One (Block) Size Fits All: PIR and SPIR with Variable-Length Records via Multi-Block Queries

Ryan Henry, Yizhou Huang, and Ian Goldberg
Cheriton School of Computer Science
University of Waterloo
Waterloo ON, Canada N2L 3G1
{rhenry, y226huang, iang}@cs.uwaterloo.ca

Abstract

We propose a new, communication-efficient way for users to fetch multiple blocks simultaneously in Goldberg's robust information-theoretic private information retrieval (IT-PIR) scheme. Our new multi-block IT-PIR trades off some Byzantine robustness to improve throughput without affecting user privacy. By taking advantage of the recent Cohn-Heninger multi-polynomial list decoding algorithm, we show how realistic parameter choices enable the user to retrieve several blocks without increasing the communication or computation costs beyond what is required to retrieve a single block, and argue that the resulting scheme still maintains essentially optimal Byzantine robustness in practice. We also derive optimal parameters for our construction, which yields communication costs within a small factor of the lowest possible.

With our new multi-block IT-PIR protocol as a starting point, we construct four new symmetric PIR (SPIR) protocols that each support variable-length database records. By decoupling the PIR block size from the lengths of individual database records, we are free to fix the block size to its communication-optimal value without artificially restricting the contents and layout of the records. Moreover, it is straightforward to augment three of our four new SPIR constructions with efficient zero-knowledge proofs about the particular records a user is requesting in a given query; this makes it easy to implement pricing and access control structures over the records using standard techniques from the literature. The resulting SPIR protocols are therefore well suited to privacy-preserving e-commerce applications, such as privacy-friendly sales of e-books, music, movies, or smart phone and tablet apps.

Keywords **— Private information retrieval, symmetric PIR, oblivious transfer, usable PIR, zero-knowledge proofs, privacy-enhancing technologies.**

I. Introduction

Privacy-enhancing technologies (PETs) are technologies that aim to empower users with control over the dissemination and use of information about themselves and about their day-to-day activities. Modern PETs employ sophisticated cryptographic techniques to facilitate interactions that would otherwise appear impossible to conduct in a privacy-friendly way. This approach lets PETs derive their privacy guarantees from the security properties of the underlying cryptographic primitives they use, which in turn derive their security properties from basic facts (or conjectures) about *information theory* or *computational complexity theory*. A good deal of modern cryptography focuses on the latter, with many cryptographic protocols relying on assumptions about it being infeasible — rather than impossible — for an adversary to extract some private information by observing or participating in a protocol run. The security proofs for these computationally secure PETs therefore hold with respect to adversaries that have only limited (computational and algorithmic) resources at their disposal, but not necessarily with respect to "all-powerful" adversaries that can solve presumed hard problems like factoring or computing discrete logarithms. This is in contrast to information-theoretically secure PETs: if a PET is information-theoretically secure, then no amount of resources (or future algorithmic advances) can give the adversary *any* advantage in extracting the user's private information. Of course, proving that a system can provide such strong privacy guarantees requires some equally strong (non-computational) assumptions. One common such assumption, which is relied upon by secret sharing schemes [36], some cryptographic voting protocols [13, 35], mix networks [19] and onion routing networks [21], among others, is that no more than some threshold number of agents are malicious and colluding against the user to extract her secrets.

Private Information Retrieval

Private information retrieval (PIR) is one particular class of PET that helps users retrieve information from a database in a way that is highly respectful of privacy. A user's query encodes a set of keywords [15], the indices of certain records [16], or some simple SQL statements [33], and the database server processes the query and responds without learning any nontrivial information about what data the user is after. In computationally secure PIR (CPIR), the user employs public-key cryptography to encode these keywords, indices, or SQL statements in her query in a way that enables the database server to respond with the correct (encrypted) data, while making it computationally infeasible for the database server to learn what data the user has requested. In information-theoretically secure PIR (IT-PIR), the stricter privacy requirement presents somewhat of a paradox, which precludes such a reliance on public-key cryptography: the user's query must not contain *any* information about what particular data she is requesting, yet the database server must nonetheless respond in a way that lets the user extract this very data!

The trivial way to solve the PIR problem is to have the database server respond to every query with the entire database, and then let the user carry out her own local keyword searches, index lookups, or SQL statement evaluations. This trivial solution may be information-theoretically secure, but it is not very interesting and it is highly impractical for large databases since the communication cost is linear in the length of the database. To exclude this and related trivial solutions, the PIR literature only considers protocols whose total communication cost is strictly less than (and scales sublinearly with) the length of the database. Alas, it is not obvious that sublinear communication can actually make PIR more practical than the trivial solution; indeed, in their oft-cited 2007 study, Sion and Carbunar found that not one CPIR protocol from the arsenal at their disposal could — or likely ever will — outperform the trivial PIR protocol, given the relative speeds of processors and networks and the trends in how quickly they increase [37]. However, that paper only considered CPIR protocols (and only those published prior to 2007 — at least one later CPIR protocol [31] has been shown to be faster than the trivial solution [34]), which naturally raises the following question: *can IT-PIR do any better?* Intuition might suggest that it surely cannot; in fact, it is not immediately clear that IT-PIR with sublinear communication is even possible. After all, the database server knows precisely what sequence of bits it receives from and sends to each user. If some database bits are not somehow "included" in the response, then given a sufficiently clever algorithm and sufficient computational resources, it seems inevitable that the database server could deduce something about which database bits the user has requested. In their seminal paper on PIR [16], Chor et al. showed that nontrivial IT-PIR is indeed impossible when there is only a single database server; however, they then went on to construct a *multi-server* IT-PIR protocol whose security holds if not every database server colludes against the user. Several other IT-PIR protocols have since been proposed [3, 23, 25, 38], each building on Chor et al.'s idea of sharing queries among multiple noncolluding database servers.[1] In their 2011 follow-up to Sion and Carbunar's paper, Olumofin and Goldberg found that a number of multi-server IT-PIR protocols in the literature are indeed more efficient than trivial PIR, in some cases by up to three orders of magnitude [34].

Our contributions

Chor et al.'s foundational work on PIR modeled the database as a string of n bits out of which the user retrieves the i^{th} bit while keeping the index i of that bit private. A handful of subsequent papers have extended this basic model by subdividing the n-bit database into some number r of b-bit *blocks*, out of which the user retrieves the i^{th} block — rather than the i^{th} bit — without revealing i. This latter model more closely approximates real-world databases than does the former, but it is still insufficient for modeling databases with variable-length records, such as those serving multimedia content. This work therefore initiates the study of PIR over such variable-length records. Our main contributions are as follows:

1) We revisit Goldberg's robust IT-PIR protocol [25] and extend it to support *multi-block queries*, wherein the user fetches several b-bit blocks in a single query without revealing either the number of blocks fetched or the index of any block. Multi-block PIR queries naturally lead to PIR queries over variable-length records without padding or other efficiency-harming workarounds.
2) We redo Goldberg's optimal-block-size analysis, taking into consideration both our new multi-block queries and the lengths of individual database records. Our analysis indicates that, in practice, the expected communication cost and Byzantine-robustness of our protocol are both within a small factor of the lowest possible (with near-optimal robustness relying on the assumption that the PIR servers are rational agents).
3) We extend our new multi-block IT-PIR to construct four new symmetric PIR protocols, each of which supports queries over variable-length records. We empirically evaluate the performance of two of these

[1]The intuitive objection to IT-PIR can be salvaged as follows. Barring any clever precomputation scheme (and associated auxiliary storage), the *computational* cost of any PIR protocol (information-theoretic or otherwise) must be at least linear in the size of the database, since the user cannot possibly learn anything about a database bit that the server does not use to help compute the query response [2].

new protocols and find that they are practical for use in certain application domains, even on modest hardware configurations.

II. Background

A. Goldberg's robust IT-PIR

Our construction in this paper extends Goldberg's robust IT-PIR [25], which is more or less a generalization of Chor et al.'s original IT-PIR protocol. We focus on Goldberg's IT-PIR for three reasons: 1) IT-PIR protocols like Goldberg's protocol are faster than any known CPIR protocol by an order of magnitude or more [34], 2) the communication cost of Goldberg's protocol is within a small factor of optimal if the user is interested in a relatively large block of data [25] (see below), which is the case in, e.g., certain database-driven e-commerce applications, and 3) while conceptually quite simple, Goldberg's scheme has a rich algebraic structure that we exploit in Section III. Goldberg models the database as an r-by-s matrix $\mathbf{D}$ over some finite field $\mathbb{F}$. Each row of $\mathbf{D}$ is a single database *block*; that is, $\mathbf{D}$ consists of r blocks, each of which contains some data represented by a string of s field elements. Users query $\mathbf{D}$ for a block using the following basic fact from linear algebra: if e_j is the j^{th} standard basis vector of $\mathbb{F}^r$ (i.e., the length-r row vector over $\mathbb{F}$ with unity in column j and zero elsewhere), then taking the vector-matrix product $e_j \cdot \mathbf{D}$ yields row j of $\mathbf{D}$. Goldberg uses Shamir's polynomial secret sharing scheme [36] to split e_j componentwise into ℓ vectors of shares, which the user submits to ℓ different database servers. Each database server returns the product of its respective share vector with $\mathbf{D}$. By the linearity of Shamir secret shares, interpolating the query responses componentwise at $x = 0$ still yields row j of $\mathbf{D}$, even though no database server has been given any information about the index j. This most basic form of the protocol only supports retrieval by index, but standard tricks allow for queries that are more expressive (for example, keyword searches [15] and SQL queries [33]) on top of this framework.

Robustness. Goldberg's protocol is *optimally robust* with respect to the number of malicious or Byzantine database servers that it can tolerate. [20] Suppose the user encodes her query using secret sharing polynomials of degree (at most) $t < \ell$, and that $k \leq \ell$ servers respond. Given these parameters, Shamir secret sharing information-theoretically hides the contents of the user's query from any coalition of at most t of the ℓ database servers. Goldberg [25] notes that if $t < k$ and $v \leq k - \lfloor\sqrt{kt}\rfloor - 1$, then Guruswami-Sudan list decoding [26] can extract the correct block from the set of responses even when up to v servers return (possibly maliciously correlated) incorrect responses. More recently, Devet et al. [20] showed that replacing Guruswami-Sudan list decoding with the recent Cohn-Heninger multi-polynomial list decoding algorithm [17] extends robustness in Goldberg's protocol to the theoretical limit of up to $v = k - t - 2$ Byzantine database servers. Looking ahead, our main contribution in this work is to observe that Devet et al.'s optimally robust variant of Goldberg's protocol leaves sufficiently many degrees of freedom for a single query vector to evaluate to several standard basis vectors at different inputs. We leverage this observation to construct *multi-block queries* that can dramatically improve the throughput of Goldberg's scheme. Before doing so, however, we shall first examine some other aspects of the protocol that will be impacted by this modification.

