Single-Server Private Information Retrieval with Sublinear Amortized Time

Henry Corrigan-Gibbs[1], Alexandra Henzinger[1(✉)], and Dmitry Kogan[2]

[1] MIT, Cambridge, MA, USA
{henrycg,ahenz}@csail.mit.edu
[2] Fordefi, Tel Aviv, Israel
dkogan@cs.stanford.edu

Abstract. We construct new private-information-retrieval protocols in the single-server setting. Our schemes allow a client to privately fetch a sequence of database records from a server, while the server answers each query in average time sublinear in the database size. Specifically, we introduce the first single-server private-information-retrieval schemes that have sublinear amortized server time, require sublinear additional storage, and allow the client to make her queries adaptively. Our protocols rely only on standard cryptographic assumptions (decision Diffie-Hellman, quadratic residuosity, learning with errors, etc.). They work by having the client first fetch a small "hint" about the database contents from the server. Generating this hint requires server time linear in the database size. Thereafter, the client can use the hint to make a bounded number of adaptive queries to the server, which the server answers in sublinear time—yielding sublinear amortized cost. Finally, we give lower bounds proving that our most efficient scheme is optimal with respect to the trade-off it achieves between server online time and client storage.

1 Introduction

A private-information-retrieval protocol [34,35] allows a client to fetch a record from a database server without revealing which record she has fetched. In the simplest setting of private information retrieval, the server holds an n-bit database, the client holds an index $i \in \{1, \ldots, n\}$, and the client's goal is to recover the i-th database bit while hiding her index i from the server.

Fast protocols for private information retrieval (PIR) would have an array of applications. Using PIR, a student could fetch a book from a digital library without revealing to the library which book she fetched. Or, she could stream a movie without revealing which movie she streamed. Or, she could read an online news article without revealing which article she read. More broadly, PIR is at the heart of a number of systems for metadata-hiding messaging [7,32], privacy-preserving advertising [8,60,70,88], private file-sharing [40], private e-commerce [66], private media-consumption [62], and privacy-friendly web browsing [72].

Unfortunately, the *computational cost* of private information retrieval is a barrier to its use in practice. In particular, to respond to each client's query,

O. Dunkelman and S. Dziembowski (Eds.): EUROCRYPT 2022, LNCS 13276, pp. 3–33, 2022.
https://doi.org/10.1007/978-3-031-07085-3_1

Beimel, Ishai, and Malkin [14] showed that the running time of a PIR server must be at least linear in the size of the database. This linear-server-time lower bound holds even if the client communicates with many non-colluding database replicas. So, for a client to privately fetch a single book from a digital library, the library's servers would have to do work proportional to the total length of all of the books in the library, which is costly both in theory and in practice.

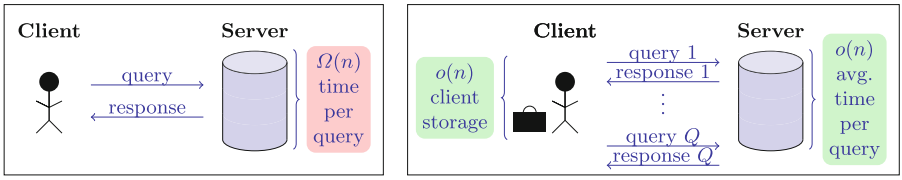
Towards reducing the server-side cost of PIR, a number of prior works [7,39, 64,68,79] observe that clients in many applications of PIR will make a sequence of queries to the same database. For example, a student may browse many books in a library; a web browser makes many domain name system (DNS) queries on each page load [80]; a mail client may check all incoming URLs against a database of known phishing websites [16,72]; or, an antivirus software may check the hashes of executed files against known malware [72]. The lower bound of Beimel, Ishai, and Malkin [14] only implies that a PIR server will take linear time to respond to the client's very first PIR query. This leaves open the possibility of reducing the server-side cost for subsequent queries. In other words, in the multi-query setting, we can hope for the *amortized* server-side time per query to be sublinear in the database size.

Indeed, there exist an array of techniques for constructing PIR schemes with sublinear amortized server-side cost. Yet, prior PIR schemes achieving sublinear amortized time come with limitations that make them cumbersome to use in practice. Schemes that require multiple non-colluding servers [39,72,89] demand careful coordination between many business entities, which is a major practical annoyance [4,15,81,90]. In addition, the security of these schemes is relatively brittle, since it relies on an adversary not being able to compromise multiple servers, rather than on cryptographic hardness. Recent offline/online PIR schemes [39,72,89] require, in the single-server setting, the server to perform a linear-time preprocessing step *for each query*. Thus, these schemes cannot have sublinear amortized time. Batch-PIR schemes [7,64,68,79], which require the client to make all of her queries at once, in a single non-adaptive batch, do not apply to many natural applications (e.g., digital library, web browsing), in which the client decides over time which elements she wants to query.

