The Best of Both Worlds: Combining Information-Theoretic and Computational PIR for Communication Efficiency

Casey Devet and Ian Goldberg

University of Waterloo, ON, Canada
{cjdevet,iang}@cs.uwaterloo.ca

Abstract. The goal of *Private Information Retrieval* (PIR) is the ability to query a database successfully without the operator of the database server discovering which record(s) of the database the querier is interested in. There are two main classes of PIR protocols: those that provide privacy guarantees based on the computational limitations of servers (CPIR) and those that rely on multiple servers not colluding for privacy (IT-PIR). These two classes have different advantages and disadvantages that make them more or less attractive to designers of PIR-enabled privacy enhancing technologies.
We present a hybrid PIR protocol that combines two PIR protocols, one from each of these classes. Our protocol inherits many positive aspects of both classes and mitigates some of the negative aspects. For example, our hybrid protocol maintains partial privacy when the security assumptions of one of the component protocols is broken, mitigating the privacy loss in such an event. We have implemented our protocol as an extension of the Percy++ library so that it combines a PIR protocol by Aguilar Melchor and Gaborit with one by Goldberg. We show that our hybrid protocol uses less communication than either of these component protocols and that our scheme is particularly beneficial when the number of records in a database is large compared to the size of the records. This situation arises in applications such as TLS certificate verification, anonymous communications systems, private LDAP lookups, and others.

1 Introduction

One major goal of *privacy enhancing technologies* (PETs) is to give control over the dissemination of personal information to the users that the information pertains to. PETs rely on underlying primitives to provide a guarantee of privacy to users; these are generally primitives from fields such as cryptography and information theory. Many PETs protocols use the former, relying on assumptions about the infeasibility of solving a specific problem with a limited amount of computing resources. The advantage of the latter (information theory) approach, on the other hand, is that it provides the guarantee that no amount of computing resources will allow an adversary to discover the user's private information. However, using information-theoretic primitives instead of cryptographic ones requires some alternative assumption to support the protocol's privacy guarantees. An assumption used in many PETs, including mix networks [9], secret sharing [28], onion routing [12] and some voting protocols [5,24], is that no more than some threshold of agents are colluding against the user to discover the private information.

1.1 Private Information Retrieval

Private Information Retrieval (PIR) is a PET that allows a user to query a database for some records without letting the operator of the database server learn anything about the query or the retrieved records. The most trivial form of PIR is for the client to download the entire database from the server and do the query herself. This is private because the user has not revealed any information about which record she is interested in, yet she still retrieves the record by finding it in the content of the entire database. In a 2007 study, Sion and Carbunar concluded that no single-server PIR protocol would likely outperform this trivial download PIR protocol [29]. However, more recent work has shown that there are indeed non-trivial PIR protocols that perform better than downloading the entire database [22]. PIR has applications in many privacy-sensitive applications, including patent databases [3], domain name registration [21], anonymous email [26], anonymous communication networks [20], and electronic commerce [16].

As a PET, a PIR protocol gets its privacy guarantees from its underlying primitives. One class of PIR protocols, called *computational PIR* (CPIR) encodes the query in such a way that the database server can serve records, while learning nothing about the queries or retrieved records. These privacy guarantees are based on the assumption that some problem is hard or impossible to solve given a limit on computational power. Olumofin and Goldberg [22] showed in 2011 that it is possible for a CPIR protocol to outperform the trivial download protocol. In particular, they showed using empirical results that the CPIR protocol by Aguilar Melchor and Gaborit [2] is faster than trivial download when using typical network connections. One advantage of many CPIR protocols is the ability to use *recursion* to reduce the communication costs. This technique is illustrated by Aguilar Melchor and Gaborit with their CPIR protocol [2].

The other class of PIR protocols, called *information-theoretic PIR* (IT-PIR) does not rely on the assumption that a cryptographic primitive is hard to solve with limited computing resources. In 1995, Chor et al. showed that non-trivial IT-PIR is impossible when there is only a single database server [8]. To combat this result, they designed a multi-server IT-PIR protocol that guarantees privacy as long as not all of the servers are colluding together against the user. Several IT-PIR protocols have since been proposed [4, 13, 15, 16] that use similar non-collusion assumptions. Olumofin and Goldberg [22] also showed that a number of these multi-server IT-PIR protocols perform better than the trivial download PIR.

There are five contributing factors to the speed of a PIR query for a particular protocol:

1. the time for the client to generate a private query;
2. the communication time required to send the query to the server(s);
3. the time for the server(s) to apply the query to the database;
4. the communication time required for the response from the server to the client; and
5. the time for the client to decode the response(s).

Over time, proposed PIR protocols have incrementally improved some or all of these time factors. This paper begins by comparing two PIR protocols, one from each of these two classes. We analyze the costs of the five factors listed above for these protocols in an attempt to improve their performance.

Our main contribution is a novel *hybrid* PIR protocol that incorporates aspects of both classes, including the recursive property of single-server CPIR and the low communication and computation costs of IT-PIR. Our protocol has lower costs, while incorporating the positive properties of both classes.

Our protocol is particularly well suited for databases that consist of a large number of relatively small records. As a practical example of where PIR over databases of this shape would be beneficial, consider the problem of determining the validity of TLS web server certificates. A web client, on receiving a TLS certificate from a server, must check to see whether the certificate is revoked, typically with the Online Certificate Status Protocol (OCSP) [25], or with the recently proposed Certificate Transparency (CT) Protocol [19]. However, doing these lookups will reveal the site the client is visiting to the OCSP or CT servers. PIR has been proposed [17] as a way for clients to privately determine the validity of these certificates. Other applications of PIR over databases of this shape could include sensor network data retrieval [30], private LDAP lookups [27], and efficient retrieval of network information in anonymous communications systems [20].

1.2 Notation

For clarity, we will use the following notation throughout the paper:

- D denotes the database.
- D_i denotes the i^{th} record of the database D.
- n is the number of records in the database.
- s is the size of each record in bits.
- i_0 is the index of the database record that a client wants to retrieve.

Additional notation will be introduced in Sections 2.1, 3.1 and 4.1 to support the protocols presented in those sections.

2 Computational PIR

One class of PIR contains all protocols that assume that the server(s) are computationally bounded to make their privacy guarantees. These protocols generally show that breaking the security of their system would require an adversary to solve a problem that is believed to be hard. These types of assumptions are often used in cryptography, security and privacy; for example, the RSA public-key cryptosystem assumes that factoring large numbers is hard when an adversary has limited resources.

Computational PIR was first introduced by Chor and Gilboa in 1997 [7]. They showed that weakening the adversary to a computationally bounded entity improves the communication costs of PIR. Their work was soon followed by a protocol by Kushilevitz and Ostrovsky [18] that used the same computationally bounded adversary in their model, but did not require multiple servers as previous CPIR protocols did. This protocol relied on the assumption that the Quadratic Residuosity problem is difficult to solve.

One advantage of single-server CPIR protocols is that they can be used recursively to improve the communication cost of PIR. This idea was introduced by Kushilevitz and Ostrovsky in addition to their new single-server CPIR protocol [18]. To do this, we evenly split our database into a set of virtual records, each one containing an equal number of the actual records. The client then queries the server for a particular virtual record, but instead of returning the result to the client, the server holds on to it. The result of the first query is treated as a virtual database containing smaller virtual records. The client then queries for one of the virtual records of this virtual database. The scheme continues in this fashion until we are left with the response for a single (actual) record, which is sent to the client. This idea will be further explored in the next section.