Communication cost. The communication-optimal block size for Goldberg's *single-block* IT-PIR occurs when $r = s = \sqrt{N}$ for a database comprised of N field elements (where $k \approx \ell$ is unknown in advance to the user). For each block that a user fetches from the database, she sends r field elements to each of ℓ servers and receives s field elements from each of k responding servers; thus, the total communication cost per query is $(\ell + k)\sqrt{N}$ field elements when $r = s$, and somewhat higher otherwise. If the user is interested in an entire block of $s \approx \sqrt{N}$ field elements, then this cost is within a 2ℓ factor of the theoretical optimum (which itself is clearly bounded below by the s field element communication cost of a *non-private* query for the same data).[2] Of course, insisting that each record has length $s = \sqrt{N}$ and that $r = s$ is quite restrictive in practice; one cannot generally rely on all records in a database being of a fixed length, nor on the number of records being somehow related to their lengths. Using a suboptimal choice of parameters can address part of this problem, but it also serves to increase the communication cost. A second possible workaround immediately comes to mind: pack multiple records into a block (together with some padding) to handle records that are smaller than $\sqrt{N}$ field elements, and require users to submit multiple queries to retrieve records that are larger than $\sqrt{N}$ field elements. However, if users must submit multiple queries to retrieve large records, then it becomes necessary for *all users* to submit the *maximum possible number* of queries needed

[2]There is a large body of research on communication-efficient IT-PIR, and quite a few protocols in the literature have asymptotically lower communication cost than Goldberg's protocol if the user only wishes to retrieve a *single bit* (or a small number of bits) from an n-bit database. The state of the art in this respect appears to be Yekhanin's 3-server IT-PIR based on locally decodable codes, which has communication complexity $n^{O(1/\lceil\lg p\rceil)}$ for any Mersenne prime p [38]. Assuming that there are infinitely many Mersenne primes, this yields a scheme with communication complexity $n^{O(1/\log\log n)}$ for infinitely many n. Nonetheless, it is clear that no protocol — private or otherwise — can use less than $O(s)$ communication to fetch an s-element block. If $s \geq \sqrt{N}$ and we treat ℓ as a constant, then Goldberg's IT-PIR achieves this optimal asymptotic communication cost, and does so with only a reasonably small coefficient (i.e., less than $\ell + k$) hidden behind the "big O".

for *any* record, regardless of the size of the actual record they seek. Otherwise, the database servers could infer some information about the records a user is requesting by observing how many blocks she queries for. A similar line of reasoning reveals that, as long as there exists *at least one* record that is $\sqrt{N}$ field elements long, then the above observation regarding the near-optimality of Goldberg's IT-PIR holds; moreover, as we will see in Section III, using an appropriate choice of parameters extends this near-optimality to any database containing a record that is *more than* $\sqrt{N}$ field elements long.

B. Symmetric PIR and oblivious transfer

Symmetric PIR (SPIR) is a variant of PIR that extends privacy protection to the database servers by insisting that users must not learn any nontrivial information about parts of the database that they do not explicitly request [24]. A close relative of SPIR is 1-out-of-n oblivious transfer [8] ($\binom{1}{n}$OT, or just OT); SPIR and OT are so similar, in fact, that many researchers do not distinguish between them. For the purposes of this paper, SPIR refers to protocols that use strictly sublinear communication, while OT refers to protocols that use (at least) linear communication. Kushilevitz and Ostrovsky note that it is theoretically possible to transform any PIR protocol into SPIR using general zero-knowledge proof techniques and some encryption [30]. We illustrate the distinction between SPIR and OT by comparing two schemes that are constructed using Kushilevitz and Ostrovsky's suggestion; that is, we compare Henry et al.'s multi-server SPIR protocol [28] with Camenisch et al.'s simulatable adaptive OT protocol [12]. The former protocol derives from Goldberg's IT-PIR and the latter from trivial PIR (i.e. downloading the entire database). For brevity, we only give a high-level overview of both protocols and refer the reader to the respective references for further details.

Henry et al.'s SPIR. The user in Henry et al.'s scheme forms a query exactly as in Goldberg's IT-PIR, and then commits to each of her secret sharing polynomials using Kate et al.'s polynomial commitments [29], which we briefly review in Section IV-A. (One consequence of using polynomial commitments in this way is that the underlying IT-PIR must be instantiated with a prime field $\mathbb{F}$ whose order is large enough to satisfy certain cryptographic assumptions. This has a small, though not insignificant, impact on the efficiency of the underlying IT-PIR.) She then uses (noninteractive, batch) zero-knowledge proofs to convince each database server that the committed polynomials are consistent with the set of shares in her query, and that the committed polynomials evaluate (componentwise) to a standard basis vector e_j at $x = 0$, for some $1 \leq j \leq r$. Of course, a clever user could still obtain some information about other parts of the database either by sending a different vector of commitments to each database server, or by choosing her vector of secret sharing polynomials nonrandomly so that it also passes through one or more additional standard basis vectors at different inputs $x \neq 0$. (We use this latter "attack" in our multi-block query construction in Section III and throughout Section IV.) Henry et al. thwart both of these attacks in one fell swoop: the database servers each seed a pseudorandom generator (PRG) with a common secret value (for example, a digest of the entire database) and the vector of polynomial commitments from the user. Each database server then uses the output of this PRG to *rerandomize* the user's query before processing it. (Intuitively, this rerandomization replaces each of the non-free coefficients in the secret sharing polynomials that contain the user's response with new, uniform random ones that are unknown to the user.) The polynomial commitments and zero-knowledge proofs that Henry et al. add to Goldberg's IT-PIR protocol increase the communication cost by only a small constant factor (plus a small additive term), so their protocol preserves the sublinear asymptotic communication cost of the underlying IT-PIR. The linear computation of the IT-PIR dominates the computation cost, albeit with some additional overhead owing mostly to $\Theta(r)$ full-length exponentiations (in an elliptic curve group) for the user, and $\Theta(r)$ short exponentiations (i.e., with ≈ 40-bit exponents) for each database server [28, Figure 1].

Camenisch et al.'s OT. Camenisch et al. take an entirely different approach with their OT protocol. In an initialization phase, the (single) database server encrypts each individual record using a different, specially chosen cryptographic key, and then publishes the encrypted database for any user to download in its entirety. To retrieve a plaintext record from the encrypted database, the user must first download the *entire* encrypted database (to avoid revealing which portion she is interested in), and then obtain the appropriate decryption key for her desired record from the database server. To accomplish this, Camenisch et al. cleverly employ a *unique signature scheme*[3]: the key needed to decrypt the record at index j is just (perhaps some publicly known function of) the unique signature on the message "j" under the database server's public key. Hence, to decrypt a record, the user just requests a blind signature on that record's index, and attaches a zero-knowledge proof that attests to the well-formedness of the blinded message. Privacy for the database server follows from the security of the encryption and the (one-more-)unforgeability of the signature scheme, while privacy for the user follows from the (unconditional) hiding of the blind signature scheme and the trivial download step. The zero-knowledge proofs

[3]A *unique signature scheme* is a cryptographic signature scheme with the nonstandard property that, for any given (message, public key) pair, there exists one and only one valid signature on the given message under the given public key.

and blind signature that the client uses to retrieve her decryption key increase the communication cost of the underlying trivial PIR by a factor of two plus a small constant; thus, the protocol preserves trivial PIR's linear asymptotic communication cost.

C. Pricing and access control

The Henry et al. SPIR protocol and the Camenisch et al. OT protocol have more in common than just being constructed from simpler PIR protocols using Kushilevitz and Ostrovsky's heuristic: the creators of both schemes have augmented their respective protocols to support some additional, nonstandard features that make them suitable for deployment in scenarios outside of the standard SPIR and OT use cases. In particular, both protocols support flexible *pricing* [9, 28] and *access control* [10, 11, 28] structures. As such, variants of either protocol would seem to be particularly well suited to use in privacy-preserving e-commerce applications such as privacy-friendly sales of e-books, music, movies, or smart phone and tablet apps. Henry et al. refer to their extended SPIR, which provides simultaneous support for pricing and access control, as *priced SPIR* (PSPIR); Camenisch et al. call their pricing-extended OT protocol *priced OT* (POT), and their access-control-extended OT protocol *OT with access control* (OTAC). Unfortunately, both Henry et al.'s PSPIR protocol and Camenisch et al.'s POT/OTAC protocols possess certain characteristics that make them less than ideal for e-commerce in practice: Henry et al. require each database block to be (padded to) a fixed length, which is unrealistic for some types of multimedia files like movies or music, and Camenisch et al. assume a static database that is small enough for each user to download in its entirety. It is (intentionally) straightforward to extend three of our four new SPIR protocols in Section IV with either Henry et al.'s or Camenisch et al.'s zero-knowledge proofs (or related techniques) to implement pricing and access control, although full details of how to do this is beyond the scope of the present paper. Moreover, each of the resulting schemes is free from the aforementioned shortcomings; they all use strictly sublinear communication and do not place unrealistic restrictions on the size of the database or the lengths of records contained therein. Measurements by Henry et al. indicate that using their PSPIR protocol on a 44-gigabyte database is more than two orders of magnitude faster than trivial download over a 9 Mbps broadband Internet connection [28], with the performance gap increasing as the database grows in size. Each of our SPIR protocols is at least as efficient as their construction is, and we therefore conclude that our protocols are among the most practical choices in the literature for real-world-scale privacy-preserving e-commerce applications.

III. Multi-block queries in Goldberg's IT-PIR

In Section II-B, we described a potential attack — previously noted by Henry et al. [28] — on symmetric variants of Goldberg's IT-PIR. In particular, Henry et al. note that the user might try to cheat the database servers by encoding several standard basis vectors into a single query vector, thus enabling her to learn about several database blocks with just that one query. It turns out that, if the user encodes $q > 1$ basis vectors in a query using degree (at most) $t+q-1$ secret sharing polynomials, this attack is information-theoretically undetectable by any coalition of up to t cooperating servers. The proof of this assertion echoes the security proof for conventional Shamir secret sharing: if a coalition of database servers has t vectors of such shares, then for each hypothesized set of q standard basis vectors, the coalition can construct one and only one vector of polynomials of degree at most $t+q-1$ that passes componentwise through the t given and q hypothesized vectors. By construction, each candidate set of hypothesized basis vectors is equally likely, and the coalition gains no information about the actual set of vectors. (The generalization of Shamir secret sharing that this attack implicitly uses is really a *ramp scheme* [5], since an adversary with access to more than t but fewer than $t+q$ vectors of shares has some incomplete information about the q secret basis vectors.) The key idea of this section is to recast Henry et al.'s observation as a *feature* of — rather than an attack against — the underlying IT-PIR protocol. In particular, we show how to construct *multi-block* queries to fetch several blocks for the (communication and computation) cost of one, thus greatly reducing the multiplicative factor that separates the cost of Goldberg's protocol from the theoretically optimal cost.

Suppose that $\mathbf{D}$ is an N-field-element database that uses the communication-optimal block size in Goldberg's ℓ-server IT-PIR with privacy threshold t. Recall that the communication cost to fetch a single block from $\mathbf{D}$ is then $\left(\ell+k\right)\sqrt{N}$ field elements when k servers respond. The cost to fetch a large record that occupies q blocks by using q consecutive queries (as suggested in Section II-A) is $q\cdot\left(\ell+k\right)\sqrt{N}$ field elements. Guruswami-Sudan list decoding can provide robustness for each such query against up to $v=k-\lfloor\sqrt{kt}\rfloor-1$ Byzantine database servers, while using Devet et al.'s suggested multi-polynomial list decoding increases this to $v=k-t-2$. We thus observe potential for a tradeoff between Devet et al.'s newfound robustness and the cost of querying for q blocks at once: for fixed ℓ, t, k, requesting q blocks together in a *single query* (for any $1 \le q \le k-t-1$) using Henry et al.'s "attack" decreases the robustness bound v from $k-t-2$ to $k-t-q-1$ servers, *but does not increase the communication cost beyond* $\left(\ell+k\right)\sqrt{N}$ *field elements* (provided the actual number of Byzantine responses v' is sufficiently small compared to

N	size of database in field elements
ℓ	total number of PIR servers
t	privacy threshold (max. coalition size)
k	number of PIR servers that respond
v	Byzantine-robustness bound
v'	actual number of Byzantine responses
q	number of blocks the user is requesting

Table 1: Listing of relevant parameters from the above problem setup. The parameters are subject to the following constraints: N is a perfect square; $k \le \ell$ and $t < k$; and $v = k - t - q - 1 \ge 0$ (which implies that $q \le k - t - 1$).

$v = k - t - q - 1$; see below). As a point of reference for the remainder of this section, Table 1 lists each of the relevant parameters in the above setup.