The world of private-information-retrieval is thus in an undesirable state: the practical applications are compelling, but existing schemes cannot satisfy the deployment demands (single server, adaptive queries, small storage, based on implementable primitives) while avoiding very large server-side costs.

1.1 Our Results

This paper aims to advance the state of the art in private information retrieval by introducing the first PIR schemes that simultaneously offer a number of important properties for use in practice: they require only a single database server, they have sublinear amortized server time, they allow the client to issue its database queries adaptively, and they require extra storage sublinear in the database size (Fig. 1). Our schemes further rely only on standard cryptographic primitives and incur no additional server-side (per client) storage, making them attractive even



(a) Standard single-server PIR [74].

(b) *This work*: Single-server, many-query PIR with sublinear amortized time and storage.

Fig. 1. Comparison of single-server PIR models, on database size n.

when many clients query a single database. One limitation of our schemes is that they require more client-side storage and computation than standard PIR schemes, though we give lower bounds showing that some of these costs are inherent to achieving sublinear amortized server time. While the schemes in this paper may not yet be concretely efficient enough to use in practice, they demonstrate that sublinear-amortized-time single-server PIR is theoretically feasible. We hope that future work pushes PIR even closer to practice.

Specifically, in this paper we construct two new families of PIR schemes:

Single-Server PIR with Sublinear Amortized Time from Linearly Homomorphic Encryption. First, we show in Theorem 16 that any one of a variety of standard assumptions—including quadratic residuosity, decision Diffie-Hellman, decision composite residuosity, and learning with errors—suffices to construct single-server PIR schemes with sublinear amortized time. In particular, on database size n, if the client makes at least $n^{1/4}$ adaptive queries, our schemes have: amortized server time $n^{3/4}$, amortized communication complexity $n^{1/2}$, client storage $n^{3/4}$, and amortized client time $n^{1/2}$. (When describing protocol costs in this section, we hide both $\log n$ factors and polynomials in the security parameter.) More generally, the existence of linearly homomorphic encryption with sufficiently compact ciphertexts and standard single-server PIR with polylogarithmic communication together imply the existence of our PIR schemes. Our client-side costs are much larger than those required for standard stateless PIR—which needs no client storage and requires client time polylogarithmic in the database size. Our schemes thus reduce server-side costs at some expense to the client.

Single-Server PIR with Sublinear Amortized Time and an Optimal Storage/Online-Time Trade-Off from Fully Homomorphic Encryption. Next, we show in Theorem 19 that under the stronger assumption that fully homomorphic encryption exists, we can construct PIR schemes with even lower amortized server time and client storage. In particular, we construct a PIR scheme that on database size n, and as long as the client makes at least $n^{1/2}$ queries, has amortized server time $n^{1/2}$, amortized communication complexity $n^{1/2}$, client storage $n^{1/2}$, and amortized client time $n^{1/2}$. (In contrast, from linearly homomorphic encryption, we get schemes with larger server time and client storage $n^{3/4}$.)

Lower Bounds on Multi-query PIR. Finally, we give new lower bounds on PIR schemes in the amortized (i.e., multi-query) setting. We give one lower bound

against PIR schemes that allow the client to make its queries adaptively, and another against schemes that require the client to make its queries in a non-adaptive batch. In the *adaptive* setting, we show in Theorem 21 that any multi-query PIR scheme on database size n in which: the client stores S bits between queries, the server stores the database in its original form, and the server runs in amortized online time T, it must be that $ST \geq n$. This lower bound implies that our fully-homomorphic-encryption-based PIR scheme achieves the optimal trade-off (up to $\log n$ factors and polynomials in the security parameter) between online server time and client storage, when the servers store the database in unmodified form.

In Theorem 23, we show that a similar lower bound holds, even if the client makes all of its queries in a single *batch* and if the client is able to store some precomputed information about the database contents. In particular, if the client stores at most S bits of information, the client makes a batch of Q non-adaptive queries, the server stores the database in its original form, and the server runs in amortized time T per query, it must hold that $ST + QT \geq n$. This generalizes the bound of Beimel, Ishai, and Malkin [13], who prove this result for the case $S = 0$. Our bound implies that when:

- $S < Q$, existing batch PIR schemes [68] achieve optimal amortized time even in the setting in which the client can obtain some preprocessed advice about the database contents, and
- $S > Q$, our new fully-homomorphic-encryption-based PIR scheme achieves optimal amortized time,

up to $\log n$ factors and polynomials in the security parameter.

