rPIR: Ramp Secret Sharing based Communication Efficient Private Information Retrieval

Lichun Li, Nanyang Technological University, lilichun@gmail.com
Michael Militzer, Xvid Solutions, mmilitzer@xvidsolutions.com
Anwitaman Datta, Nanyang Technological University, anwitaman@ntu.edu.sg

Even as data and analytics driven applications are becoming increasingly popular, retrieving data from shared databases poses a threat to the privacy of their users. For example, investors/patients retrieving records about interested stocks/diseases from a stock/medical database leaks sensitive information to the database server. PIR (Private Information Retrieval) is a promising security primitive to protect the privacy of users' interests. PIR allows the retrieval of a data record from a database without letting the database server know which record is being retrieved. The privacy guarantees could either be information theoretic or computational. Ever since the first PIR schemes were proposed, a lot of work has been done to reduce the communication cost in the information-theoretic settings - particularly the question communication cost, i.e., the traffic from the user to the database server. The answer communication cost (the traffic from the database server to the user) has however barely been improved. When question communication cost is much lower than the record length, reducing question communication cost has marginal benefit on lowering overall communication cost. In contrast, reducing answer cost becomes very important. In this paper we propose ramp secret sharing based mechanisms that reduce the answer communication cost in information-theoretic PIR. We have designed four information-theoretic PIR schemes, using three ramp secret sharing approaches, achieving answer communication cost close to the cost of non-private information retrieval. Evaluation shows that our PIR schemes can achieve lower communication cost and the same level of privacy compared with existing schemes. Our PIR schemes' usages are demonstrated for realistic settings of outsourced data sharing and P2P content delivery scenarios. Thus, our approach makes PIR a viable communication efficient technique to protect user interest privacy.

Additional Key Words and Phrases: Private Information Retrieval, Information-theoretic PIR, Ramp Secret Sharing

1. INTRODUCTION

Sharing of information and data forms the cornerstone of collaboration, and is vital in many walks of life. Even while retrieving data from shared databases is an indispensable way of obtaining up-to-date information, it also poses a threat to the privacy of data users. For example, investors retrieve records of interested stocks from a stock database may wish to keep the identities of retrieved records private to the database server. Encrypting records before sharing cannot provide this sort of privacy. This is not safe especially for a database having many users, e.g. a free public database or a database that anyone can pay to access to. A trivial way to protect user privacy is user downloading the whole database. Obviously, this is not communication efficient and often impractical.

[Chor1995] proposed PIR (Private Information Retrieval) to protect user interest privacy in accessing shared databases. PIR allows users to retrieve a record from a database without letting the database server know which record is being retrieved. Direct use of PIR allows to query a record by the record's index (identity). Based on PIR, advanced privacy-preserving queries like queries by keywords [Chor1997] and SQL-like queries over relational databases [Reardon2007] also can be realized. This privacy primitive has been used in diverse settings including patent databases [Asonov2001], pharmaceutical databases [Asonov2001], email systems [Sassaman2005], e-commerce [Mane2012], P2P file sharing systems [Miceli2011], etc. As an alternative to PIR, an individual user's interest may be protected using anonymizers like onion routers [Syverson1997] to relay encrypted records from the database server to the user. However, anonymizers can't protect the overall interest of all users, which itself may

contains sensitive information. It's also hard to hide user interests using anonymizers if the database is shared with only one or a few users.

There are two types of PIR: itPIR (information-theoretic PIR) and cPIR (computational PIR). The privacy in cPIR is guaranteed subject to computational bounds on the server, while itPIR provides unconditional privacy. Originally, cPIR was considered computationally impractical [Sion2007]. Subsequently, more computationally efficient cPIR schemes like [Trostle2011; Melchor2007] were invented. However, they are only practical for restricted database sizes, and their communication costs are also higher than most itPIR schemes'. In contrast, itPIR's computational cost is several orders of magnitude lower, making it relatively practical computationally [Olumofin2011].

In this paper, we focus on itPIR schemes. It was shown in [Chor1995] that, in terms of communication cost, there is no single-server itPIR scheme better than the trivial one, that of transferring the whole database to the user. It was also shown that itPIR schemes with much lower communication cost can be realized if the database is replicated to multiple servers and the servers don't collude with each other. Accordingly, several 1-private multi-server itPIR schemes were proposed, where any single server can't learn any information about the retrieved record's index. A natural generalization is t-private k-server itPIR [Beimel2005; Barkol2007], which is resistant to up to t colluding servers out of the pool of k servers in total. As shown in Fig. 1, a user uses a t-private k-server PIR to retrieve a record by following steps: the user generates questions based on the record's index; the user sends each server a question; each server returns the user an answer; the user recovers the record based on all the answers. As long as no more than t servers collude, they do not learn any non-trivial information about the record's index.

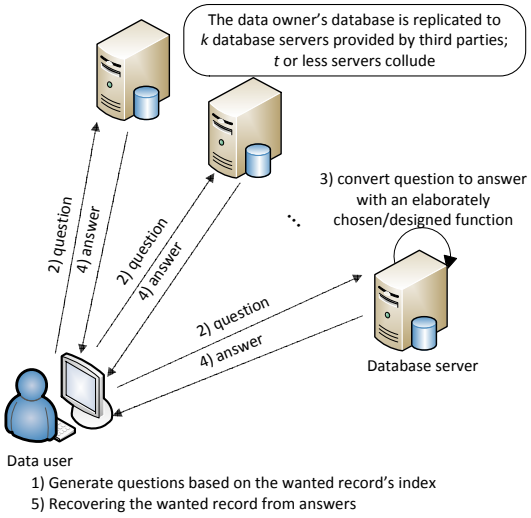

Fig. 1. t-private k-server itPIR

Problem statement. Early works [Beimel2005; Yekhanin2007] were focused on reducing itPIR's question communication cost, i.e. the traffic from the user to servers. The *Answer Cost Index* (ACI), i.e., the ratio of answer traffic amount to recovered records' size, of early k-server itPIR schemes was k, where one retrieves an l-bits record from k servers, each server's answer length being also l bits. A very recent work, [Henry2013], reduced ACI to $k/(\lfloor\sqrt{k \times t}\rfloor - t)$: to retrieve $(\lfloor\sqrt{k \times t}\rfloor - t)$ l-bits data items

(which are record segments) from k servers, one l-bit answer from each server is needed. Reducing the answer communication cost further is important for many practical settings.

- First, this is the case when question communication cost is much lower than answer communication cost.

Most itPIR schemes can make question communication cost much lower than answer communication cost with a few servers if record length is not very small. Take a database with $n = 2^{32}$ (about 4 billion) records as an example. With no more than 5 servers, quite a few itPIR schemes like [Beimel2005; Yekhanin2007] can achieve a question length of $O(n^{1/4}) = O(2^8)$ bits or less, while the answer length is at least the same as record length. Question cost is much lower than answer cost if the record length becomes significantly larger than 2^8 bits, i.e. 32 bytes. Also, by hiding the wanted record among just a large portion of the database instead of the whole database, most itPIR schemes can make question cost much lower than answer cost with a few servers. Hiding the retrieved record among a huge database is very computationally costly and even not computationally practical. As far as we know, in the experiments reported in PIR literature, the number of records is less than 1 million, and the biggest database size is 1 TB. For example, in the performance tests [Olumofin2011], the latencies of querying a 100 GB database using two itPIR schemes are both tens of seconds. Hiding the retrieved record among a large portion of a database is secure enough usually [Wang2010] and less computationally costly. The database portion can be viewed as a smaller database. Then most itPIR schemes can make question cost much lower than answer cost if retrieving records from the smaller database.

Further reducing question cost is useless for lowering the overall communication cost when question communication cost is much lower than answer communication cost. Reducing answer cost may give more incentives to use PIR based solutions widely. Additionally, such an approach may also be used as an alternative to anonymizers like onion routers in some situations. In that context, we note that PIR and anonymizers are not interchangeable for all sort of privacy needs, but in some situations they are.

When the sum of ACI and QCI (Question Cost Index: the ratio of question traffic amount to retrieved records' size) is lower than 2, PIR outperforms anonymizers in terms of communication cost. When the sum is close to 1, PIR's communication cost is in fact close to the cost of non-private information retrieval. In this paper, we show that we can push itPIR's ACI to below 2 and even close to 1 if given enough servers. (The side effects of using more servers will be discussed and evaluated in this paper.) We can also achieve a question communication cost much lower than answer communication cost, i.e. QCI much lower than ACI, at the same time.

- Second, commodity-based PIR protocols [Crescenzo2001], which use a PIR scheme as a building block, are more sensitive to answer communication cost. In these protocols, independent service providers, called commodity servers, sell off-line security commodities (messages) to database servers and users. Commodity-based PIR reduces the online question communication to $O(\log n)$ by using off-line messages from commodity servers as private inputs. Reducing ACI is essential to reduce online communication in such settings.

Our techniques and contributions. This paper uses multi-query along with ramp secret sharing to reduce itPIR's answer communication cost. Here *multi-query* means querying multiple records or segments of one or more records in one retrieval. Multi-query allows reconstructing multiple data items (records or record segments) from all servers' answers. Note that it is not necessary that an end-user queries multiple records in order to benefit from the proposed approach. Instead, each record can be divided into multiple segments, and each record segment be treated as an individual record in the context of multi-query PIR. Then we have a new "virtual" database of

new records, and the original record is retrieved by using multi-query to retrieve all its segments. In this paper, we will use the term data item, which may refer to record or record segment. The mechanism of using multi-query along with ramp secret sharing will be discussed in Section 2.1.

Our contributions are threefold:

1) We use ramp secret sharing in the design of four itPIR schemes. This new family of PIR schemes, which we call rPIR, is able to reduce ACI asymptotically close to 1 (non-private information retrieval has ACI of 1) at the price of increasing the number of servers, though thereby increasing the cost of database updates and potentially also raising the risk of privacy leakage (which we evaluate). Specifically, two multi-query itPIR schemes (we call them PIR1 & PIR2) using additive ramp secret sharing are proposed. They both use a trivial ramp secret sharing for $t = 1$, and [Kurihara2009]'s ramp secret sharing for $t > 1$. Two further schemes (PIR3 & PIR4) using [Yamamoto1986]'s ramp secret sharing are proposed. PIR1 and PIR3 can be used to retrieve multiple records or record segments in one multi-query retrieval, while PIR2 and PIR4 are usable to retrieve segments of a record only.

2) In addition to proposing four rPIR schemes to reduce answer cost, we evaluate and compare their communication cost, computational cost and their resilience against lost and wrong answers by theoretical analysis. Considering different limits on the number of servers and privacy requirements, communication cost is evaluated for both the case of static databases and that of frequently changing databases with non-negligible data update overheads across multiple servers.

3) We demonstrate rPIR's usages in two scenarios (outsourced data sharing and P2P content delivery) by theoretical analysis as well as with experiments. We show that communication cost can be reduced under realistic settings for these two scenarios. If rPIR is used to retrieve record segments instead of records, its computational cost is higher compared with traditional itPIR (i.e. itPIR not using ramp secret sharing). We also show that this tradeoff of computational cost for communication cost is worthy in these scenarios. Outsourced data sharing is a typical scenario, and the itPIR literature considers this scenario mainly. The data owner shares data via multiple outsourced storages. rPIR can be used here if data sharing applications are sensitive to communication cost. For example, data are shared to mobile users with very limited or expensive wireless bandwidth. For another example, the data owner pays a lot of money for outsourced storage's bandwidth usage. In the P2P content delivery scenario, PIR or anonymizers may be used to hide users' interests. cPIR's usage in this scenario has been studied in [Miceli2011]. [Miceli2011] proposed to reduce Freenet's[Clarke2001] traffic by replacing the anonymizer-based content relay with cPIR. However, [Miceli2011] showed that current cPIR schemes are not practical because of cPIR's high computational cost. itPIR has much lower computational cost, but its usage in P2P content delivery has never been studied. The reason probably is that traditional itPIR schemes can not reduce traffic by replacing content relay. In contrast, the proposed rPIR outperforms anonymizers in terms of communication cost, and provides a new practical alternative.

Organization. Next, in Section 2 we discuss the necessary background and related works. In Sections 3 and 4 we present our rPIR schemes using additive ramp secret sharing [Kurihara2009] and [Yamamoto1986]'s ramp secret sharing respectively. The communication cost of these novel schemes are evaluated in Section 5. In Section 6, we show rPIR's usage in outsourced data sharing and P2P content delivery with realistic settings, and benchmark rPIR's communication and computational costs. Finally, we conclude in Section 7.