Let us briefly examine what happens when the user sets q so as to preserve the Guruswami-Sudan robustness bound of up to $v = k - \lfloor\sqrt{kt}\rfloor - 1$ Byzantine servers from Goldberg's original IT-PIR paper [25].[4] To achieve this bound, she simply fixes the number of blocks per query as

$$\begin{aligned} q &= \big(k - t - 2\big) - \big(k - \lfloor\sqrt{kt}\rfloor - 1\big) + 1 \\ &= \lfloor\sqrt{kt}\,\rfloor - t \end{aligned}$$

(using secret sharing polynomials of degree at most $t + q - 1$). It is clear that the level of privacy does not change: the secret sharing still perfectly hides the query if at most t out of the ℓ database servers collude. If the number of Byzantine database servers happens to be $v' < k - \lfloor\sqrt{k(t+q-1)}\rfloor$, then the communication and computation costs are identical to those for a standard, single-block query, which is a q-fold improvement in throughput compared to issuing q consecutive queries to fetch the same set of blocks. Otherwise, if $v' \ge k - \lfloor\sqrt{k(t+q-1)}\rfloor$, then the user may need to issue up to $m = \lceil v'/(k - v' - \lfloor\sqrt{kt}\rfloor)\rceil$ queries to achieve unique decoding with high probability. This expression is (tightly) bounded above by v' queries, which is at most $k - \lfloor\sqrt{kt}\rfloor - 1$ given our choice of q. (In general, if a query encodes q standard basis vectors and v' servers are Byzantine, then the user must issue up to $m \le \lceil v'/(k - v' - t - q)\rceil$ queries to guarantee unique decoding with high probability [20].) Fortunately, when $m > 1$ it turns out that, although each of the $m - 1$ "follow-up" queries must involve the same set of k database servers that responded to the first query, the follow-up queries do not need to be for the same set of q blocks. In particular, if the user already plans to fetch mq or more blocks from the database, then the follow-up queries introduce no communication or computation overhead compared to the case of $v' = 0$ Byzantine servers [20]. (Equivalently, if the user ultimately plans to request B blocks, then using multi-block queries does not introduce *any* overhead — indeed, it *usually* reduces both communication and computation cost — provided $\lceil B/q\rceil \ge \lceil v/(k - v - t - q)\rceil$.) Therefore, the worst case overhead for our example of setting q to preserve the Guruswami-Sudan bound occurs when $m = k - \lfloor\sqrt{kt}\rfloor - 1$ and the user seeks to fetch exactly $q = \lfloor\sqrt{kt}\rfloor - t$ blocks. When this worst case occurs, multi-block queries do not improve costs compared to issuing separate, single-block queries; indeed, it is easy to show that $m \ge q$. Nonetheless, when $v' \le \frac{q-1}{q}\big(k - \lfloor\sqrt{kt}\rfloor\big)$, we have that $m < q$ and therefore that the communication and computation cost are always strictly lower than those resulting from q consecutive single-block queries. To make the above example more concrete, suppose that $\ell = k = 10$ and $t = 5$ so that the user sets $q = (10-5-2) - (10 - \lfloor\sqrt{50}\rfloor - 1) + 1 = 2$ blocks per query, which results in a Byzantine-robustness bound of $v = 10 - (5+2-1) - 2 = 2$ servers. This choice of parameters yields a two-fold increase in throughput when there are $v' \le 1$ Byzantine servers, or when the user seeks $B \ge 4$ blocks; moreover, the throughput is no worse than with single-block queries when there are $v' = 2$ Byzantine servers and the user seeks only $B < 4$ blocks.

It is trivial to modify the parameters from the above example to return additional blocks per query (at a cost of lower Byzantine robustness), or to get better Byzantine robustness (at a cost of returning fewer blocks per query). For example, again considering the case of $\ell = k = 10$ and $t = 5$, the user can set $q = 3$ blocks per query to get a robustness bound of $v = 1$ servers, which always yields a three-fold improvement in throughput compared to single-block queries (provided the query succeeds because there are only $v' \le 1$ Byzantine servers). Moreover, we point out that the user could pack an additional q' blocks into each query *without reducing robustness* by sharing her query among q' additional database servers. Doing so incurs an *additive* increase rather than a *multiplicative* increase in the communication cost of the query, and no change in per-sever computation cost. (That is, the communication cost increases from $\big(\ell+k\big)\sqrt{N}$ field elements to $\big(\ell+k+2q'\big)\sqrt{N}$ field elements instead of $q'\cdot\big(\ell+k\big)\sqrt{N}$ field elements.) However, this latter trick has the undesirable side effect of increasing the user's exposure to potentially malicious database servers without increasing the privacy threshold to compensate, so we do not explore it any further.

Our analysis so far suggests that multi-block queries can easily improve throughput compared to single-block queries by a factor of two or more, even for modest parameter choices. However, it turns out that the above analysis actually *understates* the expected throughput gains from switching to multi-block queries: it categorically overstates the need to maintain high robustness (thus, low throughput) for most queries, and assumes that both

[4]Preserving the Guruswami-Sudan robustness bound is a somewhat arbitrary goal, since we argue below that even less robustness should suffice if one accepts the premise that the PIR servers are all rational agents.

single-block and multi-block queries use a block size that provides optimal communication cost only for single-block queries. The remainder of this section discusses how one should select parameters to get optimal throughput with multi-block queries.

Making the case for tolerating reduced robustness. Good robustness with respect to Byzantine database servers is an immensely useful property for an IT-PIR protocol to possess; therefore, some remarks about why it is reasonable to tolerate reduced robustness are in order. Suppose that the user chooses q such that her query is robust against up to v Byzantine database servers. In this case, if she receives at most v incorrect responses, then not only can she recover her q requested blocks, *but she also learns which subset of database servers provided her with Byzantine responses*. Once the user catches some database server misbehaving, she can (and should) exclude it from her future queries. Thus, a coalition of malicious (but *rational*) database servers that wish to disrupt a user's queries should return incorrect responses if and only if the coalition is larger than the robustness threshold v for the given query. We already noted above, however, that single- and multi-block queries are information-theoretically indistinguishable by any coalition of up to t database servers; moreover, the non-collusion assumption states that larger coalitions do not exist among the database servers. Suppose that an "average" query can tolerate at most $\bar{v} < k-t-2$ Byzantine responses, but that some nontrivial fraction of queries are for just $q = 1$ blocks, and can therefore tolerate up to $v = k-t-2$ Byzantine responses. In this case, no rational coalition of malicious database servers should respond incorrectly to a query unless it has at least $k-t-1$ members that are involved in that query (in which case the coalition will *always* succeed in disrupting the query, regardless of q). Taking this line of reasoning to its logical extreme, users could choose q (hence, v) such that each query is just robust enough to tolerate the expected failure rate for honest servers, and only occasionally issue optimally robust queries to unmask members of coalitions whose size is at most $k-t-2$. We leave a more detailed game-theoretic analysis along these lines to future work.

In light of the above observations, we shall assume in our analyses throughout the remainder of the paper that the number of Byzantine servers v' is low enough that $m = 1$; that is, we will equate the cost of retrieving a record using a multi-block query with the cost of issuing a *single* multi-block query that requires no follow up queries. We reiterate that even if this assumption turns out to be false in practice (i.e., if v' is large enough to make $m > 1$), then once the user issues $m \leq v'$ queries, the cost per retrieved block is always as low as our optimistic analysis predicts.

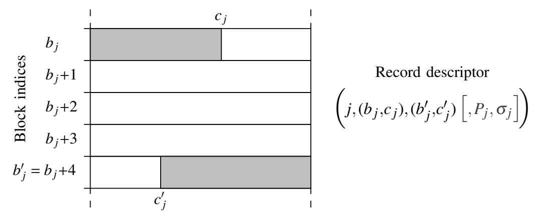


Figure 1: A record that spans five consecutive PIR blocks. The record has index j; it begins at the c_jth field element of block b_j and ends at the c'_jth field element of block $b'_j = b_j + 4$. In the descriptor for this record, P_j is an optional set of metadata about the record, such as information regarding its price or access criteria, and σ_j is an optional cryptographic signature on the rest of the tuple. Looking ahead to Section IV, Protocol 4 and the zero-knowledge proofs that extend Protocols 2 – 4 with pricing and access control use the optional values P_j and σ_j.

Selecting the communication-optimal block size. Suppose the largest record in the database $\mathbf{D}$ is S field elements long. For a fixed block size of s field elements, this longest record spans at most $B = \lceil (S-1)/s \rceil + 1$ blocks (cf. Figure 1). If each IT-PIR query can retrieve up to q blocks, then users must submit $Q = \lceil B/q \rceil$ queries to retrieve a record without leaking any information about that record to the database servers. Suppose that all servers respond to the user's queries (i.e., that $\ell = k$); the communication cost is therefore $2\ell \cdot Q\left(\lceil N/s \rceil + s\right)$ field elements for each record that the user retrieves, and the communication-optimal block size s is the positive integer that minimizes this expression. (It is clear that the *computation*-optimal block size is just any block size that minimizes Q; i.e., any block size for which $Q = 1$.)

Case 1 ($q > 1$): Clearly, we also have that $Q = 1$ for the communication-optimal block size since $Q\left(\lceil N/s \rceil + s\right) \geq \left(\lceil N/(sQ) \rceil + sQ\right)$, with equality holding if and only if $Q = 1$. (In other words, if a block size of s yields $Q > 1$, then switching the block size to $s' = sQ$ always reduces both communication *and* computation cost.) We therefore want the (positive integer) value of s that minimizes the sum $\lceil N/s \rceil + s$ subject to $Q = 1$. The smallest positive integer value of s for which $Q = 1$ is $s = \lceil (S-1)/(q-1) \rceil$; hence, the communication-optimal block size is $s = \max\left(\lceil (S-1)/(q-1) \rceil, \lceil \sqrt{N} \rceil\right)$.

Case 2 ($q = 1$): Because $B = \lceil (S-1)/s \rceil + 1 \geq 2$, it follows that $Q = \lceil B/q \rceil \neq 1$ in general; that is, without making some assumptions about the layout of the database, a single query does not suffice to retrieve an arbitrary record when $q = 1$. The "worst case" occurs when some block contains a single field element from a longest record, in which case $S - 1$ field elements from that record appear in other blocks. By setting $s \geq S - 1$ we can ensure that each of these other field elements are in the same block; therefore, the communication-optimal block size is $s = \max\left(S-1, \lceil \sqrt{N} \rceil\right)$, which always yields $Q = 2$.

In both of the above cases, the communication cost per query is $\Theta\big(\max\{S/q, \sqrt{N}\}\big)$. It may be possible to reduce communication costs by using data-dependent optimizations; i.e., by rearranging the records and introducing padding as appropriate to eliminate all unnecessary block overflows. The best possible communication cost occurs, for example, when all records have a fixed length $S \geq \sqrt{N}$, in which case setting $s = \lceil S/q \rceil$ with $Q = 1$ suffices for any q.

Communication-optimality of multi-block IT-PIR. Returning to our database-driven e-commerce example, wherein users seek to purchase e-books, music, movies, or smart phone and tablet apps from a PIR database, we note that PIR's linear computation requirement places a practical upper limit on the size of the database. Under reasonable assumptions about the database records, it follows for such applications that $(S-1)/(q-1) > \sqrt{N}$; i.e., that the actual communication cost is always $\Theta(S/q)$ in practice. For example, suppose the largest file in an online video store that supports up to $q \leq 3$ blocks per query is a movie occupying just 700 megabytes: the length of such a database would have to exceed *120 petabytes* before $(S-1)/(q-1) \leq \sqrt{N}$, which seems far too large for the linear computation cost of PIR to be economically feasible on modern hardware. In real-world e-commerce scenarios, we therefore expect the communication cost of multi-block IT-PIR to be within a factor of $(\ell+k)/(q-1)$ of optimal. As S increases relative to $\sqrt{N}$, this overhead factor approaches $k/(q-1)$.

IV. SPIR Constructions

This section presents four new SPIR constructions that extend Goldberg's IT-PIR. Each of our constructions supports variable-length records with the (multi-block) optimal communication cost. The first construction directly extends Goldberg's IT-PIR protocol into SPIR using ephemeral, record-level encryption and a key retrieval strategy inspired by Naor and Pinkas' oblivious polynomial evaluation protocol [32]. The second and third protocols replace ephemeral encryption with static encryption and use Camenisch et al.'s OT (or POT/OTAC) and Henry et al.'s SPIR (or PSPIR), respectively, to let users retrieve — perhaps by purchasing — the long-term decryption keys for the records they seek. The fourth protocol generalizes Henry et al.'s (P)SPIR protocol to support both *multi-block* and *sub-block* queries, thereby supporting queries for variable-length records in a plaintext PSPIR database. Each construction offers a different set of features and different performance and usability characteristics, all of which is summarized in Table 2 and Section IV-B.