1.2 Overview of Techniques

We construct our new PIR schemes in two steps. First, we construct a new sort of *two*-server PIR scheme. Second, we use cryptographic assumptions to "compile" the two-server scheme into a single-server scheme.

Step 1: Two-server offline/online PIR with a single-server online phase. In the first step (Sect. 3), we design a new type of *two*-server offline/online PIR scheme [39]. The communication pattern of the two-server schemes we construct is as follows:

1. *Offline phase.* In a setup phase, the client sends a setup request to the first server (the "offline server"). The offline server runs in time at least linear in the database size and returns to the client a "hint" about the database state. The hint has size sublinear in the length of the database.
2. *Online phases (runs once for each of Q queries).* Whenever the client wants to make a PIR query, it uses its hint to issue a query to the second server (the "online server"). The online server produces an answer to the query in time sublinear in the database size and returns its answer to the client. The total communication in this step is sublinear in the database size.

The client can run the online phase Q times—for some parameter Q determined by the PIR scheme—using the same hint and without communicating with the offline server. After Q queries, the client discards its hint and reruns the offline setup phase from scratch.

Prior offline/online PIR schemes [39] require the client to communicate with *both* servers in the online phases, whenever the client makes multiple queries with the same hint. (If the client only ever makes a single query, the client can communicate with only one server in the online phase, but then the scheme cannot achieve sublinear amortized time.) In contrast, our schemes crucially allow the client to only communicate with a single server (the online server) in the online phase. Unlike schemes for private stateful information retrieval [83], the online phase in our scheme runs in sublinear time.

To build our two-server offline/online PIR scheme, we give a generic technique for "compiling" a two-server PIR scheme that supports a *single* query with sublinear online time into one that supports *multiple* queries with sublinear online time. Plugging the existing single-query offline/online PIR schemes with sublinear online time [39,89] into this compiler completes the two-server construction.

Provided that the offline server time is $\widetilde{O}(n)$ and the number of supported queries is at least n^{ϵ}, for constant $\epsilon > 0$, this two-server scheme already allows adaptive queries and has sublinear total amortized time and sublinear client storage. The only limitation is that it requires two non-colluding servers.

Step 2: Converting a two-server scheme to a one-server scheme. The last step (Sects. 4 and 5) is to convert the two-server PIR scheme into a one-server scheme. Following Corrigan-Gibbs and Kogan [39], we have the client encrypt the hint request that she sends to the offline server using a fully homomorphic encryption scheme. (As we discuss in Sect. 4, Aiello, Bhatt, Ostrovsky, and Rajagopalan [2] proposed a similar technique for converting multi-prover proof systems to single-prover proof systems, formalizing the approach of Biehl, Meyer, and Wetzel [18].) The offline server can then homomorphically answer the client's hint request in the offline phase while learning nothing about it. At this point, the client can execute both the offline and online phases with the same server, which completes the construction.

To construct the PIR schemes from weaker assumptions (linearly homomorphic encryption), we exploit the linearity of the underlying two-server PIR scheme. In particular, we show that the hint that the client downloads from the offline server corresponds to a client-specified linear function applied to the database. With a careful balancing of parameters and application of linearly homomorphic encryption and standard single-server PIR, we show that the client can obtain this linear function without revealing it to the database server.

The construction of our most asymptotically efficient PIR scheme, which appears in Sect. 5, implicitly follows essentially the same two-step strategy. The only difference is that achieving the improved efficiency requires us to design a new two-server offline/online PIR scheme for multiple queries from scratch. The offline phase of this scheme requires the server to compute non-linear functions

of the client query—and thus requires fully homomorphic encryption—but the online time of the scheme is lower, which is the source of efficiency improvements.

Lower Bounds. Our first lower bound (Theorem 21) relates the number S of bits of information the client stores between queries and the amortized online time T of the PIR server, for PIR schemes in which the server stores the database in unmodified form. In particular, we show that $ST = \widetilde{\Omega}(n)$. To prove this lower bound, we show that if there is a single-server PIR scheme with client storage S and amortized online T, there exists a two-server offline/online PIR scheme for a single query with hint size S and online time T. Then, applying existing lower bounds on such schemes [39] completes the proof.

Our second lower bound (Theorem 23) considers the setting in which the client makes a batch of queries at once. We prove this result using an incompressibility argument [3,41–43,50,95], showing that the existence of a better-than-expected PIR scheme would yield a better-than-possible compression algorithm. (As we discuss in the full version [38], it is not clear whether it is possible to derive the same bound from the elegant and more modern "presampling" method [36,37,92]).