2.1 Aguilar Melchor and Gaborit's Protocol

Without being faster than the trivial download protocol for modest-sized databases, a PIR protocol is not very useful. The main problem with the CPIR protocols already discussed is that they do not generally perform queries faster than the trivial protocol. In 2007, Aguilar Melchor and Gaborit introduced a lattice-based single-server CPIR scheme with promising results [2]; we denote this protocol as AG07. In 2011, Olumofin and Goldberg [22] empirically showed that this protocol outperforms the trivial protocol, thus suggesting that CPIR may indeed be practical.

The idea behind their protocol is to add noise to the query in a way that the server cannot discover which record the client is interested in, but with the secret information that the client has, she can remove the noise from the server's response.

Notation: For this protocol we add the following notation:

- Each record in the database is encoded as an $L \times N$ matrix of w_{AG}-bit words, where N is a security parameter and $L = \left\lceil \frac{s}{w_{AG} \cdot N} \right\rceil$.
- $q \approx 2^{2 \cdot w_{AG}}$ is the *hard noise constant*.
- $p \approx 2^{3 \cdot w_{AG}}$ is the prime modulus of the field used for arithmetic. All matrices in the protocol are over $\mathbb{Z}_p$; the entries in the above database record matrices just happen to have relatively small values ($< 2^{w_{AG}}$) in $\mathbb{Z}_p$.

Aguilar Melchor and Gaborit [2] suggest the values $w_{AG} = 20$, $N = 50$, $q = 2^{40}$, and $p = 2^{60} + 325$ for the above parameters.

Protocol: A client wants to retrieve record i_0 from the database. For each database record, she generates two matrices, one that has been made noisy and one that has not. For the query matrices corresponding to record i_0 she adds hard noise (relatively large disturbances) and for the others she adds soft noise (small disturbances). The privacy guarantees for this protocol assume that the server can not distinguish between query matrices with hard noise and soft noise. (For details, see Algorithm A.1 in Appendix A.)

When the client sends the query to the server, the amount of communication (in bits) is $6N^2 w_{AG} \cdot n$.

To process the query, each record in the database is represented as an $L \times N$ matrix whose terms are words of size w_{AG} bits. When the server receives the query, it multiplies each database record D_i by the corresponding query matrix M_i and adds the results to get R. For details, see Algorithm A.2.

The server sends the response R back to the client. The amount of communication (in bits) for this step is six times the size of each record, or $6s$.

Finally, when the client receives the response, she removes the soft noise to reveal the database record D_{i_0} that she requested (see Algorithm A.3).

The privacy of this protocol relies on the assumption that the *Hidden Lattice Problem* and the *Differential Hidden Lattice Problem* are hard to solve by computationally bounded adversaries [2]. Aguilar Melchor and Gaborit use related problems in coding theory to justify these assumptions.

Recursive AG07: As stated above, this CPIR protocol can be performed recursively to improve the communication cost of the scheme. We get optimal communication for a given recursive depth d if we split our database into $\sqrt[d]{n}$ virtual records at each iteration.

For example, if we have a database with 125 records and we are performing this recursive protocol with depth 3, in the first iteration we separate the database into $\sqrt[3]{125} = 5$ virtual records, each containing 25 actual records. This client will query the server for the virtual record that her wanted record belongs to, but instead of sending the result R_1 back to the client, the server will hold onto it. In the second iteration, we split up the result R_1 from the first iteration into 5 virtual records, each containing the encoding of 5 actual records. The client will query the server for the virtual record that contains her desired record and again the server holds onto the result R_2. Finally, for the last iteration, the server will split the result R_2 from the second iteration into 5 virtual records, each containing the encoding of one actual record. The client queries the server for the record that she is interested in and server sends the result R_3 of this last iteration to the client. The client must then perform the decoding algorithm (Algorithm A.3) 3 times, once for each iteration, to recover the database record.

In general, this improves the client-to-server communication cost to $6N^2 w_{AG} \cdot \sqrt[d]{n}$ bits. However, each iteration of the protocol increases the size of the result by a factor of 6. This makes the server-to-client communication cost $6^d s$ bits. Thus, it is important to find the appropriate recursive depth to balance out this decrease in client-to-server communication and the increase in server-to-client communication.

Advantages and Disadvantages: One advantage of this protocol is that it only requires a single server. As shown later in Section 3, multi-server protocols generally assume that some threshold of the servers are not colluding. CPIR protocols, however, remain secure even if all servers (or the one server in the single-server case) are trying to discover the client's private query.

The AG07 protocol also has the advantage that it can be used recursively, with a relatively low compounding overhead factor (6). As shown above, we can use this property to significantly improve the communication cost incurred by the protocol.

The main disadvantage of this scheme is that the security is based on lattice problems that are not well understood. Because of this, some clients may not completely trust the privacy of their queries. As stated by Aguilar Melchor et al. in a subsequent paper [1] and by Olumofin and Goldberg [22], the protocol resists known lattice-based attacks, but the protocol and its privacy assumptions are new and may not be secure.

Another disadvantage of the AG07 protocol is that it is considerably slower than many IT-PIR schemes [22]. This is due to the amount of computation involved in encoding the queries and because the server is performing a matrix-by-matrix multiplication (as compared to a vector-by-matrix multiplication used by some IT-PIR schemes).

3 Information-Theoretic PIR

The other class of PIR protocols, information-theoretic PIR (IT-PIR), includes all PIR schemes whose privacy guarantees hold no matter how computationally powerful and adversarial the server(s) may be. In 1995, Chor et al. [6] showed that any single-server IT-PIR scheme must have communication cost at least that of the trivial protocol. To avoid this problem, they developed IT-PIR protocols that used multiple servers. Since then, a variety of multiple-server IT-PIR schemes have been formulated [4, 13, 15, 16], making improvements on Chor et al.'s protocols. One of these improvements is *robustness*—the ability to retrieve the correct database records even when some of the servers are down or return incorrect or malicious responses.

An advantage to multiple-server IT-PIR is that it generally incurs smaller communication and computation costs. Like CPIR protocols, multiple-server IT-PIR protocols also need to make some assumptions to guarantee privacy; a commonly used assumption is that at most some threshold of the servers are colluding to discover the contents of a client's query.

3.1 Goldberg's Protocol

In 2007, Goldberg introduced a multiple-server IT-PIR protocol that was both efficient and provided for greater robustness than previous schemes. The idea is to use Shamir secret sharing [28] to split the client's query across multiple servers, and error-correcting codes to combine the responses. We denote this protocol by G07.

Notation: For this protocol we add the following notation:

– The database is laid out as an $n \times m$ matrix of w_G-bit words. Each record is one row of this matrix, and $m = \left\lceil \frac{s}{w_G} \right\rceil$.
– ℓ is the number of servers.
– k is the number of servers that respond to the query.
– t is the privacy level—no coalition of t or fewer servers can learn the query.
– v is the number of Byzantine servers—these are servers that may give incorrect responses.
– $\mathbb{F}$ is the field used for arithmetic ($|\mathbb{F}| \geq 2^{w_G}$). All vectors and matrices in the protocol are over $\mathbb{F}$.

