1]), Algorithm 1 above generates a suitable N in expected polynomial time in $\ell \cdot k$. Given that N ϕ-hides all $p_j, 1 \leq j \leq \ell$, a malicious client can query the database server with p_j as members of the prime sequence. Since the client knows the factorization of N, it can determine the value x_j (corresponding to the prime p_j) by checking whether the server's response has a p_j^{th} root, for all $1 \leq j \leq \ell$. Without knowing N's factorization, the server is not able to tell whether N ϕ-hides any p_i, according to the ϕ-hiding assumption. Therefore it is hard for the server to distinguish a malicious query from an honest one.

4. A SECURE SYMMETRIC PIR PROTOCOL

In this section, we propose a single-database computational PIR protocol that ensures database protection by releasing one data item at one time, and is suitable for practical use. We chose to extend the QRA-based PIR technique from [6] since it is, to our knowledge, the only cPIR protocol that achieves good performance in practice [5]. Our approach consists of two steps: first, in Section 4.1 we eliminate the disclosure due to returning redundant elements to the client. Next, in Section 4.2 we show how to defend against SCA attacks that use factorization to infer excessive information.

4.1 Protecting against RA Attacks

As discussed in Section 2.1, the KO protocol returns the client the entire array $Z = [z_1, \dots, z_t]$, $t = \sqrt{n}$, where each element z_j corresponds to the product of the numbers in query array Y masked by data items in row j of matrix M. We aim to limit the disclosure to a single element z_j corresponding to the row of data item x_i

requested by the client (i.e., $j = \lfloor i/t \rfloor$). For this purpose, we make use of a 1-*out-of-n* oblivious transfer protocol proposed in [7].

Although both our proposed solution to single-database sPIR and the one suggested in [7], further referred to as NP, use a PIR and a 1-*out-of-n* protocols in their constructions, the difference is that in our solution we use KO (PIR scheme) and 1-*out-of-n* protocols in a serial way, whereas in NP a (generic) PIR scheme is used to replace one step of 1-*out-of-n*. Both schemes can achieve a comparable communication complexity $O(\sqrt{n})$. However, whereas both schemes require $O(n \log n)$ evaluation of pseudo random functions and $O(n)$ modular multiplications, our scheme needs only $O(\sqrt{n})$ public-key operations, but NP will need $O(n)$ public-key operations, if a comparable implementation is adopted. Given that one public-key operation is a lot more expensive than a single modular multiplication in general, our proposed scheme is more suitable for practical applications.

In our solution, we employ the 1-*out-of-n* protocol as an extension of KO, and we encrypt the elements of the Z array, each with an independent symmetric encryption key K_i. Since the original KO protocol sends the entire Z to the client in the first place, we do not incur any additional communication cost to that extent. We do, however, incur more communication cost during the 1-*out-of-t* protocol that retrieves the decryption key. More specifically, we provide the array $Z = [z_1, \ldots, z_t]$ as input to the 1-*out-of-t* protocol. The client needs to learn the value z_j, $j = \lfloor i/t \rfloor$, in order to reconstruct data item x_i. By only disclosing to the client the value of key K_j, we ensure that no additional data is learned by the client due to redundancy (i.e., redundancy attacks are thwarted). Denote by $\ell = \log t$ the number of bits necessary to encode any index value. A 1-*out-of-t* protocol can be constructed by using ℓ runs 1-*out-of-*2 protocol. Interested readers can find the construction details in [9].

The proposed extension of the KO protocol has two main cost components: encrypting each value in array Z, and performing ℓ runs of the 1-*out-of-*2 protocol. The computational complexity of encrypting Z is $O(t)$, whereas no additional communication overhead is incurred for transferring the encrypted Z to the client. The 1-*out-of-*2 protocol requires the server and the client each to perform two public-key operations for all bits of K_j. The communication overhead introduced by applying 1-*out-of-*2 can be optimized in real-world applications [9].

4.2 Protecting against SCA Attacks

Note that the attack on the KO protocol, described in Section 3.1, relies on the fact that the adversary receives (or recovers, if a random cover is used) a z_i which is noticeably smaller than the modulus N. However, for a client to correctly retrieve the desired bit $M_{a,b}$ in the KO protocol, only the residuosity of the server response z_a is needed. Note that, 1) multiplying the number with a QR does not change its residuosity, and 2) the server can easily generate a random QR (by squaring a random number) without needing to know the factorization of the modulus N. Therefore, to defend against the subliminal-channel attack, the server can *blind (multiply)* each response z_i with a random QR ρ_i modulo N, and return to the client $z'_i = \rho_i \cdot z_i \pmod N$, $\quad 1 \le i \le t$. All z'_i created in this way have the same residuosity as z_i, and with overwhelming probability are approximately the same size as the modulus N. This still allows an honest client to recover the desired bit by following the procedures instructed in the KO protocol, and at the same time prevents a malicious client from retrieving more bits via integer factorization. Since there are around $N/4$ QRs, each of which could be a blinding value, the malicious client cannot effectively unblind the retrieved value, hence is not able to reconstruct the subliminal channel as shown in Section 3.1 to learn additional data items.

The random blinding requires the server to generate a random square modulo N, and perform one more modular multiplication for each row of the data matrix M. The extra computational cost introduced by blinding is $O(\sqrt{n})$, negligible compared to the original KO protocol which requires $O(n)$ modular multiplications. It is clear that blinding does not increase the communication cost.

5. CONCLUSIONS

In this paper, we identified vulnerabilities common to the most prominent existing computational PIR protocols which can be exploited by an adversary in order to gain knowledge on large amounts of data stored at a database server. We also proposed extensions that protect the data stored at the server, in addition to the privacy of the client. We have also implemented our proposed protocol. Interested readers may find more details in [9].

Acknowledgements

The work reported in this paper has been partially supported by the NSF grant 0712846 "IPS: Security Services for Healthcare Applications," and the MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

6. REFERENCES

[1] E Bach and J Shallit. *Algorithmic Number Theory*. MIT Press, Cambridge, MA, USA, 1996.

[2] A Beimel, Y Ishai, E Kushilevitz, and J-F Reymond. Breaking the $O(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 261–270, 2002.

[3] C Cachin, S Micali, and M Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414. Springer, 1999.

[4] B Chor, O Goldreich, E Kushilevitz, and M Sudan. Private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[5] G Ghinita, P Kalnis, A Khoshgozaran, C Shahabi, and K-L Tan. Private queries in location based services: anonymizers are not necessary. In *Proceedings of ACM SIGMOD*, pages 121–132, New York, NY, USA, 2008. ACM.

[6] E Kushilevitz and R Ostrovsky. Replication is NOT needed: Single database, computationally-private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 364–373, 1997.

[7] M Naor and B Pinkas. Oblivious transfer and polynomial evaluation. In *STOC '99: Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 245–254, 1999.

[8] K Narayanam and C Rangan. A Novel Scheme for Single Database Symmetric Private Information Retrieval. In *Proceedings of Annual Inter Research Institute Student Seminar in Computer Science (IRISS)*, pages 803–815, 2006.

[9] N Shang, G Ghinita, Y Zhou, and E Bertino. Controlling data disclosure in computational pir protocols (extended abstract). `http://rmal.info/papers/sgzb2009.pdf`. The full version of this paper.