1.3 Related Work

Multi-server PIR. Chor, Goldreich, Kushilevitz, and Sudan [35] introduced private information retrieval and gave the first protocols, which were in the multi-server information-theoretic setting and achieved communication $O(n^{1/3})$. A sequence of works [5,11,12,21,22,33,46,49,55,96] then improved the communication complexity of PIR, and today's PIR schemes can achieve sub-polynomial communication complexity in the information-theoretic setting [46] and logarithmic communication complexity in the computational setting [22]. Multi-server PIR schemes are more efficient, both in terms of communication and computation, than single-server schemes. However, the security of multi-server PIR relies on non-collusion between the servers, which can be hard to guarantee in practice.

Single-Server PIR. Kushilevitz and Ostrovsky [74] presented the first single-server PIR schemes, based on linearly homomorphic encryption. A sequence of works then improved the communication complexity of single-server PIR, and showed how to construct PIR schemes with polylogarithmic communication from a wide range of public-key assumptions, such as the ϕ-hiding assumption [28,53], the decisional composite-residuosity assumption [30,77], the decisional Diffie-Hellman assumption [45], and the quadratic-residuosity assumption [45].

Recent works [1,4,6,52,81] have used lattice-based encryption schemes to improve the concrete efficiency of single-server PIR, in terms of both communication and computation. The goal is to get the most efficient single-server PIR schemes subject to the linear-server-time lower bound. These techniques are complementary to ours, and applying lattice-based optimizations to our setting could improve the concrete efficiency of our protocols.

Computational Overhead of PIR. All early PIR protocols required the servers to perform work linear in the database size when responding to a query.

Table 1. A comparison of single-server, many-query PIR schemes. We present the per-query, asymptotic costs of each scheme, on a database of size n, where each of m clients makes many PIR queries and at most $\hat{m}$ clients may be corrupted. We omit poly-logarithmic factors in n and m, along with polynomial factors in the security parameter. For lower bounds, we denote the extra client storage by S. We use ϵ as an arbitrarily small, positive constant. We amortize the costs over the number of queries that minimizes the per-query costs. For each scheme, the table indicates:
– the additional cryptographic assumptions made beyond single-server PIR with poly-logarithmic communication,
– the number of queries (per client) over which we amortize,
– whether the client makes her queries adaptively or as a batch,
– the *amortized* number of bits communicated per query,
– the *amortized* client and server time per query, and
– the additional number of bits stored by the client and the server between queries.
For schemes in the offline/online model, the communication and computation costs are taken to be the sum of the offline costs, amortized over the number of queries supported by a single offline phase, and the online costs. The extra server storage does not include the n-bit database, stored by the server. The extra client storage does not include the indices queried, even if these indices are queried as a batch.

Scheme (extra assumptions)	*Per-client queries*	*Adaptive?*	*Per-query comm.*	Per-query time		Extra storage	
				Client	Server	Client	Server
Batch PIR [68,64,6]	Q	×	1	1	$\frac{n}{Q}$	0	0
Stateful PIR [83]	$n^{1/2}$	✓	$n^{1/2}$	n	n †	$n^{1/2}$	0
Single-query single-server PIR							
Standard [74,28]	1	✓	1	1	n	0	0
Offline/online [39] (LHE)	1	✓	$n^{2/3}$	$n^{2/3}$	n	$n^{2/3}$	0
Offline/online [39] (FHE)	1	✓	$n^{1/2}$	$n^{1/2}$	n	$n^{1/2}$	0
Download entire DB	$n^{1-\epsilon}$	✓	n^{ϵ}	n^{ϵ}	n^{ϵ}	n	0
Doubly-efficient PIR							
Secret key (OLDC) [29,25]	$n^{1-\epsilon}$	✓	n^{ϵ}	n^{ϵ}	n^{ϵ}	1	mn
Public key (OLDC+**VBB**) [25]	1 *	✓	n^{ϵ}	n^{ϵ}	n^{ϵ}	0	n
Private anonymous data access							
Read-only [63] (FHE)	1 *	✓	$\hat{m}$	$\hat{m}$	$\hat{m}$	$\hat{m}$	$\hat{m}n^{1+\epsilon}$
This work							
Theorem 16 (LHE)	$n^{1/4}$	✓	$n^{1/2}$	$n^{1/2}$	$n^{3/4}$	$n^{3/4}$	0
Theorem 19 (FHE)	$n^{1/2}$	✓	$n^{1/2}$	$n^{1/2}$	$n^{1/2}$	$n^{1/2}$	0
Lower bounds, for Q queries, *on schemes storing the database in its original form*							
Standard PIR [14]	Q	×	–	–	$\geq \frac{n}{Q}$	–	–
This work (Theorem 21)	Q	✓	–	–	$\geq \frac{n}{S}$	S	0
This work (Theorem 23)	Q	×	–	–	$\geq \frac{n}{S+Q}$	S	0

† The number of public-key operations is $n^{1/2}$.