Typically, $w_G = 8$, $\mathbb{F} = GF(2^8)$, and records are an integer number of bytes, so that s is a multiple of 8, and $m \cdot w_G = s$ exactly.

Protocol: To query the server for record i_0, a client creates the elementary vector $\mathbf{e_{i_0}}$ with a 1 in the i_0th place, and 0 everywhere else. She then creates ℓ Shamir secret shares $\mathbf{v_1}, \ldots, \mathbf{v}_\ell$ for $\mathbf{e_{i_0}}$ in the field $\mathbb{F}$. Algorithm B.1 details how these shares are created.

Each server is then sent one of these shares. The communication cost from the client to each server is then $n \cdot w_G$ bits.

The server simply multiplies their query vector $\mathbf{v_j}$ by the database D to get a response vector $\mathbf{r_j}$, and sends it back to the client. This makes the communication cost from each server to the client $mw_G = s$ bits.

In this protocol, we assume that some number of servers $k \leq \ell$ respond to the query. Even if $k \neq \ell$, meaning that not all servers responded, the client may still be able to recover the database record. This is because the use of Shamir secret sharing in the query makes the server responses Shamir secret shares for the database record D_{i_0}. This implies that the client only needs $k > t$ of the responses (where $t < \ell$) to successfully recover the record.

Similarly, we also do not need to assume that all of the servers are behaving correctly. The client can treat the responses as Reed-Solomon error correction codewords and use a Reed-Solomon decoding algorithm to recover the database record D_{i_0}. As shown by Devet et al. [11], the client can decode the database record in polynomial time as long as the number of Byzantine servers v is bounded by $v < k - t - 1$. They also show that this bound is the optimal bound on the number of tolerable Byzantine servers.

The Shamir secret shares are generated from a degree-t polynomial where $t < k$. By the properties of Shamir secret sharing, any coalition of at most t servers will not gain any information about the secret $\mathbf{e_{i_0}}$. However, if at least $t + 1$ of the servers collude, they will be able to discover $\mathbf{e_{i_0}}$; that is, the query is information-theoretically private assuming that at most t servers are allowed to collude. We note that there is a trade off between the level of robustness and the privacy level—the client can chose a value of t to provide the wanted privacy up to and including $t = \ell - 1$ (all but one of the servers colluding), but then there is no robustness.

Advantages and Disadvantages: As discussed above, the main advantage of the G07 protocol over other protocols is that is it robust and can handle missing and/or incorrect server responses. This allows us to combat some stronger adversarial servers that maliciously alter their responses in an attempt to block the client from recovering the database record. We note that the AG07 single-server CPIR scheme has no robustness since there is only one server and missing or incorrect responses from that server can not be overcome.

The G07 protocol has low communication cost and computation time. It is also very simple to implement on the server side. A series of works since 2011 have shown that Goldberg's protocol is faster than the trivial protocol [22] and have added improvements to the performance by using distribution of computation [10] and advanced error-correction algorithms [11].

This protocol sacrifices some level of privacy to gain robustness. Because of this we need to assume that there is no collusion between some number of servers. In some settings, it is unclear how this requirement can be enforced or detected. This uncertainty may make this protocol less desirable than others with different privacy guarantees.

Adaptation for Hybrid Security: When he introduced his IT-PIR scheme in 2007 [15], Goldberg proposed an extension to create a scheme whose privacy relied on a hybrid of information-theoretic and computational primitives. This extended scheme provides information-theoretic protection of the query as long as no more than t servers collude, but retains computational protection when any number of the servers collude.

This is accomplished by encrypting the query with an additive homomorphic cryptosystem—G07 used the Pailler cryptosystem. The client will encrypt the query before it is sent to the servers. When the servers receive the query, they multiply it by the database, but use the homomorphic property. In the case of the Pailler cryptosystem, the server would use multiplication in the place of addition and exponentiation in the place of scalar multiplication. The response that the client receives is decrypted before the regular G07 decoding operations are performed.

Though this hybrid scheme relies on two assumptions for privacy (the information-theoretic assumption that no more than t servers collude and the assumption that adversaries do not have the computational resources to break the additive homomorphic cryptosystem used), as long as one of them holds, the protocol still guarantees perfect privacy of the query.

This added protection comes at an extreme cost, however: the hybrid version of G07 is *3–4 orders of magnitude slower* [15] than the pure information-theoretic version. In the next section, we will introduce a new approach to hybrid PIR that combines the benefits of CPIR and IT-PIR without the overhead of previous proposals.

4 Hybrid PIR

In this work, we propose a hybrid protocol that combines a multiple-server IT-PIR protocol with a single-server CPIR protocol. Our goal is to incorporate the positive aspects of each protocol into our hybrid protocol, while mitigating the negative aspects of each. In particular, we want to join the low communication and computation cost of multiple-server IT-PIR schemes with the recursion of single-server CPIR schemes to improve the communication cost of PIR queries relative to both classes of protocols.

Our scheme will use a recursive depth of d as in the AG07 CPIR scheme. However, the first layer of recursion will be performed using the chosen multiple-server IT-PIR protocol. On each server, the remainder of the recursive steps will be done on the result of each previous step using the chosen recursive single-server CPIR scheme.

4.1 Notation

Our hybrid protocol will use the notation outlined in Sections 1.2, 2.1, and 3.1 as well as:

- Ψ is the multiple-server IT-PIR protocol being used.
- Φ is the single-server recursive CPIR protocol being used.

– γ_u is the number of virtual records that the database is split into for the u^{th} step of recursion of the hybrid scheme. It is required that $n \leq \prod_{u=1}^{d} \gamma_u$.
– δ_u is the number of actual records in each virtual record at the u^{th} step of recursion of the hybrid scheme. If the database does not evenly split, dummy records are appended to the end of the database to make each virtual record the same size.
– π_u is the index of the virtual record that the client's desired actual record i_0 is in at the u^{th} step of recursion.

We outline how to optimally choose the values for γ_u and δ_u in Section 5.1.

4.2 Protocol

Our protocol is generalized to use the implementer's choice of inner protocols. We use Ψ to denote the multiple-server IT-PIR inner protocol and use Φ to denote the single-server recursive CPIR inner protocol. We use this notation because our protocol is very well suited for a modular implementation. That is, an implementation of this scheme could easily swap inner protocols for other suitable protocols.

Algorithm 4.1 outlines how to generate a query for this protocol. To query the database servers, the client must determine the index π_u of the virtual record that her desired record i_0 is contained in, at each step u of the recursion. She then creates a multiple-server IT-PIR Ψ-query for index π_1 and sends each server its part of the query. Then for each remaining recursive step $u \in \{2, \ldots, d\}$, she creates single-server CPIR Φ-queries for index π_u and sends this same query to each of the servers.

Algorithm 4.2 outlines the server-side computations for this protocol. In each recursive step u, the server splits the database into γ_u virtual records, each containing δ_u actual records. For the first step, the server uses the IT-PIR Ψ server computation algorithm. For the remainder of the steps, the server uses the CPIR Φ server computation algorithm. The result of the last recursive step is sent back to the client.