2. BACKGROUND & RELATED WORK

Definitions and Notations Before presenting the related work, we provide some definitions and notations that will be used throughout the rest of this paper. If not specified otherwise, all notations will bear the meanings as given below:

ACI (Answer Cost Index): the ratio of answer traffic amount to retrieved records' size

QCI (Question Cost Index): the ratio of question traffic amount to retrieved records' size

CCI (Communication Cost Index): the ratio of traffic amount to retrieved records' size

n: number of data items or records in the database

l: data item length or record length in bits

k: number of database servers used in a retrieval

t: the maximum number of colluding servers that a PIR scheme can withstand

m: number of retrieved data items in one multi-query retrieval

e_i: the unit vector where '1' is at the ith position.

$X[i]$: the i-th element of a set X.

$X[i_1][i_2]..[i_d]$: the $(i_1, i_2, ..i_d)$-th element of a d-dimensional matrix X.

$\times$: scalar multiplication of two scalars or a scalar with a vector/matrix

$\cdot$: dot product of two vectors/matrixes

$\otimes$: tensor multiplication of two vectors/matrixes

2.1. Preliminary: Information-theoretic PIR and Secret Sharing

[Beimel2005; Beimel2012] found that secret sharing and share conversion are the key techniques used implicitly or explicitly by most itPIR schemes, and these schemes share the same basic mechanism. In this paper, we extend this mechanism considering the case of multi-query and ramp secret sharing. rPIR schemes and most existing itPIR schemes share the extended mechanism, which is given as follow. The user uses a t-private secret sharing scheme to break a secret containing the index(es) of the retrieved data item(s) (we call it input secret) into shares, and send each share to a server as a question. Each server converts the received question to an answer locally with an elaborately chosen/designed function, and sends the answer to the user. Each answer is a share of a secret containing the value(s) of the retrieved data item(s) (we call it output secret) under a t-private secret sharing scheme, which may not be the same scheme used to break up the input secret. Then the user recovers the output secret from answers.

Many itPIR schemes use perfect threshold secret sharing, while we apply ramp secret sharing instead in this work. In perfect threshold secret sharing, there is a threshold t. Any non-trivial information of the secret cannot be learned from t or less shares, while the entire secret can be learned from more than t shares. In ramp secret sharing, e.g., [Kurihara2009; Yamamoto1986], there are two thresholds t and K. Any non-trivial information of the secret cannot be learned from t or less shares; partial information can be learned from $t+1$ to $K-1$ shares; the entire secret can be learned from K or more shares. The properties of secret sharing assures the t-private property of PIR schemes.

If using perfect secret sharing, the input secret contains all information of the requested data item's index, and the output secret is the item's value. If using ramp secret sharing, the input secret contains all information of multiple requested data items' indexes, and the output secret is requested items' values. In perfect secret sharing, share length is no shorter than secret length. Ramp secret sharing was invented

Table I. Best ACIs of PIR schemes not using ramp secret sharing

PIR scheme	ACI	QCI
t-private $(t+1)$-server PIR from [Chor1995]	$t+1$	$O(ACI \times n/l)$
t-private $(t+1)^d$-server PIR from [Chor1995]	$(t+1)^d$	$O(ACI \times n^{1/d}/l)$
[Beimel2005]'s binary PIR protocols	$d \times t+1$	$O(ACI \times n^{1/d}/l)$
[Beimel2005]'s main PIR protocols (set $k = d \times t+1$)	$d \times t+1$	$O(ACI \times n^{1/d}/l)$
t-private PIR based on [Yekhanin2007] and [Barkol2007]	3^t	$ACI \times t \times n^{O(1/\log\log n)}/l$
t-private scheme based on [Efremenko2009] and [Barkol2007]	3^t $2^{r\times t}$ $(r>1)$	$ACI \times t \times n^{O(\sqrt{\log\log n/\log n})}/l$ $ACI \times t \times n^{O(\sqrt[r]{(\log n)^{1-r}(\log\log n)^{r-1}})}/l$
t-private PIR based on [Itoh2010] and [Barkol2007] $(r>1)$	$\leq (3\times 2^{r-2})^t$	$ACI \times t \times n^{O(\sqrt[r]{(\log n)^{1-r}(\log\log n)^{r-1}})}/l$
t-private scheme based on [Chee2011] and [Barkol2007] $(r>1)$	r is even: $\leq \sqrt{3}^{r\times t}$ r is odd: $\leq (8\times\sqrt{3}^{r-3})^t$	$ACI \times t \times n^{O(\sqrt[r]{(\log n)^{1-r}(\log\log n)^{r-1}})}/l$

to reduce share length. That's why we use ramp secret sharing to design PIR schemes with low answer communication cost.

2.2. itPIR Schemes with Low Answer Cost

Communication cost is one of the major barriers to PIR's use in practice. Efforts to design PIR schemes with low communication cost, excluding the recent work [Henry2013], focused on reducing question communication cost, but the lowest answer communication cost remained the same.

Henry et al's work [Henry2013] is, to our best knowledge, the only other ramp secret sharing based PIR scheme in literature. PIR3 in this paper is essentially very close to Henry et al's scheme (though designed independently), but they differ in two aspects. First, [Henry2013]'s PIR scheme uses a more complicated data item recovery method that is resistent to wrong answers returned by malicious servers. Second, they are used in different situations with different settings. [Henry2013] uses its scheme when data item length l is not smaller than data item count n, because that makes its scheme's CCI (Communication Cost Index) lowest. [Henry2013]'s setting reduces ACI and CCI to $k/(\lfloor\sqrt{k\times t}\rfloor - t)$ and $(1+n/l)\times k/(\lfloor\sqrt{k\times t}\rfloor - t)$ respectively by sacrificing robustness, i.e. reducing the number of malicious servers that it can withstand. In contrast, we designed the rPIR schemes for situations where question length can be much less than answer length. Only when $l \gg n$, PIR3's question length is much less than answer length, and we propose its use. Otherwise, we choose PIR2 and PIR4. In our settings, we reduce answer cost by increasing server count, which allows us to make ACI and CCI lower than 2 and even close to 1.

The smallest answer length in traditional itPIR schemes (i.e. schemes not using ramp secret sharing) is the same as the record length. So the best ACI is equal to the number of database servers k. Table I lists all the best traditional itPIR schemes that we could identify. To the best of our knowledge, the listed ACIs are the lowest amongst traditional PIR schemes.

To compare ACI and QCI, the schemes listed in Table I are all t-private schemes for retrieving l-bit length records. Most schemes are generalized or transformed from the 1-private schemes or from the schemes for retrieving 1-bit length records as originally given in the referenced papers. Transforming a scheme for 1-bit records to one for l-bit records is trivial. The last four schemes in Table I are transformed from the 1-private schemes in [Yekhanin2007; Efremenko2009; Itoh2010; Chee2011] using [Barkol2007]'s method, which can transform any 1-private k-server itPIR scheme to a t-private k^t-server scheme. The first two schemes in Table I are generalized from two 1-private schemes in [Chor1995]. We will summarize these two generalized schemes in Section 3 since two of our schemes are designed based on them.

2.3. Methods to Reduce Any PIR's Communication Cost

Other than designing new itPIR schemes with low communication cost, efforts have also been taken to develop methods that can be generally applied to any PIR scheme to reduce communication cost. These methods can reduce any PIR scheme's question cost and also the answer cost of some schemes. However, the best ACI of itPIR is not improved.

[Ishai2004] proposed a family of codes called batch codes to reduce PIR's communication and also computational cost. Batch code can be used only when the user retrieves multiple records. The database is encoded into multiple blocks using the batch code, whereas the block size is smaller than the database size. The user employs a PIR scheme to retrieve a data item from every block, and reconstructs multiple records from the retrieved data items. Any cPIR or itPIR scheme may be used in conjunction with batch codes. With batch codes, every block is treated as a smaller database and, since computational cost, question length and some PIR schemes' answer length are related to database size, all those complexities can be reduced. However, itPIR schemes' best answer length is the same as the data item length, which is not linked to database size. Obviously, the total length of retrieved data items can't be smaller than the total length of all records reconstructed from these items. Therefore, the ACI for retrieving multiple records using batch code is not smaller than that for retrieving a single data item from a block. Hence, the best ACI of itPIR is not improved.

Bounding-box PIR has been proposed by [Wang2010] to reduce cPIR's computational and communication costs. The same idea may be used to reduce itPIR's communication cost as well: namely, by using PIR to retrieve a desired record from a large enough portion (defined by the bounding-box) of the database instead of considering the whole database. The database server then only knows that the user retrieves a record from amongst this portion. Obviously, the portion should be large enough to meet the user's privacy requirements. We can treat this portion of the database simply as a smaller database. And because computational cost, question length and some PIR schemes' answer length are linked to database size, they can be all reduced. However, existing PIR schemes' best answer length is not dependent on database size. So again, the best ACI is not improved.

[Ishai2004]'s batch code and [Wang2010]'s idea of bounding-box can be applied to any PIR scheme including our schemes. Our rPIR schemes are most useful whenever question communication cost plays less of a role relative to answer communication cost. Still, related works like batch code or bounding-box may be used together with our rPIR schemes in order to reduce question communication cost as well as computational cost.

3. RPIR USING ADDITIVE RAMP SECRET SHARING

In this section, we present PIR1 and PIR2: two rPIR schemes using additive ramp secret sharing. They are based on two PIR schemes generalized from [Chor1995], which use a trivial additive secret sharing scheme. Different from [Chor1995]'s PIR schemes, PIR1 and PIR2 both use a trivial additive ramp secret sharing scheme for $t = 1$, and [Kurihara2009]'s additive ramp secret sharing for $t > 1$. PIR1 and PIR2 are different but related. PIR1 views a data item index as a simple value, while PIR2 views a data item index as an ordered d-tuple, which is exploited to lower QCI. (d is a configurable parameter.) PIR1 can be seen as a special case of PIR2, i.e. PIR2 with setting $d = 1$. However, PIR1 is much simpler, and its usage is different. PIR1 allows retrieving multiple records or record segments in one multi-query retrieval, while PIR2 only allows to retrieve segments of a record in one retrieval.

3.1. Additive Secret Sharing

Trivial Additive Secret Sharing A secret s is represented by a vector over F_2. Suppose the vector length is h. Then $s \in F_2^h$. The user breaks s into $t+1$ additive shares by generating t uniform random vectors $V[1], V[2], V[3], ...V[t] \in F_2^h$ and an additional vector $V[t+1] = \sum_{i=1}^{t} V[i] + s$. Then $V[1], V[2], V[3], ...V[t+1]$ are additive shares of s, and $s = \sum_{i=1}^{t+1} V[i]$. Any t shares reveal no information about the secret.

Trivial Additive Ramp Secret Sharing When t is 1, we use this secret sharing scheme in PIR1 and PIR2. In all our rPIR schemes, a secret is a set of vectors called secret blocks. Suppose a secret with u blocks is $\{s_i : i = 1, 2, ..u \quad and \quad s_i \in F_2^h\}$. As shown below, the secret can be broken into $u+1$ additive shares, and any block can be recovered by adding some shares. Any individual share reveals no information about the secret.

the way to generate shares	the way to recover secret blocks
$V[1]$: a vector chosen uniformly at random	$s_1 = V[1] + V[2]$
$V[2] = V[1] + s_1$	$s_2 = V[2] + V[3]$
$V[3] = V[1] + s_1 + s_2$	$s_3 = V[3] + V[4]$
$V[4] = V[1] + s_1 + s_2 + s_3$	...
...	...
$V[u+1] = V1 + \sum_{i=1}^{u} s_i$	$s_u = V[u] + V[u+1]$
$s_i, V[i] \in F_2^h$	

(K, L, N)-threshold Additive Ramp Secret Sharing When $t > 1$, we use [Kurihara2009]'s scheme with the setting $K = N$ and $L = K - t$ in PIR1 and PIR2. In [Kurihara2009]'s scheme, a secret has $L \times (n_p - 1)$ blocks, which are broken into N shares. Each share has $n_p - 1$ pieces. n_p is a prime no less than N. Each secret block can be recovered by adding a certain combination of share pieces from any K shares. Any $K - L$ shares reveal no information about the secret. Every share piece and secret block is a vector over Field F_2 with the same length. Due to space limitation and the complexity of [Kurihara2009]'s scheme, we omit a formal description of the algorithm from [Kurihara2009] but show an example of creating shares and recovering a secret instead as below. In the example, the secret with two blocks s_1 and s_2 is broken into 3 shares. Each share has two pieces. The i-th share's j-th piece is denoted as $sp_{i,j}$.

	piece 1	piece 2
share 1	$r_1 + r_2 + s_1$	$r_3 + r_4 + s_2$
share 2	$r_1 + r_4$	$r_3 + s_1$
share 3	$r_1 + s_2$	$r_2 + r_3$
$K = N = 3$; $L = 1$; $n_p = 3$		
$s_1, s_2, r_1, r_2, r_3, r_4 \in F_2^h$		
r_1, r_2, r_3, r_4: vectors chosen uniformly at random		
$s_1 = sp_{1,2} + sp_{2,1} + sp_{2,2} + sp_{3,1}$		
$s_2 = sp_{1,1} + sp_{2,2} + sp_{3,1} + sp_{3,2}$		

3.2. PIR1: rPIR Scheme based on t-private $(t+1)$-server Single-query PIR

Preliminary: t-private $(t+1)$-server single-query PIR

PIR1 is designed based on this t-private $(t+1)$-server single-query PIR scheme, and the latter is a trivial generalization of a 1-private 2-server scheme in [Chor1995]. Suppose the database has n data items and each item's length is l bits. There are $k = t+1$ servers indexed from 1 to k, each of which holds a replication of the database. We use a set of vectors $X[1], X[1], ...X[n]$ to represent the collection of data items in the database. The i-th data item $X[i] \in F_2^l$ can be evaluated with the following linear function over

field F_2. Here, Y_j is the j-th component of the vector Y.

$$P(Y) = \sum_{j=1}^{n} X[j] \times Y_j, \quad X_j \in F_2^l \quad and \quad Y \in F_2^n \tag{1}$$

Note that $X[i] = P(e_i)$ for any $i \in \{1, 2, ...n\}$, where e_i is a unit vector in F_2^n and only the ith component is 1. Therefore, the retrieval of $X[i]$ can be reduced to an evaluation of the function $P(Y)$ on a point e_i where the servers hold $P(Y)$ and the user holds i. This scheme uses additive secret sharing to prevent up to t colluding servers to recover the secret e_i. The user retrieves a data item X_i by following steps.