* This number of per-client queries assumes that the total number of clients, m, grows sufficently large.

Beimel, Ishai, and Malkin [14] showed that this is inherent, giving an $\Omega(n)$ lower bound on the server time. Their lower bound applies to both multi-server and single-server schemes with either information-theoretic or computational security.

Many lines of work have sought to construct PIR schemes with lower computational costs, which circumvent the above linear lower bound (Table 1):

- *PIR with preprocessing* denotes a class of schemes in which the server(s) store the database in encoded form [13,14,94], which allows them to respond to queries in time sublinear in the database size. The first such schemes targeted the multi-server setting. Recent work [25,29] applies oblivious locally decodable codes [19,23,24] to construct single-server PIR schemes with sublinear server time, after a one-time database preprocessing step. However, these schemes require extra server-side storage per client that is linear in the database size. While an idealized form of program obfuscation [9] can be used to drastically reduce this storage [25], the lack of concretely efficient candidate constructions for program obfuscation rules out the use of these schemes for the time being. In contrast, the single-server schemes in this paper require only standard assumptions.
  "Offline/online PIR" schemes use a different type of preprocessing: the client and server run a one-time linear-complexity offline setup process, during which the client downloads and stores information about the database. After that, the client can make queries to the database, and the server can respond in sublinear time. Previous works [39,72,89] mostly focus on the two-server setting, where they achieve sublinear amortized time. In the single-server setting, previous offline/online PIR schemes [39] allow for only a single online query after each execution of the offline phase. As a result, in the single-server setting, the cost of each query is still linear in the database size.
  Finally, Lipmaa [78] constructs single-server PIR with slighly sublinear time by encoding the database as a branching program that is obliviously evaluated in $O(\frac{n}{\log n})$ operations. The schemes in this work achieve significantly lower amortized time, yet require the client to make multiple queries.
- *Make queries in a non-adaptive batch:* When the client knows the entire sequence of database queries she will make in advance, the client and server can use "batch PIR" schemes [6,7,31,61,64,65,68] to achieve sublinear amortized server time. The multi-server scheme of Lueks and Goldberg [79] allows the servers to simultaneously process a batch of queries from different clients, and achieves sublinear per-query time. Our schemes require only one server and achieve sublinear amortized time, even given a single client making her queries in an adaptive sequence.
- *Download and store the entire database:* If the client has enough storage space, she can keep a local copy of the entire database. The server pays a linear cost to ship the database to the client, but the client can answer subsequent database queries on her own with no server work. In contrast, the schemes in this paper avoid having to store the entire database at the client.

– *Settle on a sublinear number of public-key operations:* Private stateful information retrieval [83] schemes improve the concrete efficiency of single-server PIR by having the server do a sublinear number of public-key operations for each query. Such schemes [81,83] still require a linear number of symmetric key and plaintext operations for each query. In contrast, the schemes in this paper require sublinear amortized work of any kind, per query.

Communication Lower Bounds on PIR. A series of works give bounds on the communication required for multi-server PIR [56,93]. Single-server PIR constructions match the trivial $\log n$ lower bound (up to polylogarithmic factors).

Lower Bounds for PIR with Preprocessing. Beimel, Ishai, and Malkin [13] proved that if a server can store an S-bit hint and run in amortized time T, then it must hold that $ST \geq n$. Persiano and Yeo [84] recently improved this lower bound to $ST \geq n \log n$ in the single-server case. In this paper, we are interested in offline/online PIR schemes, in which the client stores a hint and the server stores the database in unmodified form.

Lower Bounds on Oblivious RAM. Recent work proves strong limits on the performance of oblivious-RAM [58] schemes [26,69,73,75,76]. These schemes allow the server to maintain per-client state; in our setting of PIR, the server is stateless. The PIR setting thus requires different lower-bound approaches [13].

2 Background

Notation. We write the set of positive integers as $\mathbb{N}$. For an integer $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$ and we write the empty set as $\emptyset$. We ignore issues of integrality, and treat numbers such as $n^{1/2}$ and n/k as integers. We use $\mathrm{poly}(\cdot)$ to denote a fixed polynomial in its argument. We use the standard Landau notation $O(\cdot)$ and $\Omega(\cdot)$ for asymptotics. When the big-O contains multiple variables, such as $f(n) = O(n/S)$, all variables other than n are implicit functions of n (which is the database size when it is not made explicit). The notation $\widetilde{O}(f(n))$ hides polylogarithmic factors in the parameter n, and $\widetilde{O}_\lambda(\cdot)$ hides $\mathrm{poly}(\log n, \lambda)$ factors. For a finite set $\mathcal{X}$, $x \xleftarrow{R} \mathcal{X}$ denotes an independent and uniformly random draw from $\mathcal{X}$. When unspecified, we take all logarithms base two.