We note that we can somewhat improve the performance of this scheme by starting the server-side computations for each recursive step before reading the queries for subsequent recursive steps, thus overlapping computation and communication.

When the client receives the servers' responses, she applies the corresponding decoding algorithms using the information stored during query generation in reverse order.

Algorithm 4.1 Hybrid Query Generation

```
Input: Desired record index: $i_0$
1: For each recursive step $u \in \{1, \ldots, d\}$ find the index of the virtual record $\pi_u$ that record $i_0$
   belongs to.
2: Generate a multiple-server $\Psi$-query $Q_1$ for index $\pi_1$ and send each server its part of the
   query.
3: for $u = 2 \to d$ do
4:     Generate a single-server $\Phi$-query $Q_u$ for index $\pi_u$ and send each server a copy of the
       query.
```

Algorithm 4.2 Hybrid Server Computation

```
Input: Query from client: $Q_1, \ldots, Q_d$
1: Split the database $D$ into $D^{(1)}$, a virtual database of $\gamma_1$ consecutive virtual records, each
   containing $\delta_1$ actual records.
2: Apply the $\Psi$-query $Q_1$ to database $D^{(1)}$ using the $\Psi$ server computation algorithm. The result
   is $R_1$ which will be used as the database for the next recursive step.
3: for $u = 2 \to d$ do
4:     Split the result $R_{u-1}$ into $D^{(u)}$, a virtual database of $\gamma_u$ consecutive virtual records, each
       containing the encoding of $\delta_u$ actual records
5:     Apply the $\Phi$-query $Q_u$ to database $D^{(u)}$ using the $\Phi$ server computation algorithm to get
       result $R_u$.
6: Send the final result $R_d$ to the client.
```

Algorithm 4.3 Hybrid Response Decoding

```
Input: Responses from the servers: $X_1^{(d)}, \ldots, X_k^{(d)}$
1: for $u = d \to 2$ do
2:     for $j = 1 \to k$ do
3:         Decode $X_j^{(u)}$ from server $j$ using the $\Phi$ single-server decoding algorithm to get result
           $X_j^{(u-1)}$.
4: Decode $X_1^{(1)}, \ldots, X_k^{(1)}$ simultaneously using the $\Psi$ multiple-server decoding algorithm to
   recover the database record $D_{i_0}$.
```

That is, she first uses the information from query Q_d to decode the received responses. Treating the results of that decoding as virtual responses themselves, she uses information from Q_{d-1} to decode those, and so on until she uses information from Q_1 to decode the final step. This yields the desired record. The procedure for decoding server responses for this protocol is outlined in Algorithm 4.3. We note that for all but the last step of decoding, the result from each server must be decoded separately using the single-server decoding algorithm for protocol Φ. In the last step of decoding, all server results are decoded simultaneously using the multiple-server decoding algorithm for protocol Ψ.

5 Analytical Evaluation

This evaluation of the hybrid scheme uses the G07 IT-PIR scheme for Ψ and the AG07 CPIR scheme for Φ.

5.1 Communication

The communication cost of the response from the server to the client is simply

$$C_{down} = 6^{d-1} s.$$

If we combine the communication costs for the queries at each recursive step, we get the following total cost for the query (in bits) from the client to each server:

$$C_{up} = \gamma_1 w_G + \sum_{u=2}^{d} \left(6N^2 w_{AG} \cdot \gamma_u\right).$$

To optimize C_{up}, we first find the optimal choices for the γ_u values for any given d.

After the first recursive step the result will encode $\delta_1 = \lceil \frac{n}{\gamma_1} \rceil$ records. We can optimize the CPIR query sizes by splitting the database at each remaining step into $\gamma_u = \sqrt[(d-1)]{\delta_1}$ virtual records. The cost becomes:

$$\begin{aligned} C_{up} &= \gamma_1 w_G + \sum_{u=2}^{d} \left(6N^2 w_{AG} \cdot \left(\frac{n}{\gamma_1}\right)^{\frac{1}{d-1}}\right) \\ &= \gamma_1 w_G + (d-1) \cdot 6N^2 w_{AG} \cdot \left(\frac{n}{\gamma_1}\right)^{\frac{1}{d-1}} \end{aligned}.$$

We then find the value of γ_1 that minimizes C_{up} to be:

$$\gamma_1 = \left(\frac{6N^2 w_{AG}}{w_G} \sqrt[(d-1)]{n}\right)^{\frac{d-1}{d}}.$$

Therefore, at recursive step u, we split the database as follows:

$$\gamma_u = \begin{cases} \left(\frac{6N^2 w_{AG}}{w_G} \sqrt[(d-1)]{n}\right)^{\frac{d-1}{d}} & : u = 1 \\ \left(\frac{n}{\gamma_1}\right)^{\frac{1}{d-1}} & : 2 \le u \le d \end{cases}$$

$$\delta_u = \left(\frac{n}{\gamma_1}\right)^{\frac{d-u}{d-1}}$$

With these values, our query communication cost simplifies to:

$$C_{up} = d\left(6N^2 w_{AG}\right)^{\frac{d-1}{d}} \sqrt[d]{w_G}\, \sqrt[d]{n}.$$

We observe that both the query and response cost functions (C_{up} and C_{down}) are concave up in d. Therefore, the combined communication cost can be minimized for some depth d. Since d is an integer, we evaluate the cost functions at each d starting at 1 and incrementing until we find a value for d such that the cost at d is less than the cost at $d + 1$. This value of d is our optimal depth.

Note that the combined cost function should ideally, if such information is available, take the bandwidths of both directions of our connection into account and that the different directions may have different bandwidths. This is accomplished with a simple linear weighting, such as $4C_{down} + C_{up}$ if the downstream bandwidth is 4 times that of the upstream.

Table 1 shows a comparison of the communication cost for each of the protocols in this paper.

Table 1. A comparison of the communication costs (in bits) for the PIR protocols discussed in this paper.

Protocol	Query Cost	Response Cost
AG07 [2]	$\left(6N^2 w_{AG}\right) n$	$6s$
Recursive AG07 [2]	$d\left(6N^2 w_{AG}\right) \sqrt[d]{n}$	$6^d s$
G07 [15]	$\ell w_G n$	ℓs
Our hybrid (with AG07 and G07)	$\ell d\left(6N^2 w_{AG}\right)^{\frac{d-1}{d}} \sqrt[d]{w_G}\,\sqrt[d]{n}$	$\ell 6^{d-1} s$

If we use our hybrid protocol with a depth of $d = 1$, then we are simply using the G07 protocol (with no CPIR component) and so will clearly have the same amount of communication as the G07 protocol. Since we choose the value of d that minimizes the communication cost for our hybrid protocol, we only use $d > 1$ if doing so results in a lower communication cost. Hence when we use a depth of $d > 1$, we will have a lower communication cost than the G07 scheme. Therefore, our hybrid scheme will not have a higher communication cost than G07 for any depth. For typical values of the parameters, we find that for 1 KB records, we will select $d > 1$ (and so strictly outperform G07) whenever $n > 160,000$. For 10 KB records, we see an improvement for $n > 240,000$.