A. Model

Our SPIR constructions build on the multi-block IT-PIR from Section III; thus, the model considers a set of ℓ independent database servers that each hold a complete replica of the (possibly encrypted) database. The user queries some subset of the database servers for a single database *record* (a unit of retrievable information), which may be larger than or smaller than a PIR *block* (a unit of data transfer). We use r and R to denote the number of (fixed-length) blocks and the number of (variable-length) records that comprise the database, respectively, and S to denote the length of the longest record (measured in field elements). The user is assumed to have *a priori* knowledge about the layout of any relevant portions of the database; i.e., we assume she knows the mapping between PIR blocks and database records, including the offsets into a block that specify where records begin and end (see Figure 1). How the user obtains this information is tangential to our own work; existing approaches like private keyword search [15], SQL queries [33], or trivial download of the entire index (which will typically be *much* smaller than the actual database) all suffice for this purpose. We assume that the block size is fixed to the communication-optimal size, as discussed at the end of Section III, whereas the sizes of individual records may vary independently from the block size and are dictated by their contents alone. In addition to the privacy and efficiency goals stated below, a secondary goal of our constructions is to enable the user to produce *efficient* zero-knowledge proofs about the record she is requesting in a given query. These proofs can state theorems such as "I have paid for this record" or "the access control list states that I should be allowed to access this particular record", for example. Indeed, attaching such zero-knowledge proofs to a query is straightforward in the latter three of our four constructions. To facilitate this, record descriptors (optionally) contain metadata about the record, and are signed under a cryptographic signature scheme that admits efficient zero-knowledge proofs of knowledge of a message-signature pair under a given public key (e.g., BBS+ signatures [1]).

Our protocols target the standard security requirements for SPIR:

User privacy: Queries must not leak any information about the particular records that a user requests, including the sizes of (i.e., the numbers of blocks that comprise) those records. More formally, we require that, for any honest user $\mathcal{U}$ requesting a record $A_i \in \{A_1, \ldots, A_R\}$, and for any coalition $\mathcal{C}$ of at most t dishonest database servers, the combined view of $\mathcal{C}$ upon receiving $\mathcal{U}$'s query is statistically independent of the record index i.

Data privacy: It must be infeasible for a user to learn nontrivial information about $n+1$ or more database records using only n SPIR queries. More formally,

	Protocol	Computation		Communication	Hardness assumptions
		field operations	exponentiations		
This work	1: Multi-block PIR + ephemeral encryption	$\Theta(N+\lg R)$	—	$\Theta(S/q+\lg R)$	SC, KDF
	2: Multi-block PIR + OT	$\Theta(N)$	$\Theta(1)$	$\Theta(S/q+R)$	SC, $(R{+}2)$-BDHE, $(R{+}1)$-SDH
	3: Multi-block PIR + SPIR	$\Theta(N+R)$	$\Theta(R)$	$\Theta(S/q+R)$	SC, $(t{+}q{-}1)$-SDH
	4: Multi-block SPIR	$\Theta(N+R)$	$\Theta(R)$	$\Theta(S/q+R)$	SC, KDF, $(t{+}q{-}1)$-SDH
	Camenisch et al.'s OT [12]	$\Theta(1)$	$\Theta(1)$	$2N+\Theta(1)$	$(R{+}2)$-BDHE, $(R{+}1)$-SDH
	Henry et al.'s (fixed-record length) SPIR [28]	$\Theta(N)$	$\Theta(\sqrt{N})$	$\Theta(\sqrt{N})$	t-SDH

Table 2: The "Computation" and "Communication" columns list the asymptotic computation cost (in field operations and exponentiations) and communication cost (in field elements), respectively, for each protocol considered in this paper: N is the length of the database, S is the length of the longest database record, q is the number of blocks per query, and R is the number of records. All exponentiations have $\approx$160-bit exponents for the user and $\approx$40-bit exponents for the database servers. With the exception of Protocol 1, each protocol uses $\Theta(1)$ pairing operations. For simplicity, the communication costs assume that $(S-1)/(q-1) \geq \sqrt{N}$. The "Hardness assumptions" column lists the computational hardness assumptions that must hold for data privacy: SC assumes a secure stream cipher, KDF assumes a secure key derivation function, BDHE is the bilinear Diffie-Hellman exponent assumption [7, §2.3], SDH is the strong Diffie-Hellman assumption [6, §2.3]. (User privacy is unconditional in each of the protocols.) To allow comparison with existing work, the last two rows list Camenisch et al.'s OT and Henry et al.'s SPIR.

we require that, for any (possibly dishonest) user $\mathcal{U}$ holding an arbitrary, auxiliary input α and interacting with honest database servers $\mathcal{DB}_1,\ldots,\mathcal{DB}_k$, for any set of descriptors $I=\{A_1,\ldots,A_R\}$ mapping records onto portions of an r-by-s matrix, for any $n<R$, and for any pair $(\mathbf{D}_1,\mathbf{D}_2)$ of r-by-s matrices that agree on a subset $I'=\{A_{i_1},\ldots,A_{i_n}\}$ of records described in I, the probability that $\mathcal{U}$ can, by issuing arbitrary queries to $\mathcal{DB}_1,\ldots,\mathcal{DB}_k$ for records in I', determine if $\mathcal{DB}_1,\ldots,\mathcal{DB}_k$ hold $\mathbf{D}_1$ or $\mathbf{D}_2$ is at most negligibly greater (in some suitable security parameter) than $1/2$.

In each of our protocols, the first property (user privacy) holds information-theoretically under the same noncollusion assumption that is required for user privacy in Goldberg's IT-PIR protocol, and the second property (data privacy) holds computationally; that is, the user is restricted to run polynomial time algorithms, but the database servers can be computationally unbounded. We point out the required computational assumptions in Table 2 and in the security analyses following each construction.

Protocol 1: Multi-block PIR + ephemeral encryption

Our first SPIR construction is a straightforward variant of Goldberg's IT-PIR in which the database servers use a stream cipher to encrypt each record on the fly with its own ephemeral pseudorandom key stream. The user first queries the (plaintext) database using a multi-block IT-PIR query to obtain a sequence of (ephemerally encrypted) blocks that span the desired record. Next, she uses OT to retrieve a seed that will allow her to reconstruct the key stream for (and thereby decrypt) only those parts of the retrieved blocks that correspond to a single record of her choosing. Under the assumption that the chosen stream cipher is computationally secure, the database servers do not need to check the validity of the user's IT-PIR query since the user learns nothing by requesting blocks that contain encrypted portions of other records (since she cannot reconstruct their respective key streams). Several aspects of this high-level protocol warrant some explanation.

Constructing the key streams. The ephemerally encrypted SPIR construction is instantiable with an arbitrary stream cipher, provided each database server has some way of computing the same per-record ephemeral key streams. We stress that it is crucial for the key streams to be truly unique and ephemeral; any key stream reuse can violate the data privacy requirement of the protocol. To accomplish this, we propose that each database server uses a shared secret key (which changes whenever the database changes) and some common state information (which changes after each query) to derive the ephemeral keys. Following Henry et al. [28], we point out that a cryptographic hash of the entire plaintext database is a suitable shared key, although any other cryptographically strong symmetric key would also work in practice. It is less straightforward to share a common, ephemeral state among the database servers because inter-server communication is undesirable. One simple way to securely get such a common state is to have the user designate an arbitrary database server as the *key master* for her query; the key master chooses a nonce ν and returns (ν,ς) to the user, where ς is a cryptographic signature on ν under the key master's long-term public key. The user transmits (ν,ς) to each database server as part of her query; once a server is convinced that the signature is valid, it uses ν as the "common state" from which to derive the ephemeral keys for the query. The database servers can process any query that contains a valid (ν,ς) pair without checking the freshness of ν; only the key master needs to take care to ensure that the ν it issues do not get reused (see below). (If desired, the database servers could use a common private key or a *group signature scheme* [14] to prevent the pair (ν,ς) from revealing the identity of the key master.)

Deriving and obtaining the ephemeral keys. Let $M = \lceil \lg R \rceil$ denote the number of bits in the binary representation of each record index, and let ξ and ν denote the shared symmetric key and shared state that the database servers will use to derive the set of ephemeral keys, respectively. We adapt a technique due to Naor and Pinkas [32] to enable the user to efficiently reproduce the ephemeral key stream for the record she is requesting. The database servers use a *key derivation function* (e.g., a keyed hash function) to derive a set of $2M$ symmetric keys $\{\kappa_{1_0}, \kappa_{1_1}, \ldots, \kappa_{M_0}, \kappa_{M_1}\}$ from ξ and ν. The key for the record at index $j = \sum_{i=1}^{M} b_i 2^{i-1}$ is then $K_j = \bigoplus_{i=1}^{M} \kappa_{i_{b_i}}$; here b_i denotes the i^{th} bit in the binary representation of j. The user retrieves K_j by issuing a sequence of M consecutive $\frac{1}{2}$OT queries to the key master for the necessary $\kappa_{i_{b_i}}$, and then computes K_j from the $\kappa_{i_{b_i}}$ using the above formula. The key master should respond to at most M such OT queries for $k_{i_{b_i}}$ corresponding to a given nonce ν (and every database server should refuse to respond to *any* OT queries for a nonce that it did not personally choose and sign); if so, the user can reconstruct at most one symmetric key K_j — and hence can decrypt at most one database record — per query.

Online database encryption. Encrypting each record by the ephemeral key streams is simple. Conceptually, each database server uses the output from the stream cipher to form an r-by-s matrix $\mathbf{K}$ over $\mathbb{F}$, which has the exact same layout as the database $\mathbf{D}$ (that is, if record j in $\mathbf{D}$ begins at column c_j of block b_j and ends at column c'_j of block b'_j , then the key stream produced by K_j also begins at column c_j of block b_j and ends at column c'_j of block b'_j in $\mathbf{K}$). Upon receiving a query (in the form of a vector of secret shares), the database servers each compute and return the product of their given share vectors with $\mathbf{D} + \mathbf{K}$. This is illustrated in Figure 2. In practice, the database servers never need to compute $\mathbf{K}$; rather, they can generate the elements of $\mathbf{K}$ on the fly as they compute the aforementioned product, thus using significantly less memory. The user reconstructs the encrypted record using Lagrange interpolation, just as in Goldberg's original protocol, and then uses K_j to reconstruct the key stream and decrypt the desired record.

$$\begin{bmatrix} e_j \\ \vdots \\ e_{j+q-1} \end{bmatrix} \cdot \left(\begin{bmatrix} \quad & \mathbf{D} & \quad \end{bmatrix} + \begin{bmatrix} \quad & \mathbf{K} & \quad \end{bmatrix} \right)$$

Figure 2: A multi-block IT-PIR query over the ephemerally encrypted database D. In practice, the q-by-s matrix of standard basis vectors on the left is shared using the ramp scheme variant of Shamir secret sharing in Section III and the key stream matrix K is never computed in full. The user follows up her IT-PIR query with $M = \lceil \lg R \rceil$ OT queries to retrieve the seed for one key stream in K.