We work in the RAM model, with word size logarithmic in the input length (i.e., database size n) and polynomial in the security parameter λ. We give running times up to $\mathrm{poly}(\log n, \lambda)$ factors, which makes our results relatively independent of the specifics of the computational model. An "efficient algorithm" is one that runs in probabilistic polynomial time in its inputs and in λ.

2.1 Standard Definitions

We begin by defining the standard cryptographic primitives that this work uses.

Pseudorandom Permutations. We use the standard notion of pseudorandom permutations [57]. On security parameter $\lambda \in \mathbb{N}$, a domain size $n \in \mathbb{N}$, and a key space $\mathcal{K}_\lambda$, we denote a pseudorandom permutation by $\mathsf{PRP} : \mathcal{K}_\lambda \times [n] \to [n]$.

Definition 1 (Linearly homomorphic encryption). Let $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. The scheme is *linearly homomorphic* if, for every keypair $(\mathsf{sk}, \mathsf{pk})$ that Gen outputs,

- the message space is a group $(\mathcal{M}_{\mathsf{pk}}, +)$,
- the ciphertext space is a group $(\mathcal{C}_{\mathsf{pk}}, \cdot)$, and
- for every pair of messages $m_0, m_1 \in \mathcal{M}_{\mathsf{pk}}$, it holds that

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m_0) \cdot \mathsf{Enc}(\mathsf{pk}, m_1) \in \mathcal{C}_{\mathsf{pk}}) = \mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m_0 + m_1 \in \mathcal{M}_{\mathsf{pk}})).$$

Definition 2 (Gate-by-gate fully homomorphic encryption). We use $(\mathsf{FHE.Gen}, \mathsf{FHE.Enc}, \mathsf{FHE.Dec}, \mathsf{FHE.Eval})$ to denote a symmetric-key fully homomorphic encryption scheme [51]. We say a scheme is a *gate-by-gate* fully homomorphic encryption scheme if the homomorphic evaluation routine $\mathsf{FHE.Eval}$ on a circuit of size $|C|$ and security parameter λ runs in time $|C| \cdot \mathrm{poly}(\log |C|, \lambda)$. Standard fully homomorphic encryption schemes are gate-by-gate [27,51,54].

2.2 Definition of Offline/Online PIR

Throughout, we present our new single-server PIR schemes in an offline/online model [39,83]. That is, the client first interacts with the server in an offline phase to obtain a succinct "hint" about the database contents. This hint allows the client to make many queries in a subsequent online phase. Provided that the server-side cost is low enough in both phases, the server's total amortized time (including the cost of both phases) will be sublinear in the database size.

We now give definitions for one- and two-server offline/online PIR schemes that support many adaptive queries. Our definition of offline/online PIR differs from that of prior work in one important way [39,72]. In our definition, in the two-server setting, the client may only communicate with a *single* server in the online phase. Prior two-server offline/online PIR schemes [39,72] allow the client to communicate with *both* servers in the online phase.

Definition 3 (Offline/online PIR for adaptive queries). An *offline/online PIR scheme for adaptive queries* is a tuple of polynomial-time algorithms:

- $\mathsf{HintQuery}(1^\lambda, n) \to (\mathsf{ck}, q)$, a randomized algorithm that takes in a security parameter λ and a database length $n \in \mathbb{N}$, and outputs a client key ck and a hint request q,
- $\mathsf{HintAnswer}(D, q) \to a$, a deterministic algorithm that takes in a database $D \in \{0,1\}^n$ and a hint request q, and outputs a hint answer a,
- $\mathsf{HintReconstruct}(\mathsf{ck}, a) \to h$, a deterministic algorithm that takes in a client key ck and a hint answer a, and outputs a hint h,
- $\mathsf{Query}(\mathsf{ck}, i) \to (\mathsf{ck}', \mathsf{st}, q)$, a randomized algorithm that takes in a client key ck and an index $i \in [n]$, and outputs an updated client key ck', a client query state st, and a query q,
- $\mathsf{Answer}^D(q) \to a$, a deterministic algorithm that takes in a query q, and gets access to an oracle that:

Experiment 4 (Correctness). Parameterized by a PIR scheme Π, security parameter $\lambda \in \mathbb{N}$, number of queries $Q \in \mathbb{N}$, database size $n \in \mathbb{N}$, database $D \in \{0,1\}^n$, and query sequence $(i_1, \ldots, i_Q) \in [n]^Q$.