- Using the trivial additive secret sharing introduced in Section 3.1, the user breaks the secret e_i into a set of k additive shares $V[1], V[2], ...V[k]$. So $e_i = \sum_{j=1}^{k} V[j]$.

- The user sends the vector $V[j]$ to the ith server as a question for every $j \in \{1, 2, ...k\}$. The question length is n.

- The jth server uses its received vector to compute $P(V[j])$, and returns the result to the user as an answer. k servers compute and return k answers in total. Each answer's length is l.

- The user reconstructs data item $X[i]$ with the answers using the equation below, where S_i is the set of e_i's additive shares.

$$\begin{aligned} X_i = P(e_i) = P(\sum_{V[j] \in S_i} V[j]) &= \sum_{h=1}^{n} X[h] \times (\sum_{V[j] \in S_i} V[j])_h \\ &= \sum_{V[j] \in S_i} (\sum_{h=1}^{n} X[h] \times V[j]_h) = \sum_{V[j] \in S_i} P(V[j]) \end{aligned} \tag{2}$$

PIR1: rPIR scheme based on t-private $(t+1)$-server PIR

PIR1 is designed based on the above single-query scheme. More specifically, PIR1 reuses function (1) and (2). From Section 2.1, we know that question generation, question to answer conversion and data item recovery are the key steps in an itPIR scheme. PIR1 is different from the above scheme in these steps: PIR1 uses ramp secret sharing to generate questions, and the ways of using function (1) and (2) to convert questions and recover data items are different.

The same as the above scheme, any data item in the database can be evaluated with the function $P(Y)$ in (1). Different from the above scheme, this scheme is a multi-query scheme. When $t = 1$, $k-t$ data items can be retrieved using k servers in one multi-query retrieval. When $t > 1$, $(k-t) \times (n_p - 1)$ data items can be retrieved using k servers in one retrieval. Here, $k > t$, and n_p is a prime not less than k. Each data item may be a record or record segment. A user may use this scheme to get multiple records or segments of one or more records through one or more retrievals. The user can decide retrieved data item count and used server count in each retrieval considering the number of wanted records, the number of available servers and communication cost. The user may choose the segment size so that the number of retrieved data items becomes a desire value. Here, we suppose the user chooses to use k servers to get a set of data items $\{X[i] : i \in I\}$ in one retrieval, where I is a subset of $\{1, 2, ...n\}$. The number of retrieved data items, $|I|$, is set to a desired value by the user. If $t = 1$, retrieved data item count $|I|$ is set to $k - t$. Otherwise, if $t > 1$, $|I|$ is $(k - t) \times (n_p - 1)$, where n_p is a prime not less than k.

The user retrieves data items $\{X_i : i \in I\}$ as follows.

- Each element in E is viewed as a secret block, and the user breaks the secret $E = \{e_i : i \in I\}$ into k t-private additive shares using a ramp secret sharing scheme introduced in Section 3.1. If $t = 1$, trivial ramp secret sharing is used, and each secret block is equal to the sum of some shares. Otherwise, if $t > 1$, [Kurihara2009]'s (K, L, N)-threshold ramp secret sharing is used with parameter choice $L = k - t$ and $K = N = k$. If $t > 1$, each share has $n_p - 1$ share pieces, and each secret block is equal to the sum of some share pieces.

- The user sends each share to a server as a question.

- Each server uses its received share to compute an answer, and sends the answer to the user. If $t = 1$, a server uses its received share to substitute Y, and computes $P(Y)$ as the answer. If $t > 1$, a server uses $n_p - 1$ pieces of its received share to substitute Y respectively, and computes $P(Y)$'s values at $n_p - 1$ points as the answer.

- The user recovers data items $\{X[i] : i \in I\}$ from answers. If $t = 1$ (or $t > 1$), let S_i be the set of shares (or share pieces) that add up to e_i. Then the data item $X[i]$ can be recovered using equation (2).

Communication cost. Server count is k. If $t = 1$, retrieved data item count is $k - t$, question length is n, and answer length is l. If $t > 1$, the retrieved data item count is $(k - t) \times (n_p - 1)$, question length is $n \times (n_p - 1)$, and answer length is $l \times (n_p - 1)$. So no matter whether $t = 1$ or not, ACI is $k/(k - t)$ and QCI is $ACI \times n/l$. When $t = 1$, [Kurihara2009]'s ramp secret sharing also can be used. However, it requires the user to retrieve more data items compared with using the trivial ramp secret sharing.

Computational cost. In both PIR1 and [Chor1995]'s t-private $(t + 1)$-server PIR, function (1) is used to compute answers. As the computation is in the field F_2, this function can be evaluated by XORing $O(n)$ data items. The exact number of XORed items is the number of "1"s in the function's input Y, which is $n/2$ on an average. So, on an average, $(n/2 - 1) \times l \approx n \times l/2$ bit XOR operations are needed to evaluate this function, and the computational cost depends on the database size. In PIR1, when $t = 1$ (or $t > 1$), a server computes an answer by evaluating this function once (or $n_p - 1$ times). When $t = 1$, to compute an answer, the server's computational cost is the same as the cost in [Chor1995]'s PIR. No matter whether $t = 1$ or not, evaluating times per retrieved item is $k/(k - t)$, which can be reduced to lower than 2 and even close to 1. If a data item is a record, evaluating times per retrieved record of PIR1 is lower than that of [Chor1995]'s scheme, i.e. PIR1's computational cost is lower. If retrieved data items are segments of a record, PIR1's computational cost is higher. We will analyze how the computational cost affects PIR1's usage of retrieving segments in Section 6.

3.3. PIR2: rPIR Scheme based on t-private $(t + 1)^d$-server Single-query PIR

Preliminary: t-private $(t + 1)^d$-server single-query PIR

PIR2 is designed based on this t-private $(t + 1)^d$-server single-query PIR scheme, which is generalized from a 1-private 2^d-server scheme in [Chor1995]. In the generalized scheme, each data item is indexed with an ordered d-tuple $(i1, i2, i3, ...id)$, where $i1, i2, i3, ...id \in \{1, 2, ... \sqrt[d]{n}\}$. The data items in the database are represented with a d-dimensional matrix X. Each element of X is an l-bit data item, which is viewed as a vector in F_2^l. The $(i1, i2, i3, ...id)$th data item of the database is $X's$ $(i1, i2, i3, ...id)$th element $X[i1][i2][i3]...[id]$. There are $(t+1)^d$ servers. Any data item of the database can be retrieved by evaluating the multi-linear function (3) over field F_2. $\otimes$ denotes tensor product, and $\cdot$ denotes dot product in the function. Here, $Y1, Y2...Yd$ are vectors, and $Y1_{i1}, Y2_{i2}...Yd_{id}$ are the $i1$-th,$i2$-th,...id-th components of these vectors respectively. Observing that $e_{i1} \otimes e_{i2} \otimes e_{i3} \otimes ...e_{id}$ is a d-dimensional matrix that has all elements 0 except the $(i1, i2, i3, ...id)$th element is 1, then the $(i1, i2, i3, ...id)$th data item $X[i1][i2][i3]...[id]$ can be retrieved by evaluating $P(e_{i1}, e_{i2}, e_{i3}, ...e_{id})$.

$$P(Y1, Y2, ...Yd) = X \cdot (Y1 \otimes Y2 \otimes ...Yd)$$
$$= \sum_{i1=1}^{\sqrt[d]{n}} \sum_{i2=1}^{\sqrt[d]{n}} ... \sum_{id=1}^{\sqrt[d]{n}} X[i1][i2]...[id] \times Y1_{i1} \times Y2_{i2} \times ...Yd_{id}, \quad (3)$$
$$X[i1][i2]...[id] \in F_2^l \quad and \quad Y1, Y2...Yd \in F_2^{\sqrt[d]{n}}$$

The user retrieves $X[i1][i2][i3]...[id]$ in the following steps.

- Using the trivial additive secret sharing scheme introduced in Section 3.1, the user breaks every element in $\{e_{i1}, e_{i2}, e_{i3}, ...e_{id}\}$ into a set of $t+1$ additive shares, which adds up to the element. Then the user obtained d sets of vectors: $S1, S2, S3, ...Sd$.
- The user creates a set of $(t+1)^d$ ordered d-tuples based on the above sets: $S = \{(V[j1], V[j2], V[j3], ...V[jd]) : V[j1] \in S1, V[j2] \in S2, V[j3] \in S3, ...V[jd] \in Sd\}$.
- The user sends each server a different d-tuple in the set S as a question. The question length is $d \times \sqrt[d]{n}$.
- Each server computes the value of $P(Y1, Y2, ...Yd)$ using the received d-tuple as the value of $(Y1, Y2, ...Yd)$, and sends the result to the user as an answer. The answer length is l.
- The user reconstructs $X[i1][i2]...[id]$ with all answers from $(i+1)^d$ servers using equation (4). The third equality in equation (4) follows from the multilinearity of P(Y1,Y2,Y3,...Yd).

$$X[i1][i2]...[id] = P(e_{i1}, e_{i2}, ...e_{id})$$
$$= P(\sum_{V[j1] \in S1} V[j1], \sum_{V[j2] \in S2} V[j2], ... \sum_{V[jd] \in Sd} V[jd])$$
$$= \sum_{V[j1] \in S1} \sum_{V[j2] \in S2} ... \sum_{V[jd] \in Sd} P(V[j1], V[j2], ...V[jd])$$
$$V[j1], V[j2], ...V[jd] \in F_2^{\sqrt[d]{n}} \quad (4)$$

PIR2: rPIR scheme based on t-private $(t+1)^d$-server PIR

PIR2 is designed based on the above single-query scheme. More specifically, PIR2 reuses function (3) and (4). However, the key steps in PIR2 are different: ramp secret sharing is used in question generation, and the ways of using function (3) and (4) in PIR2's question answer conversion and data item recovery are different.

Same as in above single-query scheme, any data item in the database can be evaluated with function (3). In this scheme, there are $k = \prod_{i=1}^{d}(u_i + t)$ servers, where $u_i \geq 1$ and $\prod_{i=1}^{d} u_i \geq 2$. Each data item is a record segment. Suppose the user wants to retrieve the set of data items belonging to a record $\{X[i1][i2][i3]...[id] : i1 \in I1, i2 \in I2, i3 \in I3, ...id \in Id\}$, where $I1, I2, I3, ...Id$ are d subsets of $\{1, 2, ... \sqrt[d]{n}\}$. (Because of the constraints on the relation among the retrieved data items' indexes, this scheme cannot be used to retrieve data items that are arbitrarily chosen records.) Define d sets of unit vectors as: $Ei = \{e_j : j \in I_i\}, i = 1, 2, 3, ...d$. If $t = 1$, each record is split into $\prod_{i=1}^{d} u_i$ equal-length items, and $|Ei| = |Ii| = u_i$. Otherwise, if $t > 1$, each record is split into $\prod_{i=1}^{d} u_i \times (n_{pi} - 1)$ equal-length items, and $|Ei| = |Ii| = u_i \times (n_{pi} - 1)$ where n_{pi} is a prime not less than $u_i + t$.