Comparing the formulas in Table 1, we see that the upstream cost of our hybrid protocol is no worse than that of Recursive AG07 when $\ell^d \leq \frac{6N^2 w_{AG}}{w_G}$ ($= 37500$ for the recommended parameters), and similarly for the downstream cost when $\ell \leq 6$. For many reasonable PIR setups, these inequalities are easily satisfied. Even if they are not, however, the computational savings of our scheme over Recursive AG07 (see below) more than makes up for the difference. A slight complication in the analysis arises in cases in which the optimal recursive depth d differs between the Recursive AG07 scheme and our hybrid scheme; however, we will see in Section 6 that our scheme nonetheless outperforms the Recursive AG07 scheme.

5.2 Computation

Unlike our analysis of communication, we do not have simple expressions for our computation costs. In this section we reason about the computational cost of our protocol compared to others; in Section 6.2, below, we directly measure the computation costs of our scheme using empirical experimentation. The key observation, however, is that the slower CPIR protocol is being performed over a *much smaller database* than the original. The protocol effectively consists of IT-PIR over the whole database of n records, followed by recursive CPIR over a sub-database of δ_1 records.

Query Encoding: AG07 is expensive when generating the query because it involves matrix multiplications. However, G07 is relatively cheap because it is essentially just generating random values and evaluating polynomials. We expect the hybrid scheme will be better than AG07 for this step because it replaces one iteration with the cheap

G07 scheme encoding. Our hybrid scheme may also be faster in this stage than G07 because of the addition of recursion. As when recursion is added to the AG07 protocol, we change the request from one large (size n) query into d much smaller (size a constant multiple of $\sqrt[d]{n}$) queries.

Server Computation: The AG07 scheme is also expensive compared to the G07 scheme for server-side computation. This is because AG07 uses matrix-by-matrix multiplication for the bulk of its work, whereas G07 uses vector-by-matrix multiplication. Our hybrid scheme will use the relatively cheap server computation of G07 for the first iteration where the database is its full size. The subsequent iterations will use a much smaller subset of the database, so using AG07's server computation will not add much additional expense.

Response Decoding: The last recursive step of decoding for our hybrid scheme will take the same amount of computation as the G07 scheme. Since we have $d-1$ steps of AG07 decoding as well, our hybrid protocol will not outperform G07 in the decoding step. Our hybrid protocol will also need to do any AG07 decoding once for every server at every recursive step. However, the response being decoded at each recursive step is smaller than that of the recursive AG07 protocol by a factor of 6 in our hybrid scheme. Therefore, when $d > 1$, the decoding for our hybrid protocol will be comparable to that of recursive AG07.

If there are a significant number of Byzantine servers—those who attempt to maliciously alter the result of the query—then the decoding time will be increased for the G07 iteration of our hybrid scheme, though this increase will not be very significant compared to the server computation of the G07 scheme [11].

5.3 Privacy

The AG07 scheme keeps the client's query private as long as the servers are computationally bounded and as long as the Hidden Lattice Problem and the Differential Hidden Lattice Problem are indeed hard to solve. Our hybrid scheme relies on these assumptions for perfect privacy.

The G07 scheme keeps the query private as long as no more than t servers are colluding to find the contents of the query. Our hybrid scheme also relies on this non-collusion assumption for perfect privacy.

One advantage of our hybrid scheme is that if the privacy assumptions for one of the inner protocols is broken, then the query will still be partially private as long as we use depth $d > 1$. For example, if the G07 non-collusion assumption is broken, then the colluding servers will be able to find out a subset of the database that the desired record is in, but they will not find out which record in that subset is the wanted one as long as the AG07 assumptions still hold. We similarly have partial privacy if the AG07 computational assumptions are broken and the G07 non-collusion assumptions still hold.

This "defence in depth" is a benefit because it may dull some of the fears about using a scheme that a user thinks does not adequately enough guarantee privacy. For example, if someone does not feel that the non-collusion assumption is adequate enough for the G07 scheme, they may be more comfortable using this hybrid scheme because they know that even in the event that too many servers collude, they will still maintain some privacy.

Unlike the hybrid protection extension to G07 (Section 3.1), our protocol does not provide perfect privacy if one of the two privacy assumptions fails. The advantages of using our protocol over hybrid G07 are a significant reduction in computation time and, as will be illustrated in Section 6.1, improved communication cost.

5.4 Robustness

As stated previously, the G07 scheme has the ability to correct for servers not responding or responding incorrectly. The single-server AG07 scheme, however, does not have any robustness.

In 2012, Devet et al. [11] observed that the G07 protocol can be slightly modified to be able to withstand up to $v < k-t-1$ misbehaving servers, with no extra computation or communication cost over the original protocol, in a typical setting where clients aim to fetch multiple records from the database. (The original G07 bound [15] is $v < k - \sqrt{kt}$ when only one record is retrieved.)

An advantage of using G07 in our hybrid protocol is that our hybrid protocol retains exactly the same robustness properties as that scheme: any misbehaviour will be detected at the Ψ IT-PIR multiple-server decoding step.

6 Implementation and Empirical Evaluation

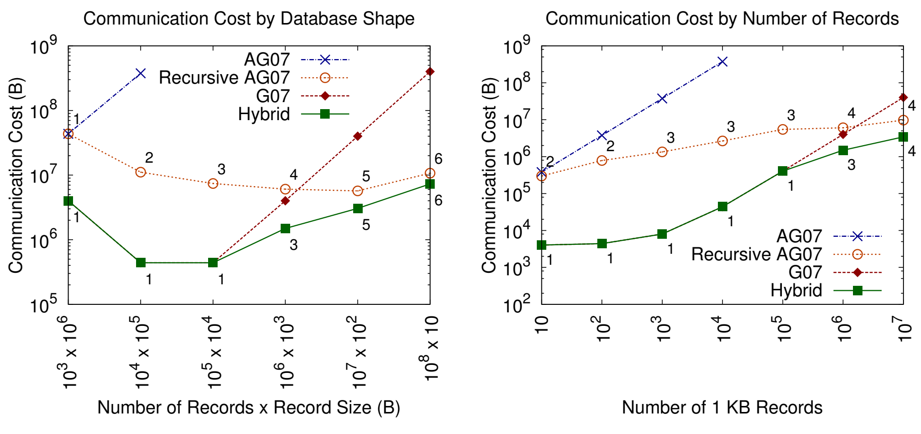
We have implemented these protocols as an extension to the Percy++ [14] library, an implementation of Goldberg's scheme from Section 3.1. We incorporated both the AG07 CPIR scheme and our hybrid scheme. Our implementation will be available in the next release of Percy++.

The implementation of our hybrid PIR system combines the implementations of the two inner protocols (G07 and AG07), using them as black boxes. Given all of the other parameters, our implementation will find the optimal depth (d) and the best way to split the database for the first (IT-PIR) iteration of the scheme (γ_1) to minimize the communication cost.

All of our multi-server queries were run on $\ell = 4$ servers and we used $t = 1$ for the G07 privacy parameter. For our client machine, we used a 2.4 GHz Intel Xeon 8870, and each server machine was a 2.0 GHz Intel Xeon E5-2650. All computations reported here were done in a single thread, so that the reported times reflect total CPU time. However, all of the computations are almost completely parallelizable [10], and using multiple cores would greatly reduce end-to-end latency, though not total CPU time.