- Compute:

$$(\mathsf{ck}, q) \leftarrow \Pi.\mathsf{HintQuery}(1^\lambda, n)$$
$$a \leftarrow \Pi.\mathsf{HintAnswer}(D, q)$$
$$h \leftarrow \Pi.\mathsf{HintReconstruct}(\mathsf{ck}, a)$$

- For $t = 1, \ldots, Q$, compute:

$$(\mathsf{ck}, \mathsf{st}, q) \leftarrow \Pi.\mathsf{Query}(\mathsf{ck}, i_t)$$
$$a \leftarrow \Pi.\mathsf{Answer}^D(q)$$
$$(h, v_i) \leftarrow \Pi.\mathsf{Reconstruct}(\mathsf{st}, h, a)$$

- Output "1" if $v_t = D_{i_t}$ for all $t \in [Q]$. Output "0" otherwise.

Experiment 5 (Security). Parameterized by an adversary $\mathcal{A}$, PIR scheme Π, number of servers $k \in \{1, 2\}$, security parameter $\lambda \in \mathbb{N}$, number of queries $Q \in \mathbb{N}$, database size $n \in \mathbb{N}$, and bit $b \in \{0, 1\}$.

- Compute:

$$(\mathsf{ck}, q) \leftarrow \Pi.\mathsf{HintQuery}(1^\lambda, n)$$
If $k = 1$: *// Single-server security*
$$\mathsf{st} \leftarrow \mathcal{A}(1^\lambda, q)$$
Else: *// Two-server security*
$$\mathsf{st} \leftarrow \mathcal{A}(1^\lambda)$$

- For $t = 1, \ldots, Q$, compute:

$$(\mathsf{st}, i_0, i_1) \leftarrow \mathcal{A}(\mathsf{st})$$
$$(\mathsf{ck}, \_, q) \leftarrow \Pi.\mathsf{Query}(\mathsf{ck}, i_b)$$
$$\mathsf{st} \leftarrow \mathcal{A}(\mathsf{st}, q)$$

- Output $b' \leftarrow \mathcal{A}(\mathsf{st})$.

  - takes as input an index $j \in [n]$, and
  - returns the j-th bit of the database $D_j \in \{0, 1\}$,

  and outputs an answer string a, and
- $\mathsf{Reconstruct}(\mathsf{st}, h, a) \to (h', D_i)$, a deterministic algorithm that takes in a query state st, a hint h, and an answer string a, and outputs an updated hint h' and a database bit D_i.

In a deployment, $(\mathsf{HintQuery}, \mathsf{HintAnswer}, \mathsf{HintReconstruct})$ are executed in the offline phase, while $(\mathsf{Query}, \mathsf{Answer}, \mathsf{Reconstruct})$ are executed in each online phase. Furthermore, we say that the PIR scheme *supports Q adaptive queries* if it satisfies the following notions of (1) correctness and (2) security for Q queries:

Correctness for Q Queries. We require that if a client and a server correctly execute the protocol, the client can recover any Q database records of its choosing, even if the client chooses these records adaptively. Formally, a multi-query offline/online PIR scheme Π satisfies *correctness for Q queries* if for every $\lambda, n \in \mathbb{N}$, $D \in \{0,1\}^n$, and every $(i_1, \ldots, i_Q) \in [n]^Q$, Experiment 4 outputs "1" with probability $1 - \mathrm{negl}(\lambda)$.

Security for Q Queries. We require that an adversarial (malicious) server "learns nothing" about which sequence of database records the client is fetching, even if the adversary can adaptively choose these indices. In the single-server setting, where the same server runs both the offline and online phase, the adversary is first given the hint request. In the two-server setting, where a separate

server runs the offline phase, the adversary only sees the online queries. (This is sufficient, as an adversarial offline server trivially learns nothing about the client's queries since the hint request does not depend on these queries.)

Formally, for an adversary $\mathcal{A}$, multi-query offline/online PIR scheme Π, number of servers $k \in \{1, 2\}$, security parameter $\lambda \in \mathbb{N}$, database size $n \in \mathbb{N}$, and bit $b \in \{0, 1\}$, let $W_{\mathcal{A},k,\lambda,Q,n,b}$ be the event that Experiment 5 outputs "1" when parameterized with these values. We define the Q-query PIR advantage of $\mathcal{A}$:

$$\mathsf{PIRAdv}_k[\mathcal{A}, \Pi](\lambda, n) := \left|\Pr[W_{\mathcal{A},k,\lambda,Q,n,0}] - \Pr[W_{\mathcal{A},k,\lambda,Q,n,1}]\right|.$$

We say that a multi-query offline/online PIR scheme Π is *k-server secure* if, for all efficient algorithms $\mathcal{A}$, all polynomially bounded functions $n(\lambda)$, and all $\lambda \in \mathbb{N}$, $\mathsf{PIRAdv}_k[\mathcal{A}, \Pi](\lambda, n(\lambda)) \leq \mathrm{negl}(\lambda)$.