The user retrieves the record's data items $\{X[i1][i2][i3]...[id] : i1 \in I1, i2 \in I2, i3 \in I3, ...id \in Id\}$ by the following steps.

- Using a ramp secret sharing scheme introduced in Section 3.1, the user breaks every secret Ei in $\{E1, E2, ...Ed\}$ into a set of $u_i + t$ t-private shares SHi, obtaining d

sets of shares: $SH1, SH2, ...SHd$. Here, each element in Ei is viewed as a secret block of Ei. If $t = 1$, trivial ramp secret sharing is used, and, for any $i \in \{1, 2, ...d\}$ and $e_j \in Ei$, the secret block e_j can be recovered by adding some shares in SHi. Otherwise, if $t > 1$, for any $i \in \{1, 2, ...d\}$, [Kurihara2009]'s (K, L, N)-threshold ramp secret sharing scheme is used to break Ei with the setting $K = N = u_i + t$ and $L = u_i$ and each share in SHi has $n_{pi} - 1$ pieces. For any $i \in \{1, 2, ...d\}$ and $e_j \in Ei$, the secret block e_j can be recovered by adding some share pieces of SHi's shares.

- The user creates a set of $\prod_{i=1}^{d}(u_i + t)$ ordered d-tuples based on above sets: $S = \{(V[j1], V[j2], V[j3], ...V[jd]) : V[j1] \in SH1, V[j2] \in SH2, V[j3] \in SH3, ...V[jd] \in SHd\}$

- The user sends each server a different d-tuple in the set S as a question. Then each receives d ordered shares.

- Each server computes an answer with its received d-tuple, and sends the answer to the user. If $t = 1$, each server computes the answer $P(Y1, Y2, Y3, ...Yd)$ using the received d-tuple as the value of $(Y1, Y2, Y3, ...Yd)$. If $t > 1$, each server computes the values of $P(Y1, Y2, Y3, ...Yd)$ at $\prod_{i=1}^{d}(n_{pi} - 1)$ points as the answer. The value of Yi is set to a share piece of the ith share in the received d-tuple. As the ith share has $n_{pi} - 1$ pieces, $\prod_{i=1}^{d}(n_{pi} - 1)$ values can be generated for $(Y1, Y2, Y3, ...Yd)$.

- The user reconstructs every data item in $\{X[i1][i2][i3]...[id] : i1 \in I1, i2 \in I2, ...id \in Id\}$ from the answers. If $t = 1$ (or $t > 1$), let $S1, S2, ...Sd$ be the sets of shares (or share pieces) that add up to $e_{i1}, e_{i2}, ..e_{id}$ respectively. Then any data item $X[i1][i2][i3]...[id]$ can be recovered using function (4).

Communication cost. In this rPIR scheme, server count is $\prod_{i=1}^{d}(u_i + t)$. If $t = 1$, retrieved data item count is $\prod_{i=1}^{d} u_i$, question length is $\sqrt[d]{n} \times d$, and answer length is l. If $t > 1$, retrieved data item count is $\prod_{i=1}^{d} u_i \times (n_{pi} - 1)$, question length is $\sqrt[d]{n} \times \sum_{i=1}^{d}(n_{pi} - 1)$, and answer length is $l \times \prod_{i=1}^{d}(n_{pi} - 1)$. So no matter whether t is 1 or not, ACI is $\prod_{i=1}^{d}(u_i + t)/u_i$. QCI is $ACI \times \sqrt[d]{n} \times d/l$ when $t = 1$. QCI is $ACI \times \sqrt[d]{n} \times \sum_{i=1}^{d}(n_{pi} - 1)/\prod_{i=1}^{d}(n_{pi} - 1)/l$ when $t > 1$. [Kurihara2009]'s ramp secret sharing also can be used when t is 1. Compared with using the trivial ramp secret sharing, it requires the user to request more data items, but QCI is a bit lower.

Computational cost. In both PIR2 and [Chor1995]'s t-private $(t + 1)^d$-server PIR, function (3) is used to compute answers. As the computation is in the field F_2, on average, $(n/2^d - 1) \times l \approx n \times l/2^d$ bit XOR operations are needed to evaluate this function. This function's computational cost depends on the database size. In PIR2, when $t = 1$ (or $t > 1$), a server computes an answer by evaluating this function once (or $\prod_{i=1}^{d}(n_{pi} - 1)$ times). In [Chor1995]'s t-private $(t + 1)^d$-server PIR, a server needs to evaluate this function once to compute an answer. As PIR2 uses more servers than [Chor1995]'s scheme does, the total computational cost in PIR2 is higher wether $t = 1$ or not. Therefore, PIR2 makes a tradeoff of computational cost for communication cost.

4. RPIR USING YAMAMOTO'S RAMP SECRET SHARING

In this section, we present PIR3 and PIR4: two rPIR schemes using Yamamoto's ramp secret sharing [Yamamoto1986]. The designs of PIR3 and PIR4 are inspired from PIR1 and PIR2. At first glance, PIR3 and PIR4 look similar to PIR1 and PIR2 respectively. But actually, PIR4's question generation and data item recovery are quite different from PIR2's, because PIR4 adopts [Yamamoto1986]'s ramp secret sharing in a elaborate way.

4.1. Yamamoto's Ramp Secret Sharing

Yamamoto's (K, L, N)-threshold ramp secret sharing scheme [Yamamoto1986] is based on Shamir's secret sharing [Shamir1979]. In PIR3 and PIR4, we use [Yamamoto1986]'s scheme over finite field F_{2^w} with the setting $K = N$, $L = K - t$ and $2^w > K$. [Yamamoto1986] described its scheme in the form of matrix operations. To facilitate its use in this paper, we re-present its scheme in the form of a $(u + t - 1)$-degree polynomial-like function (5). In our rPIR schemes, a secret is a set of u vectors called secret blocks. Suppose the vector length is h. Each secret vector s_i is a coefficient of the function, and the other coefficients are uniform random vectors. The variable z is a scalar. In this function, $K = u + t$ shares, $\{(f(z_i)) : z_i \neq 0, i = 1, 2, 3, ...K\}$, are generated by evaluating the function at K distinct non-zero points.

$$f(z) = \sum_{i=1}^{u} s_i \times z^{i-1} + \sum_{i=1}^{t} r_i \times z^{i+u-1}, z \in F_{2^w}; s_i, r_i \in F_{2^w}^h \tag{5}$$

The function (5) can be seen as a set of h polynomials below. In the polynomials, $s_{i,j}$ and $r_{i,j}$ are the j-th components of the vector s_i and r_i respectively. Therefore, all secret vectors can be recovered by interpolating the $(u+t-1)$-degree function with $u+t$ shares using a polynomial interpolation method. Any t shares reveal no information about the secret vectors.

$$\begin{aligned}
f_1(z) =& \sum_{i=1}^{u} s_{i,1} \times z^{i-1} + \sum_{i=1}^{t} r_{i,1} \times z^{i+u-1} \quad z, s_{i,1}, r_{i,1} \in F_{2^w} \\
f_2(z) =& \sum_{i=1}^{u} s_{i,2} \times z^{i-1} + \sum_{i=1}^{t} r_{i,2} \times z^{i+u-1} \quad z, s_{i,2}, r_{i,2} \in F_{2^w} \\
& \cdots \\
f_h(z) =& \sum_{i=1}^{u} s_{i,h} \times z^{i-1} + \sum_{i=1}^{t} r_{i,h} \times z^{i+u-1} \quad z, s_{i,n}, r_{i,h} \in F_{2^w}
\end{aligned}$$

We call a function like (5) as *vector polynomial*, which has a form very similar to a polynomial function. The only difference is that each coefficient is a vector instead of a scalar. In PIR3 and PIR4, we use one and multiple vector polynomials to generate questions respectively, and recover data items by interpolating another vector polynomial.

4.2. PIR3: a t-private $(m + t)$-server m-query rPIR scheme

Suppose the database has n data items and each data item's length is l bits. Every data item is indexed with an integer i, where $i \in \{1, 2, 3, ...n\}$. For any $i \in \{1, 2, 3, ...n\}$, the i-th data item $X[i] \in F_{2^w}^{l/w}$ can be evaluated with the following function over Field F_{2^w}. w is an integer much smaller than l and n, and $2^w - 1 \geq m + t$. We may need to pad each data item with less than w bits to use a small w and make $w|l$.

$$P(Y) = \sum_{j=1}^{n} X[j] \times Y_j, \qquad X[j] \in F_{2^w}^{l/w} \quad Y \in F_{2^w}^{n} \tag{6}$$

It's easy to notice that $X[i] = P(e_i)$ for any $i \in \{1, 2, ...n\}$. Therefore, the retrieval of $X[i]$ can be reduced to an evaluation of the function $P(Y)$ on a point e_i where the servers hold $P(Y)$ and the user holds i. The user wants to get m data items in one multi-query retrieval, and the set of their indexes is I. Each data item is a record or

a record segment. $k = m + t$ servers are used in the retrieval. The user retrieves the data items by the following steps.

- The user breaks m secret blocks $\{e_i : i \in I\}$ into $k = m+t$ t-private shares using the method introduced in Section 4.1. The following $(k-1)$-degree vector polynomial $Q(z)$ is used. The user generates k distinct shares: $\{Q(z_i) : z_i \in F_{2^w}$ and $z_i \neq 0, i = 1, 2, ...k\}$.

$$Q(z) = \sum_{i \in I} e_i \times z^{i-1} + \sum_{i=1}^{t} r_i \times z^{i+m-1} \quad z \in F_{2^w} \quad e_i, r_i \in F_{2^w}^n$$

- The user sends each server a distinct share $Q(z_i)$ as a question, and each server returns the answer $P(Q(z_i))$. As shown below, we define a vector polynomial function $R(z)$ as $P(Q(z))$. The second equality follows from the linearity of $P(Y)$.

$$R(z) = P(Q(z)) = \sum_{i \in I} P(e_i) \times z^{i-1} + \sum_{i=1}^{t} P(r_i) \times z^{i+m-1}$$

- The user recovers m data items, $\{P(e_i) : i \in I\}$, from all the $k = m + t$ answers. All answers are the values of the above $(k-1)$-degree vector polynomial, $R(z)$, at k points. The first m coefficients of $R(z)$, $\{P(e_i) : i \in I\}$, are the m secret blocks of the output secret, i.e. the values of m desired data items. Thus, the user can recover the data items by interpolating the vector polynomial with k answers.

Communication and computational costs. In PIR3, question length is $w \times n$, and answer length is l. ACI is $k/(k-t)$, and QCI is $ACI \times w \times n/l$. To compute an answer, a server needs to do $N \times l/w$ multiplications and $(N-1) \times l/w$ additions in the field F_{2^w}. An addition in this field is an XOR operation, and a multiplication can be implemented by looking up a table of pre-computed results. In practice, assuming data item size is multiple bytes, we can set $w = 8$ to support up to 255 servers, and the table size is only $2^8 \times 2^8 \times 8$ bits = 64 KB.

4.3. PIR4: a rPIR scheme with QCI of $O(ACI \times \sqrt[d]{n}/l)$

PIR4 uses a set of carefully designed vector polynomials to generate questions, and data items are recovered by interpolating a carefully designed vector polynomial. There are n data items in the database. Each data item's length is l bits. Each data item is indexed with an ordered d-tuple $(i1, i2, i3, ...id)$, where $i1, i2, i3, ...id \in \{1, 2, 3, ... \sqrt[d]{n}\}$. The data items of the database are represented with a d-dimensional matrix X. Each element of X is an l-bit data item, which is viewed as a vector in $F_{2^w}^{l/w}$. The data item indexed $(i1, i2, i3, ...id)$ is notated as $X[i1][i2][i3]...[id]$, where $i1, i2, i3, ...id \in \{1, 2, ... \sqrt[d]{n}\}$. Any data item of the database can be retrieved by evaluating the function P and $P1$ below. $\otimes$ denotes tensor product, and $\cdot$ denotes dot product in the functions. Observing that $e_{i1} \otimes e_{i2} \otimes e_{i3} \otimes ...e_{id}$ is a d-dimensional matrix that has all elements 0 except that the $(i1, i2, i3, ...id)$th element is 1, we can learn that $X[i1][i2][i3]...[id] = P(e_{i1}, e_{i2}, ...e_{id}) = P1(e_{i1} \otimes e_{i2} \otimes e_{i3} \otimes ...e_{id})$ for any $i1, i2, i3, ...id \in \{1, 2, 3, ... \sqrt[d]{n}\}$. e_i is a unit vector in $F_{2^w}^{\sqrt[d]{n}}$.

$$\begin{aligned} & P(Y1, Y2, ...Yd) = P1(Y1 \otimes Y2 \otimes ...Yd) \\ & = X \cdot (Y1 \otimes Y2 \otimes ...Yd) \\ & = \sum_{i1=1}^{\sqrt[d]{n}} \sum_{i2=1}^{\sqrt[d]{n}} ... \sum_{id=1}^{\sqrt[d]{n}} X[i1][i2]...[id] \times Y1_{i1} \times Y2_{i2} \times ...Yd_{id}, \\ & X[i1][i2]...[id] \in F_{2^w}^{l/w}, \quad Y1, Y2, ...Yd \in F_{2^w}^{\sqrt[d]{n}} \end{aligned} \tag{7}$$

This scheme can be used to retrieve a record's segments but not arbitrarily chosen records. Each record is divided into $m = \prod_{i=1}^{d} u_i$ equal-length data items. Suppose the user wants to retrieve the set of data items belonging to a record $\{X[i1][i2][i3]...[id] : i1 \in I1, i2 \in I2, i3 \in I3, ...id \in Id\}$, where $I1, I2, I3, ...Id$ are d subsets of $\{1, 2, ... \sqrt[d]{n}\}$, $|I1| = u_1 \geq 1, |I2| = u_2 \geq 1, ..., |Id| = u_d \geq 1$ and $u_1 \times u_2 \times ...u_d \geq 2$. Define d sets of unit vectors as: $Ei = \{e_j : j \in I_i\}, i = 1, 2, 3, ...d$. The user can retrieve the data items as follows.