Informal security and cost analysis. The above protocol information-theoretically hides the contents of the user's query under the same noncollusion assumptions as Goldberg's IT-PIR protocol, provided the chosen OT scheme also provides information-theoretic security for the user. (The OT scheme by Camenisch et al. [12] described in Section II-B, for example, provides the necessary information-theoretic protection for the user.) If the user is permitted to request at most M of the $\kappa_{i_{b_i}}$ from (only) the key master for a given nonce ν, then the user can reconstruct at most one key stream (assuming the stream cipher is secure); moreover, if the key master chooses its nonces uniformly at random from a suitable domain, then, with overwhelming probability (in the bit-length of the size of that domain), a secure key derivation function will never output the same ephemeral seed for two different records in two different queries. Similarly, since the shared secret ξ changes each time the contents of the database change, the user cannot replay ν to get two or more records encrypted under the same key stream as the database evolves. Hence, the protocol provides computational data privacy if it is instantiated with a secure stream cipher and a secure key derivation function, and if the database servers do not deviate from the protocol by reusing nonces or giving out too many $\kappa_{i_{b_i}}$. We justify this latter assumption regarding the correct operation of the database servers by pointing out that a database server who is willing to reuse nonces or permit superfluous OT queries for the $\kappa_{i_{b_i}}$ could just as easily give the user unfettered access to the plaintext database.

Both the communication cost and the computation cost of the protocol are dominated by the multi-block IT-PIR query, with some small additional overhead owing to encryption/decryption and the OT queries to fetch the seed for the key stream. The cost of this latter step scales with the logarithm of the number of records; thus, the total communication cost is $\Theta(\max\{\sqrt{N}, {}^{s}/_{q}\} + \lg R)$ field elements and the total computation cost is $\Theta(N + \lg R)$.

Unfortunately, there does not appear to be any straightforward way to augment this simple construction with additional features like pricing or access control. The next three constructions address this limitation.

Protocols 2 and 3: Multi-block PIR + (OT $\bigvee$ SPIR)

The next two protocols are very similar to one another, and are superficially quite similar to Protocol 1: the user retrieves part of the (encrypted) database using a multi-block IT-PIR query, and then uses (a variant of) either OT (in Protocol 2) or SPIR (in Protocol 3) to retrieve the appropriate decryption key for the record she seeks. At a high level, Protocol 2 is just a modified version of Camenisch et al.'s OT protocol in which the trivial download step is replaced by a multi-block IT-PIR query for the relevant portions of the database. This dramatically improves the communication cost of the protocol for large databases, at the cost of some additional computation and IT-PIR's noncollusion assumption. We note that such a tradeoff may be worthwhile in practice, since the (wall-clock) time required to query a large database with Goldberg's IT-PIR can be a few orders of magnitude lower than the time required for trivial download of that same database [34]; furthermore, since Moore's law predicts faster growth in parallel computation speeds than Nielsen's law predicts for bandwidth, this performance gap is likely to widen over time. Protocol 3 is substantially the same as Protocol 2, but it uses SPIR in place of OT to support a potentially different set of optional features at the cost of some additional computation and communication overhead.

Initializing the database. As in Protocol 1, we use a stream cipher to encrypt each record individually with its own key stream; however, instead of encrypting with ephemeral keys at query time, Protocol's 2 and 3 both use static (long-term) encryption keys in much the same way as Camenisch et al. do in their OT [12]. In the Camenisch et al. construction, the database $\mathbf{D}$ is represented by a length-N sequence of elements from the target group $\mathbb{G}_T$ of some admissible bilinear pairing $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$; each one of these N group elements corresponds to a single encrypted database record. (In other words, Camenisch et al. fix $r = N$ and $s = 1$.) Encryption and decryption are just multiplication and division, respectively, by some "pseudorandom" group element that acts as the symmetric key for a given record. In particular, the symmetric key for the record at index j is just a unique signature on the message "j" using a blinded version of Au et al.'s BBS+ signature scheme [1]. (In the case of POT or OTAC, the message also contains some metadata about record j.) To make Camenisch et al.'s approach work together with our multi-block IT-PIR and variable-length records, we represent the database as an r-by-s matrix over a finite field $\mathbb{F}$ rather than a length-N list of elements from $\mathbb{G}_T$. In the case of Protocol 2, the key stream that encrypts record j is output by a stream cipher seeded with the above unique signature. (Encryption and decryption are just componentwise addition and subtraction in $\mathbb{F}$ with the key stream output by this stream cipher, respectively.) To avoid index reuse in the case of a non-static database, we assign a globally unique identifier to each record and use this identifier in place of the record's index in the unique signature. In the case of Protocol 3, we also encrypt each record individually using a key stream output by some stream cipher; however, the seeds in Protocol 3 do not require any special structure (like being a signature on a specific value) and are therefore generated pseudorandomly.

Retrieving the decryption keys. In Protocol 2, the user retrieves her decryption key exactly as in Camenisch et al.'s OT [12] protocol, or possibly one of its variants like POT [10] or OTAC [9, 11] (with the above modification of replacing the record index by a globally unique identifier). Once she has obtained the seed, the user can use the stream cipher to reconstruct the key stream that decrypts her desired record. In Protocol 3, she similarly retrieves the appropriate seed for the stream cipher using Henry et al.'s SPIR [28] protocol.

Pricing and access control. Implementing pricing and access control on top of either protocol is trivial: simply use the built-in support for pricing and access control in Camenisch et al.'s POT and OTAC, respectively, or Henry et al.'s PSPIR, to restrict access to the decryption keys to those who have paid and satisfy the conditions set forth in the access control policy. In the case of Protocol 3, some additional features are available. For example, Henry et al.'s PSPIR protocol simultaneously supports both access control lists and what they call *tiered pricing*. Tiered pricing allows the database servers to partition users into groups (or *tiers*) and then set different prices for each tier. A user that queries the database proves in zero-knowledge that she satisfies the access control policy and that she is paying the price designated for members of her tier. Protocol 3 can also use the PSPIR protocol's bookkeeping functionality to support multiple payees and Top-K replication (see Henry et al.'s paper for details [28]). Camenisch et al. suggest that it might be possible to combine ideas from their OTAC and POT protocols together to get tiered pricing with access control in Protocol 2 as well [10], but it seems infeasible to implement bookkeeping functionality within their model.

Informal security and cost analysis. The security of Protocol 2 reduces to the security of Camenisch et al.'s protocol (modulo considerations about the use of a stream cipher), provided the noncollusion assumption for the IT-PIR holds. If the protocol is instantiated to use either Camenisch et al.'s POT [10] protocol or their OTAC [9] protocol to retrieve decryption keys, then user privacy is information-theoretic under the aforementioned non-collusion assumption, and data privacy is computational under the $(R+2)$-bilinear Diffie-Hellman exponent (BDHE) assumption [7, §2.3] and the $(R+1)$-strong Diffie-Hellman (SDH) assumption [6, §2.3]. The communication cost for the blind signature and accompanying zero-knowledge proof is in $\Theta(R)$, and the computation is in $\Theta(1)$; thus, the total communication cost per query is $\Theta(\max\{\sqrt{N}, {}^{S}\!/_{q}\} + R)$ field elements and the online computation cost is $\Theta(N)$ field operations.

Likewise, the security of Protocol 3 reduces to the security of Henry et al.'s PSPIR (again, modulo considerations about the use of a stream cipher). Hence, user privacy is information-theoretic under the aforementioned non-collusion assumption[5], and data privacy is computational under the $(t+q-1)$-SDH. Note that the database of keys is represented by an R-by-$\Theta(1)$ matrix over $\mathbb{F}$ instead of the communication-optimal square matrix for that scheme; thus, the communication cost and computation cost associated with obtaining a decryption key are both $\Theta(R)$. The total communication cost of the protocol is therefore $\Theta(\max\{\sqrt{N}, s/q\} + R)$ field elements and the total computation cost is $\Theta(N + R)$ field operations, plus $\Theta(R)$ full-length exponentiations (in an elliptic curve group) for the user and $\Theta(R)$ short exponentiations (i.e., with ≈ 40-bit exponents) for each database server.

Protocol 4: Multi-block SPIR

Our fourth and final construction generalizes Henry et al.'s PSPIR protocol [28] to handle multi-block queries, and adds online ephemeral encryption to handle sub-block queries (i.e, to handle queries involving blocks that contain parts of two or more distinct records). The result is a new PSPIR protocol that supports records of variable length in a plaintext database. The key building block of the protocol is Kate et al.'s polynomial commitments [29].

Constant-size commitments to polynomials. We briefly recall Kate et al.'s PolyCommit$_{\mathrm{DL}}$ polynomial commitments [29], which play a central role both in Henry et al.'s PSPIR protocol and in our generalization of it. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathrm{T}}$ be an admissible bilinear pairing on a group $\mathbb{G}$ of the same order as the prime field $\mathbb{F}$. A trusted initializer (or a distributed protocol) outputs a long-term public key $\mathsf{PK} = \{g^{\alpha^i} \mid 0 \le i \le T\}$, where g is a fixed generator of $\mathbb{G}$ and $\alpha \in \mathbb{Z}^*_{|\mathbb{F}|}$ is a secret (trapdoor) key, then it securely discards α. A commitment to a polynomial $f(x)$ is then $C = g^{f(\alpha)}$, which is easy to compute from PK provided $\deg f \le T$. A prover can open C to the evaluation of $f(x)$ at $x = i$ by computing the polynomial quotient $w(x)$ obtained upon dividing $f(x) - f(i)$ by $x - i$ and appealing to the polynomial remainder theorem, which states that $f(x) = w(x)(x-i) + f(i)$. That is, the prover computes a *witness* $\omega = g^{w(\alpha)}$ to the evaluation of $f(x)$ at $x = i$ and sends the triple $(i, f(i), \omega)$ to the verifier, who confirms that $e(C, g) = e(\omega, g^{\alpha}/g^{i}) \cdot e(g, g)^{f(i)}$. If $\deg f = t$, then C is information-theoretically hiding against an adversary that knows fewer than t evaluations of $f(x)$, and computationally hiding under the discrete logarithm assumption against an adversary that knows exactly t evaluations of $f(x)$ [29]. (C is trivially non-hiding against an adversary that knows more than t evaluations of $f(x)$, since such an adversary can easily interpolate said evaluations to compute $f(x)$.) C is computationally binding under the T-SDH assumption [29].

[5]Actually, user privacy is information-theoretic against coalitions of up to $t-1$ database servers, and computational under the discrete logarithm assumption against coalitions of t database servers [28].

Block types and metadata. To simplify the discussion in this section, it is convenient to distinguish between two fundamentally different block "types".

Type I blocks contain parts of two or more records.

Type II blocks contain information about only one record.

Every record is comprised of one or two Type I blocks and zero or more Type II blocks. Referring back to Figure 1, we see that record j in that example has two Type I blocks (indices b_j and b'_j) and three Type II blocks (indices $b_j + 1$ through $b_j + 3$).

Generalizing Henry et al.'s proof that a query is well formed. The crux of Henry et al.'s PSPIR protocol is an efficient zero-knowledge proof that a vector of polynomial commitments opens componentwise to a standard basis vector at $x = 0$. Given a fixed vector $\vec{f} = \langle f_1, \ldots, f_r \rangle$ of polynomials from $\mathbb{F}[x]$ and a uniform random vector $\vec{a} = \langle a_1, \ldots, a_r \rangle$ of scalars from $[1, n]$ ($n \le |\mathbb{F}|$), define the polynomial $F(x) = \vec{a} \cdot \vec{f}$. Henry et al.'s proof combines the following elementary observation about $F(x)$ with Cramer et al.'s proofs of partial knowledge [18] and Bellare et al.'s "small exponent" batch verification [4]: if $\langle f_1(0), \ldots, f_r(0) \rangle$ is a standard basis vector, then $F(0) = a_i$ for some $1 \le i \le r$; otherwise, if $\langle f_1(0), \ldots, f_r(0) \rangle$ is not a standard basis vector, then the probability that $F(0) = a_i$ for some $1 \le i \le r$ is at most $1/n$. To turn this observation into a zero-knowledge proof, the prover (in our case, the user) commits componentwise to $\vec{f}$ and sends the resulting vector of commitments $\vec{C} = \langle C_1, \ldots, C_r \rangle$ to the verifier (in our case, to each of the database servers). The verifier (or a random oracle if we use the Fiat-Shamir heuristic [22]) responds with a vector of short scalars ($\vec{a}$ above), and both parties use this challenge vector to compute the polynomial commitment $g^{F(\alpha)} = \prod_{i=1}^{r} C_i^{a_i}$ to $F(x) = \vec{a} \cdot \vec{f}$. Finally, the prover engages the verifier in a zero-knowledge proof of knowledge of y such that $y = F(0)$ and a (batch-verified) proof of partial knowledge of equality of discrete logarithms to prove she knows $i \in [1, r]$ such that $y = a_i$. (See Appendix B of Henry et al.'s paper [28] for further details.) If the verifier knows that sufficiently many other verifiers have accepted *the same proof about the same vector of commitments*, then it is trivial to extend the above zero-knowledge proof to show that a vector of r scalars from $\mathbb{F}$ corresponds to a componentwise secret sharing of a standard basis vector in $\mathbb{F}^r$. This latter step can be proved explicitly, for example, with a threshold signature, or implicitly, as we do below, by re-randomizing the query responses in a uniform way across all verifiers.