6.1 Communication

The plots in Figure 1 illustrate the amount of communication needed for our hybrid scheme and the schemes of which it is comprised. We see that our hybrid protocol uses



(a) (b)

Fig. 1. Comparison of communication used by each scheme. Plot (a) shows the communication used for queries on a 1 GB database for different database shapes. In plot (b), the record size is fixed at 1 KB and we see the communication for different numbers of records. Non-recursive AG07 imposes a limit of approximately 10,000 records so we do not have data points for larger numbers of records. Error bars are present for all data points, but may be too small to see. The datapoint labels for the Hybrid and Recursive AG07 schemes indicate the recursive depth used.

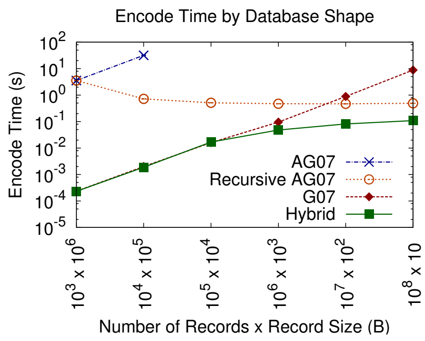
less communication than that of AG07, and no more than that of G07, verifying the analysis in Section 5.1, above.

6.2 Computation

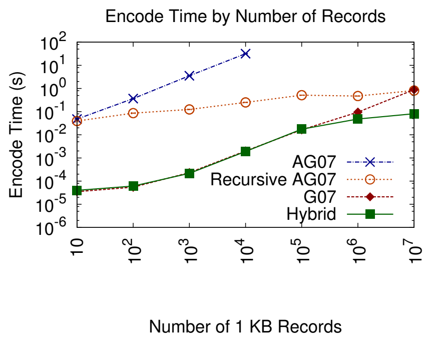
Figure 2 illustrates computation time involved in all three of the computational parts of a PIR system.

Query Encoding: Our experimental results show us that the encoding time is very much related to the size of a PIR request. This is evidenced by how similar Figure 2b is to Figure 1b (the communication associated with the same tests). The query encoding time for G07 is linear in the number of records. On the other hand, the query encoding time for the AG07 scheme is dominated by the d^{th} root of the number of records. Because of this, for larger numbers of records, the hybrid protocol encodes queries faster than G07.

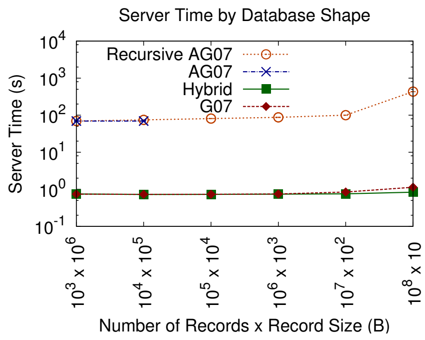
Server Computation: As expected, Figures 2c and 2d show that the server computation time of our hybrid PIR system is very comparable to that of the G07 protocol. The figures also show that our hybrid system performs its server computation approximately 2 orders of magnitude faster than Recursive AG07. As noted above, this time is also highly parallelizable; the times reported in the figure use only a single thread, and so represent total CPU time.



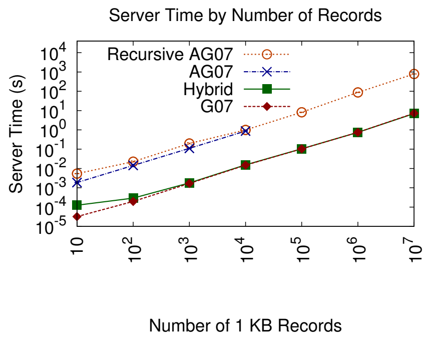
(a)



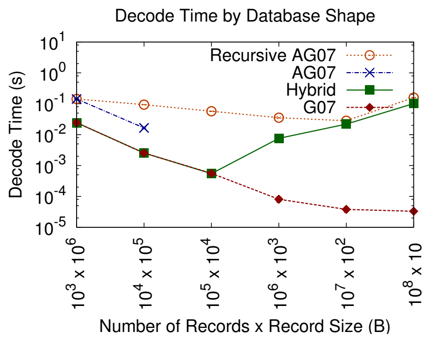
(b)



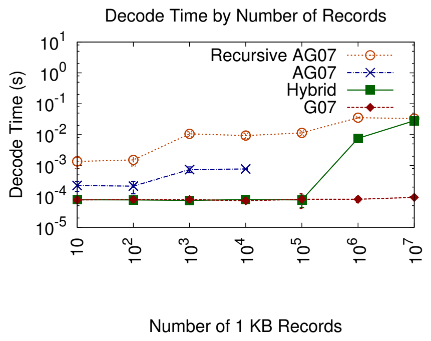
(c)



(d)



(e)



(f)

Fig. 2. Comparison of the time used by the schemes at each computation step. Plots (a,c,e) show the computation time for queries on a 1 GB database for different database shapes. In plots (b,d,f), the record size is fixed at 1 KB and we see the computation time for different numbers of records. Non-recursive AG07 imposes a limit of approximately 10,000 records so we do not have data points for larger numbers of records. Error bars are present for all data points, but may be too small to see.

Response Decoding: Figures 2e and 2f show us that when we have a depth of at least 2 for the hybrid PIR system (i.e. we have at least one iteration of AG07) , the decoding time approaches that of recursive AG07. This is because, unlike the server computation where the cheap G07 computation is being done on the first iteration when the database is large, the cheap G07 decoding is happening on the last iteration, when the response has been reduced in size by $d - 1$ iterations of AG07 decoding. For this reason, the decoding step of our hybrid PIR system is comparable to that of recursive AG07 and not the quicker G07. Even so, we note that the time of the decoding step is quite insignificant compared to the server computation step of a query.

6.3 Total Query Time

In Figure 3 we plot the total time for a query on our hybrid PIR system as well as its component protocols. We show the total time for three different connection speeds between the client and server(s). Figures 3a and 3b use a connection with 9 Mbps download and 2 Mbps upload. This connection was used by Olumofin and Goldberg [22] to represent a home user's connection in 2010. Using the same source [23], we represent a home user in 2014 in Canada or the U.S. with 20 Mbps download and 5 Mbps upload in Figures 3c and 3d. Figures 3e and 3f model a connection over 100 Mbps Ethernet.

Our results show us that the total query time needed for our hybrid PIR system is similar or better than that of G07. We also see that the total query time of recursive AG07 is approximately 2 orders of magnitude larger than that of our system.

These plots also illustrate that our hybrid PIR system does not use much communication time. This is because the total query time of the hybrid system does not improve much when the network capacity is increased. Contrast this with G07 when there are a large number of records—in this case we see a significant improvement in total query time as the network capacity increases.

7 Future Work

Parallel Server Computation We note that the bulk of the computation is on the server side of the protocol. Devet [10] describes experiments showing that the G07 protocol is almost completely parallelizable: using m threads or worker processes will improve the computation latency of G07 by a factor of m. We believe similar results are attainable for our hybrid system and as future work we intend to implement distributed server computation for this scheme.

AG07 using GPUs Aguilar Melchor et al. [1] demostrate how the AG07 scheme can be made much faster by implementing the server-side computations on GPUs instead of CPUs. Our implementation does not include this feature, but we plan on implementing it in the future and investigating how much this will speed up our hybrid protocol.