- The user breaks every secret Ei in $\{E1, E2, E3, ...Ed\}$ into a set of t-private shares using a vector polynomial $Q_i(z)$. Then d vector polynomials $Q_1(z), Q_2(z), Q_3(z), ...Q_d(z)$ are used to break up the secrets. These vector polynomials must meet the following requirements: 1) $Q_i(z)$ has $u_i + t$ terms; 2) all members in E_i are u_i coefficients in $Q_i(z)$, and the other t coefficients of $Q_i(z)$ are vectors chosen uniformly at random from $F_{2^w}^{\sqrt[d]{n}}$; 3) every member in $\{e_{i1} \otimes e_{i2} \otimes e_{i3} \otimes ...e_{id} : i1 \in I1, i2 \in I2, i3 \in I3, ...id \in Id\}$ is a coefficient of the below function $R(z)$, which is defined as $Q_1(z) \otimes Q_2(z) \otimes Q_3(z) \otimes ...Q_d(z)$. Suppose the degree of $R(z)$ is h. Let C_i be the i-th coefficient of $R(z)$. The lower the degree of $R(z)$, the better. At the end of this section, we will show a method to create such vector polynomials that can make the degree of $R(z)$ lower than $\prod_{i=1}^{d}(u_i + t) - 1$.

$$R(z) = Q_1(z) \otimes Q_2(z) \otimes Q_3(z) \otimes ...Q_d(z) = \sum_{i=0}^{h} C_i \times z^i, \quad C_i \text{ is a } d\text{-dimensional matrix.}$$

- The user generates a set of $h + 1$ d-tuple shares: $S = \{(Q_1(z_i), Q_2(z_i), Q_3(z_i), ... Q_d(z_i)) : z_i \in F_{2^w}$ and $z_i \neq 0, i = 1, 2, ...h + 1\}$. 2^w must be bigger than $h+1$, and w must be much smaller than n and l. It's easy to choose u_i and w to meet this requirement when n and l are not too small. We may need to pad each data item with less than w bits to use a small w and make $w|l$.

- The user sends each server a different d-tuple in the set S as a question.

- Each server uses its received d-tuple $(Q_1(z_i), Q_2(z_i), Q_3(z_i), ... Q_d(z_i))$ to compute $P(Q_1(z_i), Q_2(z_i), Q_3(z_i), ...Q_d(z_i))$, and sends the result to the user as an answer. We define $S(z)$ as $P(Q_1(z), Q_2(z), Q_3(z), ...Q_d(z))$. Then, as shown below, all answers are the values of $S(z)$ at $h + 1$ points, and $S(z)$ is an h-degree vector polynomial function.

$$\begin{aligned} S(z) &= P(Q_1(z), Q_2(z), Q_3(z), ...Q_d(z)) \\ &= P1(Q_1(z) \otimes Q_2(z) \otimes Q_3(z) \otimes ...Q_d(z)) = P1(R(z)) \\ &= X \cdot R(z) = \sum_{i=0}^{h} (X \cdot C_i) \times z^i = \sum_{i=0}^{h} P1(C_i) \times z^i \end{aligned} \tag{8}$$

- The user reconstructs every data item in $\{X[i1][i2]...[id] : i1 \in I1, i2 \in I2, ...id \in Id\}$ by interpolating the vector polynomial $S(z)$ with all answers. Because $R(z)$'s coefficients include the set $\{e_{i1} \otimes e_{i2} \otimes ...e_{id} : i1 \in I1, i2 \in I2, ...id \in Id\}$, we can derive from equation (8) that $S(z)$'s coefficients include $\{P1(e_{i1} \otimes e_{i2} \otimes ...e_{id}) : i1 \in I1, i2 \in I2, ...id \in Id\}$, which are the output secret's blocks, i.e. the values of retrieved data items.

Communication and computational costs. The user can retrieve $\prod_{i=1}^{d} u_i$ data items from less than $\prod_{i=1}^{d}(u_i + t)$ servers, and every answer's length is the same as the data item length. So ACI is less than $\prod_{i=1}^{d}(u_i + t)/u_i$. The question length is $w \times d \times \sqrt[d]{n}$, and QCI is $ACI \times w \times d \times \sqrt[d]{n}/l$. To compute an answer, a server needs to do $d \times n \times l/w$ multiplications and about $n \times l/w$ additions in the field F_{2^w}. Same as PIR3, an addition in this field is an XOR operation, and a multiplication can be implemented by looking up a table of pre-computed results. In practice, we can set $w = 8$ to support up to 255 servers, and the table size is only 64KB.

When $|I1| = |I2| = ... = |Id| = 1$, this scheme also works, but turns into a single-query scheme. If using our below described method to create $Q_1(z), Q_2(z), Q_3(z), ...Q_d(z)$, the degree of $R(z)$ and $S(z)$, h, is $d \times t$. So the number of servers is $d \times t + 1$. This single-query scheme's ACI and QCI are the same as the ACI and QCI of [Beimel2005]'s binary PIR protocols listed in Table I. That makes this single-query scheme one with the lowest ACI among all single-query itPIR schemes.

Creating $Q_1(z), Q_2(z), Q_3(z), ...Q_d(z)$ for PIR4

We can create $Q1(z), Q2(z)$, $Q3(z)$, ... $Qd(z)$ in the form as shown below, and assign every term's degree $d_{i,j}$ a value using a rule described later.

$$Q_1(z) = \sum_{j=1}^{u_1}(s_{1,j} \times z^{d_{1,j}}) + \sum_{j=1}^{t}(r_{1,j} \times z^{d_{1,m+j}}), \quad s_{1,j} \in E1$$

$$Q_2(z) = \sum_{j=1}^{u_2}(s_{2,j} \times z^{d_{2,j}}) + \sum_{j=1}^{t}(r_{2,j} \times z^{d_{2,m+j}}), \quad s_{2,j} \in E2$$

$$...$$

$$Q_d(z) = \sum_{j=1}^{u_d}(s_{2,j} \times z^{d_{d,j}}) + \sum_{j=1}^{t}(r_{d,j} \times z^{d_{d,m+j}}), \quad s_{d,j} \in Ed$$

$$r_{i,j} \text{is chosen uniformly at random from } F_{2^w}^{\sqrt[d]{n}}, \qquad d_{i,j} \in F_{2^w}$$

Before describing the rule of assigning degrees, we give some definitions used in the assignment method first. We call a term as a secret term if its coefficient is a secret block or the tensor product of some secret blocks. We refer to a non-secret term as a random term. A random term's coefficient is a uniform random vector, or the tensor product of some random vector(s) and secret block(s). Suppose that not all terms in $Q_1(z), Q_2(z), ..., Q_d(z)$ have been assigned degrees. We define $\widetilde{Q}_i(z)$ as the polynomial containing all the assigned-degree terms in $Q_i(z)$ for every $i \in \{1, 2, ...d\}$, and $\widetilde{R}(z)$ as $\widetilde{Q}_1(z) \otimes \widetilde{Q}_2(z) \otimes ... \otimes \widetilde{Q}_d(z)$.

$$\begin{aligned}
Q_1(z) =& s_{1,1} \times z^0 + s_{1,2} \times z^{0+1} + s_{1,3} \times z^{1+1} + s_{1,4} \times z^{2+1} + s_{1,5} \times z^{3+1} + \\
& r_{1,1} \times z^{max(4,4)+1} + r_{1,2} \times z^{max(5,4)+1} \\
Q_2(z) =& s_{2,1} \times z^0 + s_{2,2} \times z^{6+1} + s_{2,3} \times z^{6+7+1} + r_{2,1} \times z^{max(14,4+14)+1} + \\
& r_{2,2} \times z^{max(19,4+14)+1} \\
Q_3(z) =& s_{3,1} \times z^0 + s_{3,2} \times z^{6+20+1} + s_{3,3} \times z^{6+20+27+1} + \\
& r_{3,2} \times r_{3,1} \times z^{max(54,4+14+54)+1} + z^{max(73,4+14+54)+1}
\end{aligned} \tag{9}$$

As the example in (9), we assign degrees to the terms of $Q_1(z), Q_2(z), ...Q_d(z)$ by the following steps.

- Set $d_{i,1} = 0$ for $i = 1, 2, 3, ..., d$.
- Set the values of other degrees $d_{i,j}$ one by one in the order of their indexes. The degree of a secret term is set to the current degree of $\widetilde{R}(z)$ plus 1. The degree of a random term is set to max(the current degree of $\widetilde{Q}_i(z)$, the current biggest degree of $\widetilde{R}(z)$'s secret term)+1.

We define $D_2(i)$ as the degree of $Q_i(z)$, and $D_1(i)$ as the highest degree of $Q_i(z)$'s secret term. Observing from the pattern of degrees, we find that $D_1(i)$ and $D_2(i)$ can be

computed using the following formulas. The degree of $R(z)$ (defined as $Q_1(z) \otimes Q_2(z) \otimes Q_3(z) \otimes ...Q_d(z)$ in PIR4) is $\sum_{i=1}^{d} D2(i)$, which is bigger than $\prod_{i=1}^{d} u_i$ but smaller than $\prod_{i=1}^{d}(u_i + t) - 1$. In the example shown in (9), $R(z)$'s degree is $6 + 20 + 74 = 100$, which is bigger than $\prod_{i=1}^{d} u_i = 5 \times 3 \times 3 = 45$ but smaller than $\prod_{i=1}^{d}(u_i + t) - 1 = (5 + 2) \times (3 + 2) \times (3 + 2) - 1 = 174$.

$$D1(i) = \begin{cases} u_1 - 1, & i = 1 \\ (u_i - 1) \times (1 + \sum_{k=1}^{i-1} D2(k)), & d \geq i > 1 \end{cases}$$

$$D2(i) = \begin{cases} u_1 + t - 1, & i = 1 \\ t + \sum_{k=1}^{i} D1(k), & d \geq i > 1 \end{cases}$$

5. COMMUNICATION COST EVALUATION

5.1. Communication Cost under Server Count Limit

We compare our four rPIR schemes in Table II. Assuming that the server count is limited, rPIR scheme and corresponding parameter choice should be optimized under this limit to minimize the sum of ACI and QCI. Further assuming that we desire our rPIR's communication cost to approach closely that of non-private information retrieval, then PIR1 or PIR3 should generally be used if $n/l \ll 1$, and PIR2 or PIR4 otherwise with selecting the smallest value for parameter d that fullfills $n^{1/d}/l \ll 1$. In the specific case where a rPIR scheme is used in a commodity-based PIR protocol [Crescenzo2001], QCI can be fully neglected since QCI affects off-line communication cost only. Then, PIR1 and PIR3 are always preferable since they require fewer servers than PIR2 and PIR4 to reduce ACI down to same levels.

Table II. rPIR schemes comparison

scheme	server count	data item count	ACI	QCI
PIR1	k	$(t = 1)\ k - t$ $(t > 1)\ (k - t) \times n_p$ *	$k/(k - t)$	$O(ACI \times n/l)$
PIR2	$\prod_{i=1}^{d}(u_i + t)$	$(t = 1)\ \prod_{i=1}^{d} u_i$ $(t > 1)\ \prod_{i=1}^{d} u_i \times n_{pi}$ **	$\prod_{i=1}^{d}(u_i + t)/u_i$	$O(ACI \times n^{1/d}/l)$
PIR3	k	$k - t$	$k/(k - t)$	$O(ACI \times n/l)$
PIR4	$< \prod_{i=1}^{d}(u_i + t)$	$\prod_{i=1}^{d} u_i$	$< \prod_{i=1}^{d}(u_i + t)/u_i$	$O(ACI \times n^{1/d}/l)$

* : n_p is a prime and $n_p \geq k$.