The above proof can be generalized from a proof that a vector of polynomial commitments opens to a standard basis vector at $x = 0$ to a proof that a vector of polynomial commitments opens to a q-by-r *matrix* from a predefined set at $x = 0, \ldots, q-1$. (Where the componentwise opening at

$x = i$ corresponds to row i of the matrix.) The generalized proof essentially uses q parallel instances of the vector proof, but it combines them together into a single batch. Thus, the verifier sends an additional length-q column vector $\vec{c}$ with uniform random components from $[1,n]$ along with the length-s row vector $\vec{a}$ after receiving $\vec{C}$ from the prover. Both parties then compute the q-by-r matrix A obtained by taking the outer product of $\vec{c}$ and $\vec{a}$. In the original protocol, the prover shows that the polynomial $F(x) = \vec{a} \cdot \vec{f}$ evaluates to a component of $\vec{a}$ at $x = 0$; in the generalized protocol, she instead shows that *the trace* of the matrix product of A with the transpose of the matrix encoded in $\vec{f}$ is in the set of such traces corresponding to allowable matrices. This is a natural way to generalize the protocol, since the trace of a product of matrices is itself a generalization of the notion of a dot product of vectors. Full details of the generalized protocol are in Appendix A.

Handling Type I blocks in a multi-block query. The above zero-knowledge proof is sufficient to extend Henry et al.'s construction to a restricted form of multi-block SPIR, in which users can request variable-length records from a database that has only Type II blocks. (To make this secure, the database servers still need to rerandomize the user's response polynomials for other inputs $x \geq q$; this can be done with a straightforward generalization of the analogous step in Henry et al.'s protocol.) To handle Type I blocks, we roll online ephemeral encryption into the query rerandomization step. Unlike in Protocol 1, the database servers do not encrypt the entire database; indeed, it is only necessary to encrypt Type I blocks, and even then only those particular Type I blocks that the user retrieves as part of her query. If the Type I blocks are always encoded in the first two rows of the query matrix (i.e., at $x = 0$ and $x = 1$ in the response polynomials), for example, then it suffices to encrypt *the response* — rather than the database — at $x = 0$ and $x = 1$ only, irrespective of which record the user requests. To enable this, the allowable matrix corresponding to record j should be structured as follows.

Case I ($b_j \neq b'_j$): Assume that record j begins in block b_j and ends in block $b'_j \neq b_j$ (we address the case of $b_j = b'_j$ separately below). The allowable matrix corresponding to a query for record j has e_{b_j} in its first row and $e_{b'_j - i+1}$ in its second row (the order of the other rows can be arbitrary, but must be agreed on by the user and all of the servers). If $b'_j - b_j < q - 1$, then the last $q - b'_j - b_j - 1$ rows of the matrix are the zero vector.

The servers need to encrypt blocks b_j and b'_j (which may or may not be Type I blocks) in the query response. To do so, they derive two common ephemeral seeds κ_0 and κ_1 using a key derivation function seeded with a common secret that changes whenever the database changes (for example, a digest of the database) and a common ephemeral state (for example, a digest of the commitment vector for the present query). Note that, although the user can replay a query with the same commitment vector, thus causing the same set of keys to be reused, this is not a threat to security since the user will also obtain the exact same set of encrypted records.

Next, the servers iteratively apply a (publicly computable) one-way function h (for example, a cryptographic hash function) to κ_0 and to κ_1 to produce two length-s key streams. The first key stream is $\vec{K}_0 = \langle h^{(1)}(\kappa_0), \ldots, h^{(s)}(\kappa_0) \rangle$ and the second key stream is $\vec{K}_1 = \langle h^{(s)}(\kappa_1), \ldots, h^{(1)}(\kappa_1) \rangle$, where $h^{(1)}(x) = h(x)$ and $h^{(i+1)}(x) = h(h^{(i)}(x)))$ for $1 < i \leq s$. (Note that the latter key stream has its order reversed.) Let T be the upper bound on the degree of a polynomial that can be committed to using the public parameters of the polynomial commitment scheme. The servers use a commonly seeded PRG to produce a length-s vector of degree-T polynomials that passes componentwise through $\vec{K}_0$ at $x = 0$, through $\vec{K}_1$ at $x = 1$, through the zero vector at each of $x = 2, \ldots, q - 1$, and through uniform random vectors at $x = q, \ldots, T - 1$.

Finally, each server evaluates this common polynomial vector componentwise at its own respective server index and adds the resultant vector of scalars to its query response. This encrypts the (potentially Type I) blocks encoded at $x = 0$ and $x = 1$, and randomizes the query response at $x \geq q$.

Note that, given $h^{(c_j)}(\kappa_0)$, it is easy to compute the trailing $s - c_j + 1$ components of $\vec{K}_0$, but infeasible to compute any of the leading $c_j - 1$ components (and similar reasoning holds for the leading components of $\vec{K}_1$). Thus, if the user learns $h^{(c_j)}(\kappa_0)$ and $h^{(s-c'_j+1)}(\kappa_1)$, then she can decrypt only the parts of the Type I blocks encoded at $x = 0$ and $x = 1$ in her query response. The user therefore requests these two values using Camenisch et al.'s [12] OT protocol, and attaches a zero-knowledge proof of knowledge of a signature (σ_j) on "$j, (b_j, c_j), (b'_j, c'_j) [, P_j]$" such that this message is consistent with the above PIR and OT queries.

Case II ($b_j = b'_j$): The above technique fails when $b_j = b'_j$ and $c'_j < s$ (that is, when the entire record fits within a single block but does not fill it through to the s^{th} field element), since in this case the user can also decrypt part of record $j + 1$. To solve this, the database servers (always) append a third length-s key stream $\vec{K}_2$ to the end of the database (as an extra, ephemeral $(r + 1)^{\text{th}}$ record generated from the same common inputs as $\vec{K}_0$ and $\vec{K}_1$). The allowable matrix corresponding to a query for record j then has $e_{b_j} + e_{r+1}$ in its first row and e_{r+1} in its second row (and the zero vector in rows $3, \ldots, q - 1$). It is straightforward to verify that, given $h^{(c_j)}(\kappa_0)$ and $h^{(s-c'_j+1)}(\kappa_1)$ as above, the user can still decrypt only the portion of row b_j that corresponds to record j. Note that the database servers cannot distinguish between Case I and Case II queries.

Pricing and access control. Adding support for pricing and access control within this framework is trivial, since the user already proves knowledge of a signature on a message that contains both the index of the record she is requesting and the metadata about this record. It is straightforward to adapt Henry et al.'s bookkeeping protocols to work with Protocol 4, although ambiguity may emerge among records that do not contain any Type II blocks. We leave a more thorough investigation of how one might eliminate this ambiguity to future work.

Informal security and cost analysis. User privacy in the above multi-block SPIR protocol is information-theoretic against coalitions of up to $t-1$ database servers and computational against coalitions of up to t database servers. If the one-way function and the key derivation function are both computationally secure, then data privacy follows from data privacy in Camenisch et al.'s OT protocol and the soundness of the zero-knowledge proof of query well-formedness, which holds with overwhelming probability under the $(t+q-1)$-SDH assumption. The total communication cost of the protocol is $\Theta(\max\{{}^{S}\!/_{q}, \sqrt{N}\}+R)$ and the computation cost is $\Theta(N)$ field operations, plus $\Theta(R)$ full-length exponentiations (in an elliptic curve group) for the user and $\Theta(R)$ short exponentiations (i.e., with ≈ 40-bit exponents) for each database server.

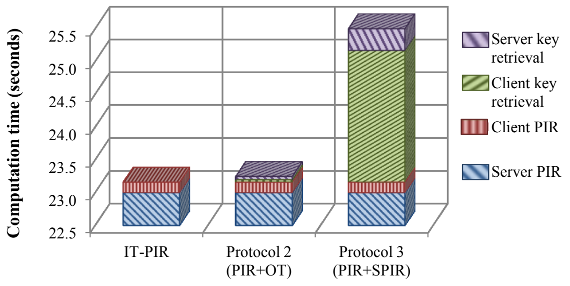
B. Discussion

We have constructed four new SPIR protocols that build on our multi-block variant of Goldberg's IT-PIR to allow SPIR queries over variable-length database records. Each protocol offers its own tradeoffs; this section briefly discusses these tradeoffs and discusses the relative merits of each protocol for different use cases.

Protocol 1 is the simplest extension of multi-block IT-PIR to SPIR in that it does not require any zero-knowledge proofs, and introduces only a slight modification to the underlying IT-PIR protocol. Unfortunately, the construction does not provide a straightforward way to integrate more sophisticated functionality like pricing and access control. Henry et al. have argued that the privacy-preserving e-commerce applications made possible by priced SPIR (in particular, their multiple-payee PSPIR [28]) may give rise to compelling arguments for the non-collusion assumption inherent in IT-PIR-based protocols; unfortunately, without a way to implement pricing, their argument does not apply to Protocol 1. Noticing certain parallels between our Protocol 1 and Camenisch et al.'s OT protocol — which *does* lend itself to integrating pricing or access control — we devised Protocol 2, which is essentially just Camenisch et al.'s protocol with the trivial download step replaced by a multi-block IT-PIR query. This latter change replaces the initial $\Theta(N)$ download step in Camenisch et al.'s protocol with a much smaller query-time download, which improves efficiency when the user only requires a relatively small subset of the database, or when the database contents are not static. In fact, Protocol 2 turns out to be more efficient than Protocol 1 because the encryption is static; the database servers need not coordinate efforts nor perform any online cryptographic operations. Protocol 2 is therefore the most efficient of our SPIR protocols with respect to computation cost. Protocol 3 is almost identical to Protocol 2, except it uses Henry et al.'s SPIR in place to Camenisch et al.'s OT; as such, Protocol 3 inherits the latter protocol's rich feature set, which includes simultaneous support for (tiered) pricing and access control lists, and novel bookkeeping features like Top-K replication and multiple payees (see Henry et al.'s [28] paper for further details on these features). This extra functionality comes at a cost of some extra communication and computation for the user to retrieve her decryption keys; the cost of this step scales with the number of records in the database. Whereas Protocols 2 and 3 combine multi-block IT-PIR with Camenisch et al.'s statically encrypted database approach, Protocol 4 adapts the ephemeral encryption strategy from our Protocol 1 to extend Henry et al.'s SPIR protocol to a variant that supports multi-block queries and variable-length records.