Security of AG07 AG07's privacy guarantees rely on the hardness of the Hidden Lattice Problem and the Differential Hidden Lattice Problem, as specified by Aguilar Melchor and Gaborit [2]. According to Aguilar Melchor et al. [1] and Olumofin et al. [22]

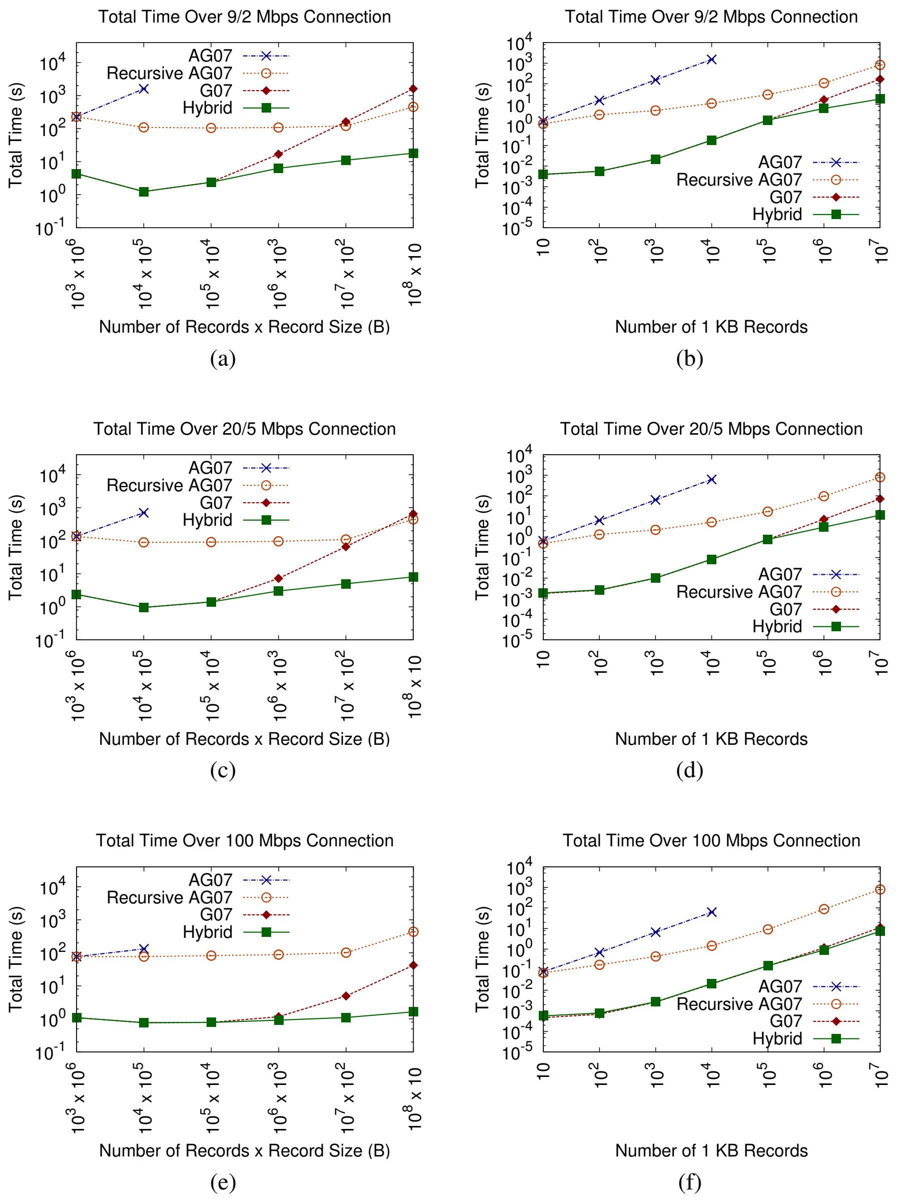


Fig. 3. Comparison of total query time for each scheme. Plots (a,c,e) show the time used for queries on a 1 GB database for different database shapes. In plots (b,d,f), the record size is fixed at 1 KB and we see the time for different numbers of records. Non-recursive AG07 imposes a limit of approximately 10,000 records so we do not have data points for larger numbers of records. Error bars are present for all data points, but may be too small to see. Connections specified as A/B Mbps indicate A Mbps download bandwidth and B Mbps upload bandwidth.

the security of this scheme is not well understood. Future work could involve investigating the security of the scheme and either developing a security proof or altering the scheme to make it provably secure.

8 Conclusion

We introduce a hybrid Private Information Retrieval protocol that combines the low communication and computation costs of multiple-server IT-PIR protocols with the ability of single-server CPIR protocols to do recursion. We show that our protocol inherits several positive aspects of both types of protocols and mitigates the negative aspects. In particular, our protocol maintains partial privacy of client query information if the assumptions made by one of the inner protocols is broken.

We have implemented our protocol as part of the open-source Percy++ library for PIR, and using this implementation, demonstrated that our protocol performs as well or better than PIR schemes by Aguilar Melchor and Gaborit and by Goldberg. Our hybrid scheme is particularly effective when the number of records in a database is large relative to the size of each record—a situation that arises naturally in a number of network scenarios, including TLS certificate checking, private LDAP lookups, sensor networks, and more.

Acknowledgements. We thank Ryan Henry for the idea that inspired us to explore this new type of Hybrid PIR. We gratefully acknowledge NSERC and ORF for funding this research. This work benefited from the use of the CrySP RIPPLE Facility at the University of Waterloo.

References

1. Aguilar Melchor, C., Crespin, B., Gaborit, P., Jolivet, V., Rousseau, P.: High-Speed Private Information Retrieval Computation on GPU. In: SECURWARE, IEEE (2008) 263–272
2. Aguilar-Melchor, C., Gaborit, P.: A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol. In: WEWORC 2007. (July 2007)
3. Asonov, D.: Private Information Retrieval: An overview and current trends. In: ECDPvA Workshop. (2001)
4. Beimel, A., Ishai, Y., Malkin, T.: Reducing the Servers' Computation in Private Information Retrieval: PIR with Preprocessing. J. Cryptology **17**(2) (2004) 125–151
5. Chaum, D., Carback, R., Clark, J., Essex, A., Popoveniuc, S., Rivest, R.L., Ryan, P.Y.A., Shen, E., Sherman, A.T., Vora, P.L.: Scantegrity II: End-to-end verifiability by voters of optical scan elections through confirmation codes. IEEE Transactions on Information Forensics and Security **4**(4) (2009) 611–627
6. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private Information Retrieval. In: 36th Annual IEEE Symposium on Foundations of Computer Science (FOCS'95). (October 1995) 41 –50
7. Chor, B., Gilboa, N., Naor, M.: Private Information Retrieval by Keywords. Technical Report TR CS0917, Department of Computer Science, Technion, Israel (1997)
8. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private Information Retrieval. J. ACM **45** (November 1998) 965–981

9. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a Type III Anonymous Remailer Protocol. In: IEEE Symposium on Security and Privacy, IEEE Computer Society (2003) 2–15
10. Devet, C.: Evaluating Private Information Retrieval on the Cloud. Technical Report 2013-05, CACR (2013) http://cacr.uwaterloo.ca/techreports/2013/cacr2013-05.pdf.
11. Devet, C., Goldberg, I., Heninger, N.: Optimally Robust Private Information Retrieval. In: 21st USENIX Security Symposium. (2012)
12. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: 13th USENIX Security Symposium. (2004)
13. Gertner, Y., Goldwasser, S., Malkin, T.: A Random Server Model for Private Information Retrieval. In: 2nd International Workshop on Randomization and Approximation Techniques in Computer Science. (1998) 200–217
14. Goldberg, I.: Percy++ project on SourceForge. http://percy.sourceforge.net Accessed February 2014.
15. Goldberg, I.: Improving the Robustness of Private Information Retrieval. In: 2007 IEEE Symposium on Security and Privacy. (2007) 131–148
16. Henry, R., Olumofin, F.G., Goldberg, I.: Practical PIR for Electronic Commerce. In: ACM Conference on Computer and Communications Security. (2011) 677–690
17. Kikuchi, H.: Private Revocation Test using Oblivious Membership Evaluation Protocol. In: 3rd Annual PKI R&D Workshop. (2004)
18. Kushilevitz, E., Ostrovsky, R.: Replication Is Not Needed: Single Database, Computationally-Private Information Retrieval. In: FOCS. (1997) 364–373
19. Laurie, B., Langley, A., Kasper, E.: Certificate Transparency. RFC 6962 (June 2013)
20. Mittal, P., Olumofin, F., Troncoso, C., Borisov, N., Goldberg, I.: PIR-Tor: Scalable Anonymous Communication Using Private Information Retrieval. In: 20th USENIX Security Symposium. (2011) 475–490
21. Olumofin, F., Goldberg, I.: Privacy-preserving Queries over Relational Databases. In: 10th International Privacy Enhancing Technologies Symposium. (2010) 75–92
22. Olumofin, F., Goldberg, I.: Revisiting the Computational Practicality of Private Information Retrieval. In: 15th International Conference on Financial Cryptography and Data Security. (2011) 158–172
23. Ookla: Net Metrics for Canada and the United States. http://www.netindex.com Accessed February 2014.
24. Ryan, P.Y.A., Schneider, S.A.: Prêt à Voter with Re-encryption Mixes. In: ESORICS. (2006) 313–326
25. Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., Adams, C.: X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. RFC 6960 (June 2013)
26. Sassaman, L., Cohen, B.: The Pynchon Gate: A Secure Method of Pseudonymous Mail Retrieval. In: In Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2005). (2005) 1–9
27. Sermersheim, J.: Lightweight Directory Access Protocol (LDAP): The Protocol. RFC 4511 (June 2006)
28. Shamir, A.: How to share a secret. Commun. ACM **22** (November 1979) 612–613
29. Sion, R., Carbunar, B.: On the Computational Practicality of Private Information Retrieval. In: Proceedings of the Network and Distributed Systems Security Symposium. (2007)
30. Xively: Public Cloud for the Internet of Things. http://www.xively.com Accessed February 2014.

A Details of the AG07 Protocol

Algorithm A.1 AG07 Query Generation

```
Input: Desired record index: $i_0$
1: Generate two random $N \times N$ matrices $A$ and $B$ such that $A$ is invertible.
2: For each $i \in \{1, \ldots, n\}$ generate random invertible $N \times N$ matrix $P_i$.
3: Generate the random scrambling matrix $\varDelta$, a diagonal invertible $N \times N$ matrix.
4: For each $i \in \{1, \ldots, n\}$ generate soft noise matrix $C_i$, an $N \times N$ matrix over $\{-1, 1\}$.
5: Convert $C_{i_0}$ to a hard noise matrix by setting all diagonal terms to $q$.
6: For each $i \in \{1, \ldots, n\}$ compute the disturbed $N \times 2N$ matrices
       $M'_i = [P_i A | P_i B + C_i \varDelta]$.
7: Choose random column permutation $\mathcal{P}$ and for each $i \in \{1, \ldots, n\}$ compute $M_i = \mathcal{P}(M'_i)$.
8: Send $M_1, \ldots, M_n$ to the server.
```

Algorithm A.2 AG07 Server Computation

```
Input: Query from the client: matrices $M_1, \ldots, M_n$
   Database records represented as $L \times N$ matrices with $w_{AG}$-bit elements: $D_1, \ldots, D_n$
1: Compute response matrix
       $R = \sum_{i=1}^{n} D_i M_i$.
2: Send $R$ to the client.
```

Algorithm A.3 AG07 Response Decoding

```
Input: Response from the server: $R$
   Random values generated in Algorithm A.1: $A, B, \varDelta, \mathcal{P}$
1: Undo the column permutation
       $R' = \mathcal{P}^{-1}(R) = [V_U | V_D]$
   where $V_U$ and $V_D$ are $L \times N$ matrices
2: Compute scrambled noise matrix
       $E' = V_D - V_U A^{-1} B$.
3: Compute the unscrambled noise matrix
       $E = E' \varDelta^{-1}$.
4: For each term $e_{x,y}$ of $E$ remove noise by computing
       $f_{x,y} = e_{x,y} - \epsilon_{x,y}$
   with
       $\epsilon_{x,y} = \begin{cases} e_{x,y} \mod q & : e_{x,y} \mod q < \frac{q}{2} \\ (e_{x,y} \mod q) - q & : \text{otherwise} \end{cases}$
5: Recover database record $D_{i_0}$ by computing words
       $d_{x,y} = f_{x,y} \cdot q^{-1}$.
```

B Details of the G07 Protocol

Algorithm B.1 G07 Query Generation

```
Input: Desired record index: $i_0$
1: Choose $\ell$ distinct non-zero elements
       $\alpha_1, \ldots, \alpha_\ell \in \mathbb{F}$.
2: Generate $n$ random degree-$t$ polynomials
       $f_1, \ldots, f_n \in \mathbb{F}[x]$
   such that $f_{i_0}(0) = 1$ and $f_i(0) = 0$ for all $i \neq i_0$.
3: For each $j \in \{1, \ldots, \ell\}$ compute
       $\mathbf{v_j} = \langle f_1(\alpha_j), \ldots, f_n(\alpha_j) \rangle$ .
4: For each $j \in \{1, \ldots, \ell\}$, send $\mathbf{v_j}$ to server $j$.
```

Algorithm B.2 G07 Server Computation

```
Input: Query from client: vector $\mathbf{v_j}$
1: Compute response vector
       $\mathbf{r_j} = \mathbf{v_j} \cdot D$.
2: Send vector $\mathbf{r_j}$ to the client.
```

Algorithm B.3 G07 Response Decoding

```
Input: Responses from the servers: $\mathbf{r_1}, \ldots, \mathbf{r_k}$
1: For $h \in \{1, \ldots, m\}$, form the codeword
       $\mathbf{c_h} = \langle \mathbf{r_1}[h], \ldots, \mathbf{r_k}[h] \rangle$ .
2: Perform Reed-Solomon decoding on each codeword $\mathbf{c_1}, \ldots, \mathbf{c_m}$ to yield polynomials
   $F_1, \ldots, F_m$.
3: Output the database record $D_{i_0} = \langle F_1(0), \ldots, F_m(0) \rangle$.
```