** : n_{pi} is a prime and $n_{pi} \geq u_i + t$.

It can be observed that PIR1 and PIR3 both provide the same ACI. Compared with PIR3, PIR1 is more robust since it does not require that all answers are received to recover a data item. Some data items still can be recovered even if some answers are missing or a few malicious servers return wrong answers. However, when $t > 1$, PIR1 requires the user to retrieve a larger number of data items per round of retrieval.

PIR4 is more efficient in terms of communication cost than PIR2 because PIR4 requires less servers and fewer retrieved data items to reduce ACI to same levels. However, PIR2 is more robust than PIR4. But if robustness is a requirement, PIR4 can be modified also to become even more robust than PIR2 at the expense of increasing server count and ACI a little. In such a PIR4 scheme modified for greater robustness, users would send additional questions to additional servers. Then, users will not need to receive all answers nonetheless be able to recover all the requested data items. Techniques from robust PIR schemes like [Goldberg2007; Goldberg2012] may be used to recover data items in such a modified, more robust scheme.

To summarize, we recommend using PIR1, PIR3 and PIR4 and focus on only their evaluation for the rest of this section.

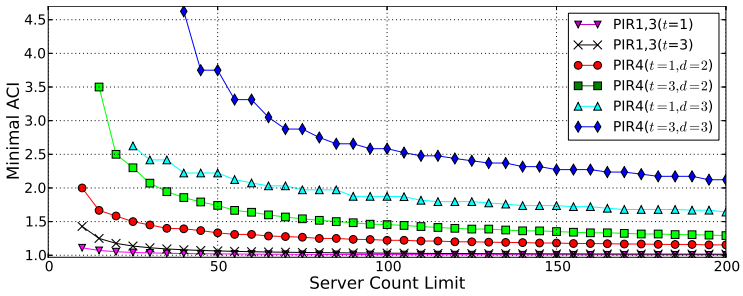

Fig. 2. Minimal ACI under different server count limits

Table III. PIR4's minimal ACI under different server count limits

t	d	server count limit	server count with minimal ACI	$u_1, ..., u_d$	minimal ACI
1	2	35	35	5, 5	1.4
1	2	40	39	4, 7	1.393
1	3	115	115	4, 4, 4	1.797
1	3	190	187	4, 4, 7	1.670

Fig. 2 depicts the minimal ACI for each of the recommended schemes at different server count limits. Compared to the best traditional itPIR schemes as listed in Table I, rPIR schemes provide several times lower ACIs. It can be further observed that ACI is reduced when the server count limit is increased. Also, ACI is smaller for lower values of t and d.

For PIR4, we determine the minimal ACI by numerical evaluation. Thereby, we compute the ACIs of all possible server counts and possible values of $u_1, u_2, ..., u_d$ under every given server count limit. Then we pick the minimal ACI for every server count limit. Some detailed results are shown in Table III. As can be observed, the server count that minimizes ACI is not always equal to the server count limit, and $u_1 = u_2 = ...u_d$ doesn't always stand.

5.2. CCI Considering Data Updating and Privacy Requirement

In case that the database doesn't change or just rarely changes and that QCI is much lower than ACI, we can put the focus on ACI in order to evaluate CCI. (Here, we also neglect the communication cost of initializing database servers assuming a server's upload traffic volume in its lifetime is much bigger than the database size.) However, in the case that the database changes often, we need to also consider the data updating cost aside of just ACI. From Table II and Fig. 2, we notice that ACI can be reduced by increasing the server count. However, with growing server count also the communication cost increases that is incurred by updating the replicated database copies among all the servers. As such, we would like to determine the optimal server count that minimizes CCI, and evaluate CCI considering ACI and data updating cost. Assuming data change rate is R_c data items/second, and the data access rate is R_a data items/second, then we can use $k \times R_c + ACI \times R_a$ to evaluate overall communication cost, and $k \times R_c/R_a + ACI$ to evaluate CCI. We show the relation between minimal CCI and server count limit in Fig. 3. Similar to Fig. 2, we determine the minimal CCI among

all the cost values of all possible server counts by numerical evaluation. From Fig. 3, it can be observed that CCI can't be reduced further if R_c/R_a is not negligible and server count limit has reached a certain level.

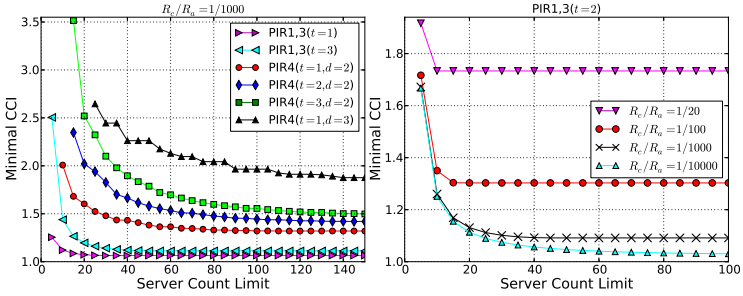

Fig. 3. Minimal CCI considering data updating and server count limit

Increasing server count may also affect privacy. Let's consider the two assumptions: 1) every server's probability of being a colluding server, p, is the same; 2) the ratio of colluding servers is no more than a threshold T. From assumption 1), the probability of privacy leakage is the probability of more than t colluding servers among all k servers, which is $\sum_{i=t+1}^{k}\binom{k}{i}p^i(1-p)^{k-i}$. Then, the probability of privacy leakage is reduced when server count k reduces or t increases. However, from Fig. 2 and Fig. 3, we see that either reducing server count limits or increasing t leads to higher ACI and CCI. Under assumption 2), the colluding server count could be as big as $round(T \times k)$. In order to protect privacy, we must ensure $t \geq round(T \times k)$, which then limits the server count. Therefore, no matter under which of the two assumptions, reducing CCI with/without data updating cost consideration is constrained by privacy requirements.

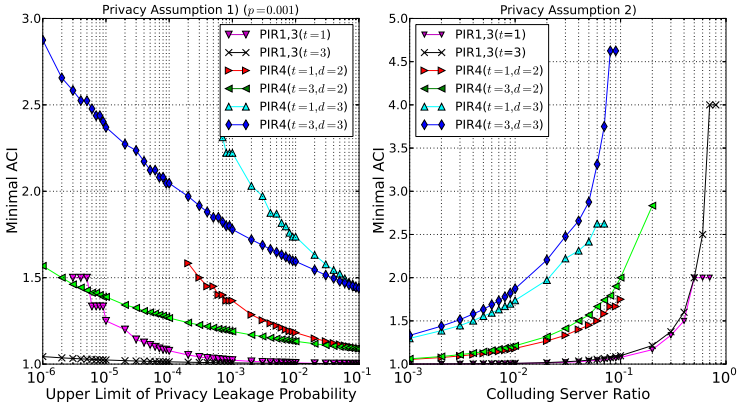

Fig. 4. Minimal ACI under privacy requirement

We depict the minimal CCI considering data updating cost under different privacy requirements in Fig. 6, and minimal ACI (i.e. CCI without considering data updating cost) in Fig. 4 and Fig. 5. In Fig. 4 and 6, we determine minimal ACI and CCI among all ACIs and CCIs for all possible server counts under constraint of different privacy requirements by numerical evaluation. For the plots with respect to privacy assumption 1), server counts are constrained by the upper limits of privacy leakage probability,

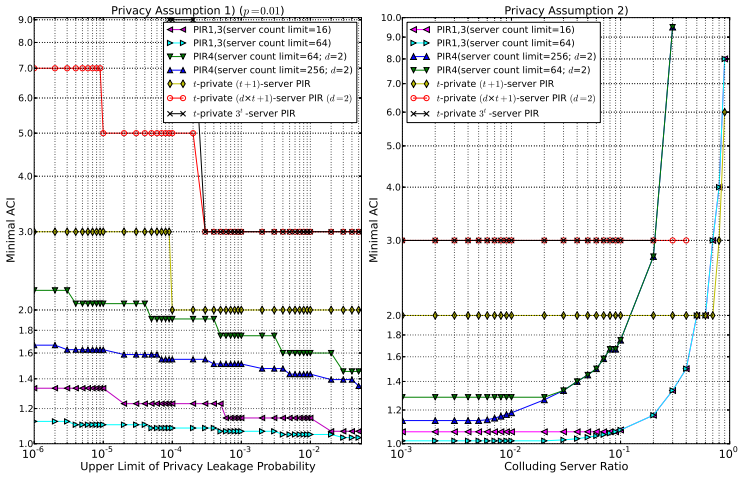

Fig. 5. Minimal ACI under privacy requirement and server count limit

meaning the maximal leakage probability that the user is willing to tolerate. For the plots with respect to assumption 2), server counts are constrained by $t \geq round(T \times k)$, which assures that no privacy leakage is possible. Fig. 5 compares our rPIR schemes and some typical schemes listed in Table 1. We vary t and limit rPIR schemes' server counts to find all schemes' minimal ACIs under privacy requirements. We can see that rPIR schemes achieve lower ACIs at most privacy levels (but at the price of higher server count), and increasing server count is more helpful to reduce ACI under privacy assumption 1).

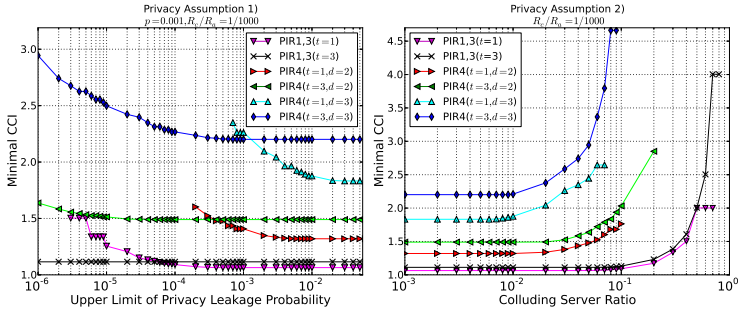

Fig. 6. Minimal CCI considering data updating and privacy requirement

Analyzing Fig. 4 and Fig. 6, we make the interesting find that the best choice of t is different under different privacy assumptions. If we consider assumption 1), PIR with bigger values for t can satisfy the requirement of reduced privacy leakage probability and lower ACI (though server count and data updating cost may be higher). But if we consider assumption 2), smaller t is better. ACI and CCI are reduced when t is set to a smaller value. Though PIR with bigger t is resistant to a bit higher colluding server ratio, ACI and CCI become very high when the colluding server ratio reaches the PIR scheme's limit. Therefore, a scenario's privacy assumption should be determined before choosing a PIR scheme and its parameters. In a scenario where every server may collude, we should apply assumption 1). In a scenario where one class of servers

can be fully trusted and another class of servers may be compromised, we should apply assumption 2).

6. RPIR APPLICATIONS AND EXPERIMENTS

Compared with traditional itPIR schemes, rPIR schemes have lower communication cost under certain settings of available server count, record length and privacy requirement. Are the cases where rPIR schemes have a lower communication cost rare? If rPIR is used to retrieve a record's segments instead of multiple records, its computational cost is higher than traditional itPIR schemes'. Is rPIR's tradeoff between communication and computational costs worthy? To answer above questions, in this section, we show rPIR's application in two scenarios, and present analysis and experiment results on communication cost as well as computational cost.

6.1. Data Sharing via Multiple Outsourced Storages

6.1.1. Privacy-preserving Outsourced Data Sharing based on itPIR.

The database is replicated to multiple outsourced storages e.g. storages provided by cloud or CDN (Content Delivery Network) providers. An outsourced storage is a "server" in PIR's model. Please note that a server means a logic entity (i.e. storage service) here, which is usually implemented with multiple machines in cloud/CDN. Suppose there is no colluding server because of the competition among storage service providers. Then a 1-private itPIR scheme can be used. Suppose hiding a retrieved record among a certain number of records is secure enough no matter how big the database is. Then, using [Wang2010]'s idea, a sufficiently large set of records containing the retrieved record is viewed as a new database, and PIR can be applied to the new database. These records are called the *cover records* of the retrieved record. In such a scenario, rPIR schemes can be used in many settings of available server count, record length and cover record count.

6.1.2. Comparing PIR1 with Traditional itPIR.

Traditional itPIR schemes' best CCI under above assumptions is about 2. This result can be achieved only if record length (in bits) is much bigger than cover record count. The result is due to [Chor1995]'s t-private $(t+1)$-server PIR scheme with the setting $t=1$. This scheme is also the most computational efficient PIR scheme. [Beimel2005]'s binary PIR protocols and main PIR protocols with setting $t=1$, $d=1$ and $k=2$ can also achieve the same CCI but at the price of higher computational cost.