Protecting the privacy of content providers. Henry et al. introduced multiple-payee PSPIR to enable several, independent *content providers* to host each other's data in a common PSPIR database. In their model, the database servers periodically perform a secure multiparty computation to determine the portion of sales revenues to which each content provider is entitled. Under such a model, it seems reasonable to expect that content providers might want to keep their data private *from one another* in addition to from the users. Beimel and Stahl introduced the notion of τ-independence to address such situations [3]: a database is τ-independent if it is infeasible for coalitions of up to τ database servers to deduce nontrivial information about records that they did not themselves contribute to the database. (Of course, by temporarily taking on the role of user, the coalition can query the database for other content providers' records; τ-independence simply ensures that the coalition can only obtain their peers' records through the same "approved" channels as regular users.) It is easy to configure Goldberg's IT-PIR — and by extension, our multi-block variant thereof — with τ-independence. (See Goldberg's paper [25] for details on this feature, and how τ interacts with the other system parameters.) Henry et al. incorrectly state that their SPIR protocol is incompatible with this feature [28], but it is fortunately easy to verify that this is not the case. There is no difficulty in using the τ-independence from Goldberg's IT-PIR in any of our protocols, although in the case of Protocol 2 it turns out not to be particularly helpful to

Computation time for IT-PIR (over encrypted database) + OT/SPIR (for decryption keys)



	IT-PIR	Protocol 2 (PIR+OT)	Protocol 3 (PIR+SPIR)
Server PIR	23 000 ± 400 ms	23 000 ± 400 ms	23 000 ± 400 ms
Client PIR	159.2 ± 0.2 ms	159.2 ± 0.2 ms	159.2 ± 0.2 ms
Client key retrieval	—	37.6 ± 0.9 ms	2 010 ± 10 ms
Server key retrieval	—	51 ± 2 ms	583 ± 4 ms
Total	≈ 23 s	≈ 23.5 s	≈ 25.5 s

Figure 3: A histogram (left) and table (right) illustrating the total (wall-clock) computation time required to query a 16-gigabyte subset of the *Librivox Free Audio Book Collection* (`https://librivox.org/`) using multi-block IT-PIR, Protocol 2 (PIR+OT) and Protocol 3 (PIR+SPIR). The IT-PIR was instantiated with a word size of $w = 8$ bits, a block size of $s = 15.69$ megabytes, $\ell = 5$ servers, and a privacy threshold of $t = 2$ servers (so that user privacy is holds if a majority of database servers are honest); in each query, the user fetched $q = 3$ blocks. The block size was chosen to accommodate the largest record in our sample of the Librivox dataset, which was 31.4 megabytes. We performed 100 trials of each experiment; the table reports the mean ± the standard deviation across all trials. The total communication cost is under 79 megabytes for each protocol (dominated by each server sending a 15.69 megabyte block to the client) — a factor of approximately 2.5 higher than a non-private query for the largest audio book in the database.

protect the encrypted records with τ-independence since the protocol permits arbitrary IT-PIR queries over the encrypted database. On one hand, if each database server has all of the decryption keys, then the τ-independence does not prevent them from retrieving arbitrary records through unverified IT-PIR queries. On the other hand, if not every database server has a copy of the decryption keys, then the user must query the *content provider* with an OT query, which reveals much information about the particular record she seeks. In the case of Protocol 3, one can implement τ-independence *over the database of decryption keys* rather than over the encrypted records, and in Protocol 4 simply enabling τ-independence on the unencrypted database suffices. Therefore, in cases where several competing content providers wish to host their data in common database, the ability to enable τ-independence in Protocols 3 and 4 justifies their slightly higher cost.

C. Performance evaluation

Henry et al. argue that their PSPIR construction may be efficient enough for practical deployment in certain e-commerce applications [28]. Our new constructions offer substantial performance improvements over their protocol for databases serving multimedia content, which suggests that our own protocols might also be practical for deployment in such application domains. To verify that this is indeed the case, we have developed a fork of Percy++, an open-source implementation of Goldberg's IT-PIR protocol, in which we implement our multi-block query construction. We also implemented Camenisch et al.'s OT and leveraged the built-in support for Henry et al.'s SPIR in the latest release of Percy++ to implement Protocol 2 and Protocol 3, respectively. We used each of these three protocols to query for records in a 16-gigabyte sample of the *Librivox Free Audio Book Collection* (`https://librivox.org/`); Figure 3 presents a summary of our findings. In our experiments, the client and all servers ran on a single host with dual Intel Xeon E5420 CPUs (2.5 GHz) and sufficient RAM to hold the entire database in memory; therefore, the figures presented in Figure 3 are for computation only, and do not include I/O time. All times are wall-clock times and those listed for the servers were measured on a per-server basis. The most expensive steps (server PIR and client overhead in Protocol 3) are highly parallelizable, but our experiments were all single-threaded.

V. Conclusion

We revisited an observation first made by Henry et al. in the context of their multi-server SPIR protocol; i.e., that the user in Goldberg's IT-PIR protocol can retrieve several blocks with a single query by replacing Shamir secret sharing with its ramp scheme variant. We pointed out how to take advantage of this observation to implement multi-block queries that trade off some Byzantine robustness for improved throughput without affecting privacy. Our multi-block IT-PIR queries are information-theoretically indistinguishable from standard, single-block queries when the number of colluding database servers does not exceed the privacy threshold. By taking advantage of the recent Cohn-Heninger multi-polynomial list decoding algorithm, we demonstrated that the user can retrieve several blocks with the same communication and computation cost as a single-block query and studied the impact of multi-block queries on Byzantine robustness. We found that our new approach can significantly reduce the constant factor separating the communication cost of Goldberg's original protocol from optimal when the database hosts realistically sized records.

With our new multi-block IT-PIR protocol as a starting point, we constructed four new SPIR protocols that each support variable-length database records. By decoupling the PIR block size from the lengths of individual database records, our new protocols can use the communication-optimal block size without artificially restricting the contents and layout of the records. Moreover, we pointed out how straightforward extensions to three of our four new SPIR constructions make it possible to construct efficient zero-knowledge proofs about the particular records a user is requesting in a given query; this makes it easy to implement pricing and access control structures over the records using standard techniques from the literature. The resulting SPIR protocols are therefore well suited to privacy-preserving e-commerce applications, such as privacy-friendly sales of e-books, music, movies, or smart phone and tablet apps.

Acknowledgements.. We thank the anonymous reviewers for their thoughtful and constructive comments. Funding for this research was provided in part by NSERC, Mprime NCE (formerly MITACS) and the Ontario Research Fund. The first author is supported by an NSERC Vanier Canada Graduate Scholarship and a Cheriton Graduate Scholarship.

References

[1] M. H. Au, W. Susilo, and Y. Mu, "Constant-Size Dynamic k-TAA," in *Proceedings of SCN 2006*, ser. LNCS, vol. 4116, Maiori, Italy, Sept 2006, pp. 111–125.

[2] A. Beimel, Y. Ishai, and T. Malkin, "Reducing the Servers' Computation in Private Information Retrieval: PIR with Preprocessing," *Journal of Cryptology*, vol. 17, no. 2, pp. 125–151, Mar 2004.

[3] A. Beimel and Y. Stahl, "Robust Information-Theoretic Private Information Retrieval," *Journal of Cryptology*, vol. 20, no. 3, pp. 295–321, Jul 2007.

[4] M. Bellare, J. A. Garay, and T. Rabin, "Fast Batch Verification for Modular Exponentiation and Digital Signatures," in *Proceedings of EUROCRYPT 1998*, ser. LNCS, vol. 1403, Espoo, Finland, Jun 1998, pp. 236–250.

[5] G. R. Blakley and C. Meadows, "Security of Ramp Schemes," in *Proceedings of CRYPTO 1984*, ser. LNCS, vol. 196, Santa Barbara, California, Aug 1984, pp. 242–268.

[6] D. Boneh and X. Boyen, "Short Signatures Without Random Oracles," in *Proceedings of EUROCRYPT 2004*, ser. LNCS, vol. 3027, Interlaken, Switzerland, May 2004, pp. 56–73.

[7] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical Identity Based Encryption with Constant Size Ciphertext," in *Proceedings of EUROCRYPT 2005*, ser. LNCS, vol. 3494, Aarhus, Denmark, May 2005, pp. 440–456.

[8] G. Brassard, C. Crépeau, and J.-M. Robert, "All-or-Nothing Disclosure of Secrets," in *Proceedings of CRYPTO 1986*, ser. LNCS, vol. 263, Santa Barbara, California, Aug 1986.

[9] J. Camenisch, M. Dubovitskaya, and G. Neven, "Oblivious Transfer with Access Control," in *Proceedings of ACM CCS 2009*, Chicago, Illinois, Nov 2009, pp. 131–140.

[10] ——, "Unlinkable Priced Oblivious Transfer with Rechargeable Wallets," in *Proceedings of FC 2010*, ser. LNCS, vol. 6052, Tenerife, Canary Islands, Jan 2010, pp. 66–81.

[11] J. Camenisch, M. Dubovitskaya, G. Neven, and G. M. Zaverucha, "Oblivious Transfer with Hidden Access Control Policies," in *Proceedings of PKC 2011*, ser. LNCS, vol. 6571, Taormina, Italy, Mar 2011, pp. 192–209.

[12] J. Camenisch, G. Neven, and abhi shelat, "Simulatable Adaptive Oblivious Transfer," in *Proceedings of EUROCRYPT 2007*, ser. LNCS, vol. 4515, Barcelona, Spain, May 2007, pp. 573–590.

[13] D. Chaum, R. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, A. T. Sherman, and P. L. Vora, "Scantegrity II: End-to-End Verifiability by Voters of Optical Scan Elections Through Confirmation Codes," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 4, pp. 611–627, Dec 2009.

[14] D. Chaum and E. van Heyst, "Group Signatures," in *Proceedings of EUROCRYPT 1991*, ser. LNCS, vol. 547, Brighton, UK, Apr 1991, pp. 257–265.

[15] B. Chor, N. Gilboa, and M. Naor, "Private Information Retrieval by Keywords," Technion, Israel, Technical report TR CS0917, 1997.

[16] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private Information Retrieval," *Journal of the ACM*, vol. 45, no. 6, pp. 965–981, Nov 1998.

[17] H. Cohn and N. Heninger, "Approximate Common Divisors via Lattices," in *Proceedings of the 10th Algorithmic Number Theory Symposium — ANTS X*, July 2012.

[18] R. Cramer, I. Damgård, and B. Schoenmakers, "Proofs of Partial Knowledge and Simplified Design of Witness Hiding Protocols," in *Proceedings of CRYPTO 1994*, ser. LNCS, vol. 839, Santa Barbara, California, Aug 1994, pp. 174–187.

[19] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a Type III Anonymous Remailer Protocol," in *Proceedings of IEEE S&P 2003*, Oakland, California, May 2003, pp. 2–15.

[20] C. Devet, I. Goldberg, and N. Heninger, "Optimally Robust Private Information Retrieval," in *Proceedings of USENIX Security 2012*, Bellevue, Washington, Aug 2012.

[21] R. Dingledine, N. Mathewson, and P. F. Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of USENIX Security 2004*, San Diego, California, Aug 2004, pp. 303–320.

[22] A. Fiat and A. Shamir, "How to Prove Yourself: Practical Solutions to Identification and Signature Problems," in *Proceedings of CRYPTO 1986*, ser. LNCS, vol. 263, Santa Barbara, California, Aug 1986, pp. 186–194.

[23] Y. Gertner, S. Goldwasser, and T. Malkin, "A Random Server Model for Private Information Retrieval or How to Achieve Information Theoretic PIR Avoiding Database Replication," in *Proceedings of RANDOM 1998*, Barcelona, Spain, Oct 1998, pp. 200–217.

[24] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin, "Protecting Data Privacy in Private Information Retrieval Schemes," in *Proceedings of STOC 1998*, Dallas, Texas, May 1998, pp. 151–160.

[25] I. Goldberg, "Improving the Robustness of Private Information Retrieval," in *Proceedings of IEEE S&P 2007*, Oakland, California, May 2007, pp. 131–148.

[26] V. Guruswami and M. Sudan, "Improved Decoding of Reed-Solomon and Algebraic-Geometry Codes," *IEEE Transactions on Information Theory*, vol. 45, no. 6, pp. 1757–1767, Sept 1999.

[27] R. Henry and I. Goldberg, "Batch Proofs of Partial Knowledge," University of Waterloo, Technical report CACR 2012-04, 2012.

[28] R. Henry, F. Olumofin, and I. Goldberg, "Practical PIR for Electronic Commerce," in *ACM CCS 2011*, Chicago, Illinois, Oct 2011, pp. 677–690.

[29] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-Size Commitments to Polynomials and Their Applications," in *Proceedings of ASIACRYPT 2010*, ser. LNCS, vol. 6477, Singapore, Dec 2010, pp. 177–194.