In contrast, PIR1 can achieve a lower CCI at the price of using a few more servers. Let's consider an example first. Suppose using 2^{10} cover records is secure enough, and record length is 2^{18} bits (32KB). If using $k=3$ servers, each record is divided into $k-t=2$ data items. ACI is $k/(k-t)=1.50$. The item count N is 2^{11}, and item length l is 2^{17} bits. In PIR1, CCI is $1+N/l$ times of ACI. Then CCI is about 1.52. As shown in Table IV, PIR1 can also achieve lower CCI than traditional itPIR does under other settings of record length and cover record count.

As discussed in Section 3.2,a server's computational cost in PIR1 is the same as the cost in [Chor1995]'s t-private $(t+1)$-server PIR scheme when $t=1$. However, PIR1 uses more servers, and the total computational and storage costs of all servers are higher. (Please note that we discuss retrieving one record's segments in each query here. If retrieving multiple records instead, PIR1's total computational cost is lower than any existing itPIR scheme's.) rPIR makes a tradeoff of computational and storage costs for communication cost. This tradeoff is worthy if users or the data owner is more sensitive to communication cost. For example, mobile users may access data with very limited

Table IV. PIR1's CCIs under different record lengths and numbers of cover records

num\len	4 KB	16 KB	64 KB	256 KB	1 MB
2^5	1.51 1.27	1.50 1.25	1.50 1.25	1.50 1.25	1.50 1.25
2^{10}	1.69 1.88	1.55 1.41	1.51 1.29	1.50 1.26	1.50 1.25
2^{15}	$> 2, > 2$	$> 2, > 2$	$1.88, > 2$	1.59 1.56	1.52 1.33

The CCIs of using 3 and 5 servers are shown in one cell. The left CCI is the CCI for 3 servers, and the right one is for 5 servers.

and expensive wireless bandwidth. For another example, the database is popular, and the data owner pays much more money for server's bandwidth usage than for storage and computing resource usage.

Let's take the price of Amazon's cloud service[Amazon:price] to estimate the expenses in the second example. If renting a high-memory cluster as a PIR server for one year, the price per month is about $1048 (Amazon's pricing: $7220 upfront for one year; $0.62 per hour). High-memory cluster is chosen because PIR needs to read a lot of records to compute an answer and it's better to put the whole database in memory to improve performance. High-memory cluster is more expensive than most other Amazon instances, which is in favor of traditional itPIR in the comparison of price. Suppose a PIR server's average upload traffic speed is 60 MB/s when using a traditional itPIR scheme. Then the data owner should pay about $11962 per month for the traffic from the server to PIR clients (Amazon's pricing: First 1GB/month free; Up to 10 TB/month $0.120 per GB; Next 40 TB/month $0.090 per GB; Next 100 TB/month $0.070 per GB). Amazon doesn't charge for the traffic from Internet to its cloud. So only ACI matters here. If using traditional itPIR, 2 PIR servers are needed, and the data owner should pay $26020 per month for 2 servers and their traffic. (We are aware that 2 PIR servers should be provided and controlled by different cloud providers. We just use Amazon's pricing to estimate the price.) If using PIR1 and 3 servers, ACI is reduced from 2 to 1.5, and a server's average upload traffic speed is reduced to 30 MB/s. Then the data owner pays $23085 per month for 3 servers and their traffic, and saves about 11% money.

6.1.3. Using Other rPIR Schemes.

PIR3 may be used in this scenario as well, but its computational cost is higher than PIR1's. As discussed in Section 3.2, about $n \times l/2$ bit XOR operations are needed to compute an answer in PIR1 when $t = 1$. In contrast, as discussed in Section 4.2, about $n \times l/m$ XOR operations and multiplication table lookups in field F_{2^m} are needed to compute an answer in PIR3. $n \times l/m$ XOR operations in field F_{2^m} is equivalent to $n \times l$ bit XOR operations. Thus, PIR3's computational cost is higher, and using PIR1 is better.

Using PIR2 and PIR4 in this scenario can hide a small retrieved record among many records, and still achieve a low CCI. For example, a 32KB record can be hidden among 1 million records with the setting $d = 2$. To achieve a ACI of 2 and a CCI approaching to 2, PIR2 and PIR4 need 12 servers and 8 servers respectively. Compared with PIR1, using PIR4 requires a relatively big number of servers to achieve a low CCI, which limits its usage. PIR2 requires more servers, which may not be practical today. If using traditional itPIR instead, 3 servers are required, and CCI is about 3. This result is due to [Beimel2005]'s binary PIR protocols and main PIR protocols with setting $t = 1$, $d = 2$ and $k = 3$. The itPIR schemes [Yekhanin2007; Efremenko2009; Itoh2010; Chee2011] listed in Table I can also achieve this result.

6.1.4. Summary. To sum up, in the outsourced storage scenario, PIR1 is useful in many settings of record length and cover record count when $t = 1$. PIR4 is useful when need

to be hide small-sized retrieved records in many records. However, PIR4's usage is very limited because it requires a relatively big number of servers.

6.2. P2P Content Delivery

6.2.1. Privacy-preserving P2P Content Delivery based on itPIR.
Because of P2P content delivery's high scalability as well as low CAPEX and OPEX, some content service providers, e.g. PPLive (now PPTV) [PPTV] and Spotify [Spotify], utilize not only their own content storage servers but also cache peers to serve contents especially big contents to their content users. In P2P content delivery, some or all user nodes cache contents locally, and serve local contents for other users. We call this kind of user nodes as cache peers and all user nodes as peers.

PIR can hide user interests from cache peers, where contents and cache peers are viewed as records and database servers respectively in PIR's model. PIR may hide user interests from the content service provider as well, or be used in pure P2P content delivery where there is no service provider and users publish and share their own contents. However, to show rPIR's usage in these settings, complicated P2P system design and P2P algorithm must be involved. Thus, in this paper, we consider only the simpler setting that there is a content service provider and the provider is trustable. A content may be a whole file (e.g. music file, patent file) or a file chunk. For simplifying the demonstration of rPIR's usage, we consider only the case that a content is a file in this paper.

There are two cases of a trustable service provider. First, users don't mind the provider but mind other users to learn their interests. For example, users don't want other users to know what security patches they are downloading [Wu2010]. Sometimes, even some seemingly harmless interests (e.g. music interests) may leak user privacy [Chaabane2012], and need protection. Second, users don't mind anyone learns their interests, but the service provider does. If using CDN (Content Delivery Network) to delivery contents, CDN providers like Akamai usually commit to content service providers to protect user data including user interest data [Akamai:policy]. When using P2P content delivery, content service providers also have incentives to protect user interest data. A service provider may see user interest data as an important asset, which can be utilized to provide better services (e.g. content recommendation) and make money (e.g. targeted advertising). The service provider doesn't want this data leaking to competitors or other parties learning this data for free.

A PIR-based P2P content delivery system is shown in Fig. 7. Usually, the database, i.e. all contents, is too big to fit in a peer's storage. So a cache peer caches only a portion of the database, and retrieved records (i.e. contents) are hidden in the portion. Usually, cache peers cache and serve popular contents only, and unpopular contents are only stored and served by service provider's content storage servers. To use multi-server itPIR, some cache peers should hold the same portion of the database, and act as database servers in PIR's model. These peers form a group called cache group. As the service provider is trustable, it can control the number of cache groups, assign peers to cache groups, and assigns each cache group a portion of the database. These network management actions should consider factors like peer heterogeneity, content popularity, proximity between peers, etc. This is out of the scope of this paper. We only use some group setting examples to analyze PIR's performance in this paper. As shown in Fig. 7, a user performs the following steps to find and retrieve a content item. First, the user obtains the content's ID and metadata by accessing the provider's portal server, e.g. a web server introducing contents. Second, the user obtains the content's index in a cache group and the group's cache peer list from the provider's tracker server, which manages the P2P content delivery system, and maintains the mappings

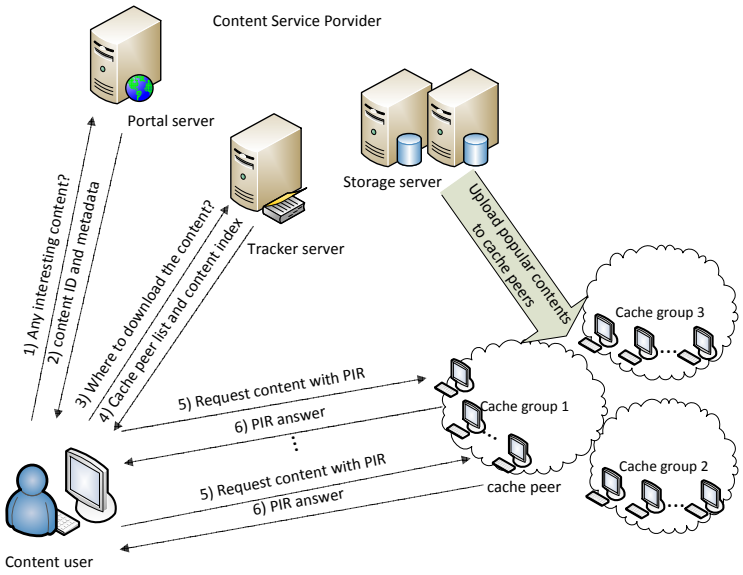

Fig. 7. Privacy-preserving P2P content delivery based on itPIR

from content IDs to information of cache groups. Third, the user retrieves the content from some or all peers in the cache group using an itPIR scheme.

6.2.2. Comparing PIR1/PIR3 with Traditional itPIR.
First, we give a cache group example, and use it throughout the comparison of PIR1/PIR3, traditional itPIR and anonymizer. Suppose each peer in the group caches the same 128 contents, and content size is 2MB. Suppose the ratio of colluding peers in the whole P2P system is 0.01. Then, in a cache group, the probability that a peer being a colluding peer is 0.01. Further, we assume a requirement of privacy leakage probability not more than 10^{-6}.

PIR1 and PIR3 can achieve a low CCI and meet the privacy requirement in such a cache group. If the group has 32 or more peers, PIR3 with setting $t = 5$ and $k = 32$ can be used. We can achieve an ACI of $k/(k-t) = 32/27 \approx 1.19$, and still meet the privacy requirement. Each content is padded with 19 bytes of zeros, and divided into $k - t = 27$ items. The item count in the group, N, is 3456, and item length, l, is 621384 bits (77673 bytes). In PIR3, CCI is $(1 + w \times l/N)$ times of ACI, and w is 8 in practice as discussed in Section 4.2. Then CCI is about 1.24. Similarly, if $t = 3$ and $k = 8$, ACI and CCI are both 1.6. Compared with PIR3, PIR1 requires to divide each content into more data items, which makes its QCI higher. As a result, the number of cache peers that PIR1 can utilize to reduce CCI is smaller. For example, PIR1 with setting $t = 3$ and $k = 8$ achieves an ACI and CCI of 1.60 and 1.63 respectively. However, if $t = 5$ and $k = 32$, ACI is reduced to 1.19, but CCI is over 2. Therefore, PIR3 can be used more widely. In Table V, we show PIR3's CCIs under some different content lengths and content counts in a cache group.

Traditional itPIR schemes may be used in above cache group too, but their communication costs are higher. To meet the privacy requirement, ACI and CCI are at least 3. The result is due to [Chor1995]'s t-private $(t+1)$-server PIR scheme with setting $t = 2$ or [Beimel2005]'s binary PIR protocols and main PIR protocols with setting $t = 2$, $d = 1$ and $k = 3$. [Chor1995]'s scheme has a lower computational cost than [Beimel2005]'s

Table V. PIR3's CCIs under different content lengths and numbers of contents in a cache group

num\len	512 KB	1 MB	2MB
32	1.60 1.33 1.24	1.60 1.33 1.21	1.60 1.33 1.20
128	1.61 1.38 1.40	1.60 1.36 1.29	1.60 1.35 1.24
512	$1.64, 1.52, > 2$	1.62 1.43 1.61	1.61 1.38 1.40
1024	$1.68, 1.71, > 2$	$1.64, 1.52, > 2$	1.62 1.43 1.61

The CCIs under three setting of t and k are shown in one cell. The left CCI is the CCI for $t = 3$ and $k = 8$; the CCI in the middle is for $t = 4$ and $k = 16$; the right one is for $t = 5$ and $k = 32$. Assuming peer collusion probability is 0.01, the privacy leakage probability is no more than 10^{-6} under these settings of t and k.