[30] E. Kushilevitz and R. Ostrovsky, "Replication is Not Needed: Single Database Computationally-Private Information Retrieval," in *Proceedings of FOCS 1997*, Miami Beach, Florida, Oct 1997, pp. 364–373.

[31] C. A. Melchor and P. Gaborit, "A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol," in *Proceedings of WEWORC 2007*, Bochum, Germany, Jul 2007.

[32] M. Naor and B. Pinkas, "Oblivious Transfer and Polynomial Evaluation," in *Proceedings of STOC 1999*, Atlanta, Georgia, May 1999, pp. 245–254.

[33] F. G. Olumofin and I. Goldberg, "Privacy-Preserving Queries over Relational Databases," in *Privacy Enhancing Technologies*, ser. LNCS, vol. 6205, Berlin, Germany, Jul 2010, pp. 75–92.

[34] ——, "Revisiting the Computational Practicality of Private Information Retrieval," in *Proceedings of FC 2011*, ser. LNCS, vol. 7035, Gros Islet, St. Lucia, Feb 2011, pp. 158–172.

[35] P. Y. A. Ryan and S. A. Schneider, "Prêt à Voter with Re-encryption Mixes," in *Proceedings of ESORICS 2006*, ser. LNCS, vol. 4189, Hamburg, Germany, Sept 2006, pp. 313–326.

[36] A. Shamir, "How to Share a Secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, Nov 1979.

[37] R. Sion and B. Carbunar, "On the Practicality of Private Information Retrieval," in *Proceedings of NDSS 2007*, San Diego, California, Mar 2007.

[38] S. Yekhanin, "New Locally Decodable Codes and Private Information Retrieval Schemes," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 13, no. 127, Oct 2006.

Appendix: ZKP from Protocol 4

Let $\vec{f} = \langle f_1, \ldots, f_r \rangle$ be a vector of degree (at most) $T = t + q$ polynomials in $\mathbb{F}[x]$, let PK be public parameters for degree-T polynomial commitments, let $\vec{C} = \langle C_1, \ldots, C_r \rangle$ be a componentwise vector of polynomial commitments to the components of $\vec{f}$, and let $\mathbf{A} \subseteq \mathbb{F}^{q \times r}$ be a set of "allowable" q-by-r matrices over $\mathbb{F}$; in the SPIR setting, each matrix in $\mathbf{A}$ will describe the layout of one of the records the user may retrieve. The tuple $(t, q, \vec{C}, \mathbf{A}, \mathsf{PK})$ is common input to both the prover (P) and the verifier (V), but only P knows $\vec{f}$. The following protocol allows P to convince V that

$$M = \begin{bmatrix} f_1(1) & \cdots & f_r(1) \\ \vdots & \ddots & \vdots \\ f_1(q) & \cdots & f_r(q) \end{bmatrix} \in \mathbf{A}; \tag{1}$$

i.e., that there exists an allowable matrix A such that, for all $i \in [1, \ldots, q]$, $\vec{f}$ evaluates componentwise to the ith row of A on input $x = i$.

V: Choose $\vec{a} \in_{\mathrm{R}} \left[0, n-1\right]^{1 \times r}$ and send it to P.

P: Choose $\gamma \in_{\mathrm{R}} \mathbb{Z}^*_{|\mathbb{F}|}$ and a generator $h \in_{\mathrm{R}} \mathbb{F}$, then compute the polynomial $F(x) = \gamma\big(\vec{f} \cdot \vec{a}\big)$. Compute $h' = h^\gamma$ and the (polynomial commitment) witnesses $\omega_1, \ldots, \omega_q$ to the evaluations of $F(x)$ at $x = 1, \ldots, q$, and send the tuple $(h, h', \omega_1, \ldots, \omega_q)$ to V.

V: Choose $\vec{c} \in_{\mathrm{R}} \left[0, n-1\right]^{q \times 1}$ and send it to P.

P and V: Compute the q-by-r challenge matrix $B = \vec{c} \otimes \vec{a}$, then construct the set of scalars $\mathbf{B} = \{\mathrm{Tr}(BA^{\mathrm{T}}) \mid A \in \mathbf{A}\}$.

P: Compute and send the commitment $\mathcal{C}' = g^{\sum_{j=1}^{q} c_j F(\alpha)}$ to V.

V: Compute $\mathcal{C} = \left(\prod_{i=1}^{r} \mathcal{C}_i^{a_i}\right)^{\sum_{j=1}^{q} c_j}$, $\Omega = \prod_{j=1}^{q} \omega_j^{c_j}$, $J = \prod_{j=1}^{q} \omega_j^{j c_j}$, and $Y = e(\mathcal{C}' \cdot J, g) / e(\Omega, g^\alpha)$.

P and V: Compute $g_{\mathrm{T}} = e(g, g)$ and engage in the following zero-knowledge proof of knowledge to show that $\log_{g_{\mathrm{T}}} Y \in \gamma \mathbf{B}$:

$$\mathrm{PK}\Big\{(\gamma) : \mathcal{C}' = \mathcal{C}^\gamma \wedge h' = h^\gamma \wedge \big(\textstyle\bigvee_{B \in \mathbf{B}} \log_{g_{\mathrm{T}}} Y = \gamma B\big)\Big\}.$$

This latter proof can be efficiently implemented using Henry and Goldberg's batch proofs of partial knowledge [27].

Lemma 1: Fix a list $\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_m$ of distinct vectors in $\mathbb{F}^r$. Then for $\vec{a} \in_{\mathrm{R}} \left[0, n-1\right]^r$, $\Pr\left[\exists i \in [1, m],\ \vec{a} \cdot \vec{v}_0 = \vec{a} \cdot \vec{v}_i\right] \leq {}^m/_n$, where the probability is taken over the random choices of $\vec{a}$.

Proof: Consider the list of distinct nonzero vectors $\vec{v}'_1, \ldots, \vec{v}'_m$ obtained by setting $\vec{v}'_i = \vec{v}_i - \vec{v}_0$ for $i \in [1, m]$. We will show that $\Pr\left[\vec{a} \cdot \vec{v}'_i = 0 \mid \vec{a} \in_{\mathrm{R}} \left[0, n-1\right]^r\right] \leq {}^1/_n$; the statement to be proved then follows from the union bound.

Let v_{ij} denote the jth component of $\vec{v}'_i$. Fix an index $j' \in [1, m]$ such that $v_{ij'} \neq 0$ and define the sum $V_{ij'} = \sum_{j \in [1,m] - j'} a_j v_{ij}$. Now, $\vec{a} \cdot \vec{v}'_i = 0$ is equivalent to $a_{j'} v_{ij'} = -V_{ij'}$; hence, $a_{j'} = {}^{-V_{ij'}}/_{v_{ij'}}$. Note that the right-hand side of this equation is independent from $a_{j'}$; hence, because each of the possible values for $a_{j'}$ occurs with equal probability, this happens with probability at most ${}^1/_n$. ■

Corollary 2: Fix a matrix $M \in \mathbb{F}^{q \times r}$ and set of matrices $\mathbf{A} \subseteq \mathbb{F}^{q \times r}$. For randomly chosen vectors $\vec{a} \in_{\mathrm{R}} \left[0, n-1\right]^{1 \times r}$ and $\vec{c} \in_{\mathrm{R}} \left[0, n-1\right]^{q \times 1}$, define the matrix $B = \vec{c} \otimes \vec{a} \in \mathbb{F}^{q \times r}$ and the set of traces $\mathbf{B} = \{\mathrm{Tr}(BA^{\mathrm{T}}) \mid A \in \mathbf{A}\}$. Then

$$\Pr\left[\mathrm{Tr}(BM^{\mathrm{T}}) \in \mathbf{B} \mid M \in \mathbf{A}\right] = 1, \text{ and} \tag{2}$$

$$\Pr\left[\mathrm{Tr}(BM^{\mathrm{T}}) \in \mathbf{B} \mid M \notin \mathbf{A}\right] \leq {}^{2|\mathbf{A}|}/_n, \tag{3}$$

where both probabilities are over the random choices of $\vec{a}$ and $\vec{c}$.

Proof: Equation (2) is immediate from the definition of $\mathbf{B}$. To prove Equation (3), observe that for any $A \in \mathbf{A} \cup \{M\}$ we can rewrite $\mathrm{Tr}(B \cdot A^{\mathrm{T}}) = \mathrm{Tr}(\vec{c} \cdot \vec{a} \cdot A^{\mathrm{T}}) = \mathrm{Tr}(\vec{a} \cdot A^{\mathrm{T}} \cdot \vec{c}) = \vec{a} \cdot (A^{\mathrm{T}} \cdot \vec{c})$, which is a dot product of two length-r vectors. We use Lemma 1 to prove

$$\Pr\left[\exists A \in \mathbf{A},\ M^{\mathrm{T}} \cdot \vec{c} = A^{\mathrm{T}} \cdot \vec{c}\right] \leq {}^{|\mathbf{A}|}/_n \tag{4}$$

for $\vec{c} \in_{\mathrm{R}} \left[0, n-1\right]^{q \times 1}$; the result then follows from another application of Lemma 1.

Consider a fixed matrix $A \in \mathbf{A}$ and index j such that the jth columns of A and M differ in at least one row. It then follows from Lemma 1 that the jth components of the vectors $A^{\mathrm{T}} \cdot \vec{c}$ and $M^{\mathrm{T}} \cdot \vec{c}$ are different with probability at least ${}^1/_n$. Equation (4) then follows from the union bound. ■

Theorem 3: The above protocol is a zero-knowledge proof that Equation (1) holds; i.e., the protocol is complete, sound (with overwhelming probability in $\lg n$), and simulatable.

Proof: To prove completeness, suppose Equation (1) holds so that $\gamma\,\mathrm{Tr}(BM^{\mathrm{T}}) = \sum_{j=1}^{q} c_j F(j) \in \gamma \mathbf{B}$. If $\omega_1, \ldots, \omega_q$ are correct witnesses for evaluations of $F(x)$ at $x = 1, \ldots, q$, then it is straightforward to verify that $e\big((\prod_{i=1}^{r} \mathcal{C}_i^{a_i})^\gamma, g\big) = e(\omega_j, g^\alpha / g^j)\, e(g, g)^{F(j)}$ for $j \in [1, q]$. Thus, since $e(\mathcal{C}', g) = \prod_{j=1}^{q} e\big((\prod_{i=1}^{r} \mathcal{C}_i^{a_i})^\gamma, g\big)^{c_j}$ and

$$\begin{aligned}
\prod_{j=1}^{q} \big(e(\omega_j, g^\alpha / g^j) \cdot e(g, g)^{F(j)}\big)^{c_j} \\
&= \prod_{j=1}^{q} \big(e(\omega_j, g^\alpha) / e(\omega_j, g^j)\big)^{c_j} \cdot e(g, g)^{\sum_{j=1}^{q} c_j F(j)} \\
&= \prod_{j=1}^{q} \big(e(\omega_j, g^\alpha) / e(\omega_j^j, g)\big)^{c_j} \cdot e(g, g)^{\gamma\,\mathrm{Tr}(BM^{\mathrm{T}})} \\
&= \big(e(\Omega, g^\alpha) / e(J, g)\big) \cdot e(g, g)^{\gamma\,\mathrm{Tr}(BM^{\mathrm{T}})},
\end{aligned}$$

it follows that $g_{\mathrm{T}}^{\gamma\,\mathrm{Tr}(BM^{\mathrm{T}})} = e(\mathcal{C}' \cdot J, g) / e(\Omega, g^\alpha)$; hence, completeness follows from the completeness of the ZKP in the final step.

Soundness follows from Corollary 2 and the soundness of the zero-knowledge proof of knowledge in the final step.

Given a simulator $\mathcal{S}$ for the ZKPoK in the final step, constructing a simulator $\mathcal{S}'$ for the entire protocol is trivial; i.e., $\mathcal{S}'$ chooses $\mathcal{C}', h', \omega_1, \ldots, \omega_q$ at random, then invokes $\mathcal{S}$ on these (and the common) inputs. ■