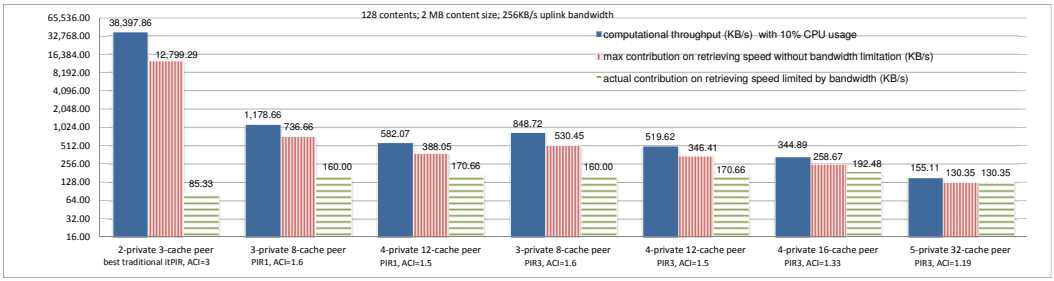

Fig. 8. a high-end cache peer's computational throughput and contribution on retrieving speed

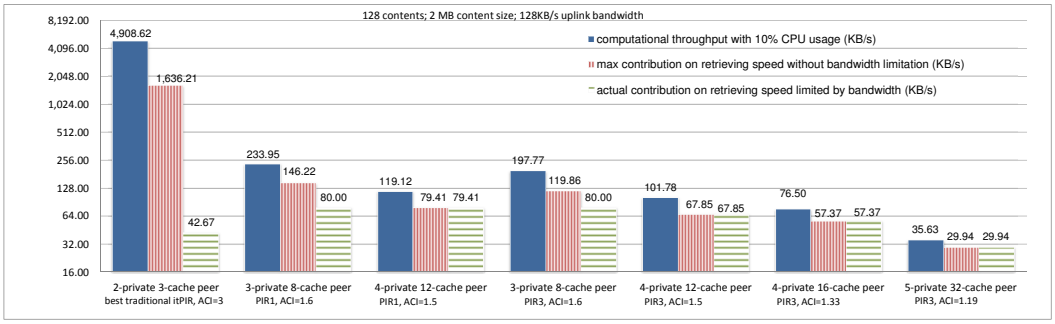

Fig. 9. a low-end cache peer's computational throughput and contribution on retrieving speed

schemes and our rPIR schemes. Compared with [Chor1995]'s scheme, rPIR makes a tradeoff of computational cost for communication cost.

To see whether above tradeoff is worthy or not, we tested the performance of PIR1, PIR3 and [Chor1995]'s t-private $(t + 1)$-server PIR scheme, which is the most efficient traditional itPIR scheme under above cache group setting. Performance was tested in high-end and low-end cache peers separately, and the results are shown in Fig. 8 and 9. A high-end cache peer has a quad-core CPU (Intel Xeon W3565) with 8MB CPU cache, and CPU frequency is 3.2 GHz. A low-end cache peer has a dual-core CPU (Intel Core2 Duo P8600) with 3 MB CPU cache, and CPU frequency is 2.4 GHz. In the tests, we constrained PIR's CPU usage in a cache peer to 10%, and measured the cache peer's computational throughput, which is the peer's speed of generating answers, i.e. (total length of generated answers)/(the time used to generate the answers). If data transfer speed is not less than computational throughput, users can retrieve contents from cache peers at the speed of (the total computational throughput of cache peers)/ACI.

We can calculate a cache peer's maximal possible contribution on the content retrieving speed as (the peer's computational throughput)/ACI.

As shown in Fig. 8 and 9, rPIR's maximal retrieving speed contribution (maximal contribution for short) and ACI drops when increasing the number of cache peers used in a retrieval. We can also see that PIR1's maximal contribution is a bit higher than PIR3's. PIR1 and PIR3 utilize uplink bandwidth more efficiently than traditional itPIR does, and their actual retrieving speed contribution (actual contribution for short) is much higher than traditional itPIR's in most cases. Though traditional itPIR's computational throughput and maximal contribution are higher, what matters is the actual contribution restricted by a upload speed limit, i.e. the cache peer's uplink bandwidth or maximal allowed upload speed (set for P2P content delivery by the user to save bandwidth for other uses). As shown in Fig. 8 and 9, if upload speed limit is lower than computational throughput, actual contribution is lower than maximal contribution. The best ACI of traditional itPIR is 3, and 2/3 uplink bandwidth used by PIR is wasted. Then traditional itPIR's actual contribution is no more than 1/3 of upload speed limit. In contrast, PIR1 and PIR3 can maximize actual contribution by adjusting the number of peers used in each retrieval, and achieve an actual contribution close to upload speed limit. In the settings of Fig. 8 and Fig. 9, when upload speed limit is lower than 3535.98 KB/s in a high-end cache peer or 438.66 KB/s in a low-end cache peer, rPIR's actual contribution is higher than traditional itPIR's. According to [testmy], the upload speeds of most countries are lower than these speeds. Also, CPU usage is limited to 10% in our experiments. Many users may accept higher cpu usages and throughputs to match higher uplink bandwidths.

6.2.3. Comparing rPIR with Anonymizer. Peers can act as anonymizers, and protect user interest in P2P content delivery. If a user wants some content cached in a cache peer, the user utilizes one or more anonymizer(s) to relay the content for it. To resist the colluding of anonymizers and the cache peer, two anonymizers are needed in a retrieval under the same privacy assumptions used above: the probability that a peer being a colluding peer is 0.01; the probability of privacy leakage must not exceed 10^{-6}. Then the CCI of a setup using anonymizer is 3, and 2/3 uplink bandwidth used by cache peers and anonymizers is wasted. Similar to traditional itPIR, users can retrieve contents from cache peers at the speed of only 1/3 of the total uplink bandwidth used by all cache peers and anonymizers. Though the computational cost of anonymizers relaying content is very low, the bottleneck is communication cost. In contrast, rPIR balances computational and communication costs, which allows the same number of peers to contribute more to content retrieving speed.

6.2.4. Using Other rPIR Schemes. PIR2 and PIR4 may be used as well. They can hide a small retrieved content among many contents, but they require many peers to achieve a low CCI and low privacy leakage probability. For example, a retrieved content of 8 KB can be hidden among 100,000 contents with PIR4's setting $d = 2$. Suppose the probability that a peer being a colluding peer is 0.01, and privacy leakage probability is not allowed to exceed 10^{-6}. A CCI of 2.67 can be achieved using 32 cache peers, and at least 95 cache peers in a cache group are required to achieve a CCI lower than 2. Because a cache peer's upload traffic should be much higher than its download traffic of the cached contents, only very popular contents should be cached in a cache group of many peers, which limits PIR4 usage. PIR2 is not recommended because it requires more peers than PIR4 does. If using traditional itPIR instead, 7 cache peers are required, and CCI is about 7. This result is due to [Beimel2005]'s binary PIR protocols and main PIR protocols with setting $t = 3$, $d = 2$ and $k = 7$.

6.2.5. Summary. To sum up, PIR1 and PIR3 can be used to hide the retrieved content among hundreds of contents, and their performance is higher than traditional itPIR's and anonymizer's. PIR4 is useful when need to be hide small-sized retrieved contents in many contents. However, PIR4's usage is limited because it requires many cache peers in a cache group to achieve a low CCI.

7. CONCLUDING REMARKS

In this paper, a new family of information theoretical PIR based on ramp secret sharing, rPIR, has been studied. Four rPIR schemes have been proposed, and three ramp secret sharing schemes are adopted in these rPIR designs. rPIR's usages has been demonstrated in outsourced data sharing and P2P content delivery scenarios. The performance of rPIR schemes have been evaluated by theoretical analysis and experiments under different numbers of available servers and privacy requirements. The results have shown that rPIR schemes can achieve several times lower communication cost and the same level of privacy compared with traditional information theoretical PIR schemes. It has also been experimentally evaluated that rPIR's tradeoff of computational cost for communication cost is worthy for some cases in outsourced data sharing scenario and most cases in P2P content delivery scenario.

REFERENCES

B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. "Private Information Retrieval." In FOCS 1995.

B. Chor , N. Gilboa , and M. Naor "Private Information Retrieval by Keywords." Technical report, TR CS0917, Department of Computer Science, Technion, 1997.

J. Reardon, J. Pound, and I. Goldberg. "Relational-Complete Private Information Retrieval." Technical report, CACR 34 (2007), University of Waterloo, 2007.

D. Asonov. "Private Information Retrieval: an Overview and Current Trends." In the ECDPvA Workshop, Informatik 2001.

L. Sassaman, B. Cohen, and N. Mathewson. "The Pynchon Gate: a Secure Method of Pseudonymous Mail Retrieval." In WPES 2005.

S. B. Mane, S. T. Sawant, and P. K. Sinha. "Using Private Information Retrieval Protocol for an E-Commerce Application." In CUBE Intl. Inf. Tech. Conf., 2012.

A. M. Miceli, B. J. Sample, C. E. Ioup, D. M. Abdelguerfi. "Private Information Retrieval in an Anonymous Peer-to-Peer Environment." In SAM 2011.

P. F. Syverson, D. M. Goldschlag, and M. G. Reed. "Anonymous Connections and Onion Routing." In S&P 1997.

A. Beimel, Y. Ishai, and E. Kushilevitz. "General Constructions for Information-Theoretic Private Information Retrieval." J. of Comp. & Sys. Sciences 71.2 (2005).

R. Sion, and B. Carbunar. "On the Computational Practicality of Private Information Retrieval." In NDSS 2007.

J. Trostle and A. Parrish. "Efficient Computationally Private Information Retrieval from Anonymity or Trapdoor Groups." In ISC 2011.

C. A. Melchor and P. Gaborit. "a Lattice-based Computationally-Efficient Private Information Retrieval Protocol." In WEWORC 2007.

F. Olumofin, I. Goldberg. "Revisiting the Computational Practicality of Private Information Retrieval." In FC 2011.

R. Henry, Y. Huang and I. Goldberg. "One (Block) Size Fits All: PIR and SPIR over Arbitrary-Length Records via Multi-block PIR Queries", In NDSS '13

I. Clarke, O. Sandberg, B. Wiley, T. W. Hong. "Freenet: a Distributed Anonymous Information Storage and Retrieval System." In Designing Privacy Enhancing Technologies, 2001.

A. Beimel, Y. Ishai, E. Kushilevitz, I. Orlov. "Share Conversion and Private Information Retrieval." In CCC 2012

G. D. Crescenzo, Y. Ishai, and R. Ostrovsky. "Universal Service-Providers for Private Information Retrieval." J. of Cryptology, 14.1 (2001).

S. Wang, D. Agrawal, and A. El Abbadi. "Generalizing PIR for Practical Private Retrieval of Public Data." In DBSec 2010.

Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. “Batch Codes and Their Applications.” In STOC 2004.

O. Barkol, Y. Ishai, and E. Weinreb. “On Locally Decodable Codes, Self-Correctable Codes, and t-private PIR.“ In APPROX 2007.

S. Yekhanin. “Towards 3-query Locally Decodable Codes of Subexponential Length.” In STOC 2007.

K. Efremenko. “3-query Locally Decodable Codes of Subexponential Length.” In STOC 2009.

T. Itoh, Y. Suzuki. “Improved Constructions for Query-Efficient Locally Decodable Codes of Subexponential Length.” IEICE Trans. on Information and Systems 93.2 (2010).

Y. M. Chee, F. Tao, S. Ling, H. Wang, and L. F. Zhang. “Query-Efficient Locally Decodable Codes of Subexponential Length.” In IEEE CCC 2011.

I. Goldberg. “Improving the Robustness of Private Information Retrieval.” In IEEE S&P 2007.

C. Devet, I. Goldberg, and N. Heninger. “Optimally Robust Private Information Retrieval.” In USENIX Security 2012.

J. Kurihara, S. Kiyomoto, K. Fukushima, and T. Tanaka. “A Fast (k, L, n)-Threshold Ramp Secret Sharing Scheme.” IEICE Trans. on Fundamentals of Electronics, Comm. and Comp. Sciences 92, (2009).

A. Shamir.“How to share a secret.” the ACM Communications 22.11 (1979).

H. Yamamoto. “Secret Sharing System using (k, L, n) Threshold Scheme.” Electronics and Comm. in Japan (Part I) (1986).

Amazon EC2 pricing, http://aws.amazon.com/ec2/pricing/

PPTV website. http://www.pptv.com/en

Spotify website. http://www.spotify.com/

D. Wu, C. Tang, P. Dhungel, N. Saxena, and K. W. Ross. “On the Privacy of Peer-Assisted Distribution of Security Patches.” In P2P 2010.

A. Chaabane, G. Acs, and M. A. Kaafar. “You are What You Like! Information Leakage through Users Interests.” In NDSS 2012.

Akamai’s privacy policy. http://www.akamai.com/html/policies/privacy_statement.html

the result of testmy.net’s upload speed test. http://testmy.net/rank/countrycode.up/