Definition 6 (Sublinear amortized time). We say that an offline/online PIR scheme has *sublinear amortized time* if there exists a number of queries. $Q \in \mathbb{N}$ such that the total server time required to run the offline and online phases for Q queries on a database of size n is $o(Qn)$. More formally, for every choice of the security parameter $\lambda \in \mathbb{N}$, database size $n \in \mathbb{N}$, and query sequence $(i_1, \ldots, i_Q) \in [n]^Q$, the total running time of $\mathsf{HintAnswer}$ (executed once) and Answer (executed Q times) in Experiment 4 must be $o(Qn)$.

Remark 7 (Handling an unbounded number of queries). A scheme with sublinear amortized time for some number of queries $Q \in \mathbb{N}$ immediately implies a scheme with sublinear amortized time for any larger number of queries, including a number that is a-priori unbounded. One can obtain such a scheme by "restarting" the scheme every Q queries and rerunning the offline phase from scratch. The amortized costs remain the same.

Remark 8 (Malicious security). In our definition (Definition 3), following prior work [39], the client's queries do not depend on the server's answers to prior queries. In this way, our PIR schemes naturally protect client privacy against a malicious server—the server learns the same information about the client's queries whether or not the server executes the protocol faithfully.

Remark 9 (Correctness failures). Our definition does not require that correctness holds if the client makes a sequence of queries that is correlated with the randomness it used to generate the hint request. A stronger correctness definition would guarantee correctness in all cases (i.e., with probability one). Strengthening our PIR schemes to provide this form of correctness represents an interesting challenge for future work.

Remark 10 (Handling database changes). In many natural applications of private information retrieval, the database contents change often. Naïvely, whenever the database contents change, the client and server would need to rerun the costly hint-generation process. In the limit—when the entire contents of the database changes between a client's queries—rerunning the hint-generation step is inherently required. When the database changes more slowly, prior work on

offline/online PIR [72], building on much earlier work in dynamic data structures [17], shows how to update the client's hint at modest cost. In particular, when a constant number of database rows change between each pair of client queries, the scheme's costs do not change, up to factors in the security parameter and logarithmic in the database size. These techniques from prior work apply directly to our setting, so we do not discuss them further.

3 Two-Server PIR with a Single-Server Online Phase and Sublinear Amortized Time

In this section, we give a generic construction that converts a two-server offline/online PIR scheme that supports *a single query* into a two-server offline/online PIR scheme that supports *any number of adaptive queries*. The transformation has three useful properties:

1. If the original PIR scheme has linear offline server time, then the resulting multi-query scheme has linear offline server time as well.
2. If the original PIR scheme has sublinear online server time, then the resulting multi-query scheme has sublinear online server time as well.
3. During the online phase—when the client is making its sequence of adaptive queries—the client only communicates with one of the servers. (In contrast, prior two-server PIR schemes with sublinear amortized time [39,72] require the client to communicate with *both* servers in the online phase.)

After presenting the generic transformation (Lemma 11) in this section, we instantiate this transformation in Sect. 4 and use it to construct single-server PIR schemes with sublinear amortized time.

Lemma 11 (The Compiler Lemma). *Let Π be a two-server offline/online PIR scheme that supports a single query. Then, for any database size $n \in \mathbb{N}$, security parameter $\lambda \in \mathbb{N}$, and number of queries $Q < n$, Construction 15, when instantiated with a secure pseudorandom permutation, is a two-server offline/online PIR scheme that supports Q* adaptive queries *and whose offline and online phases have communication, computation, and client storage costs dominated by running $O(\lambda Q)$ instances of Π, each on a database of size n/Q.*

To prove the lemma, we must show that the scheme of Construction 15 satisfies the claimed efficiency properties, along with correctness and security. Efficiency follows by construction. We give the full correctness and security arguments in the full version of this paper [38].

Remark 12. In the PIR scheme implied by Lemma 11, the online-phase upload communication (from the client to server) is in fact only as large as the upload communication required for running a *single* instance of the underlying PIR scheme Π on a database of size n/Q.

Before giving the construction that proves Lemma 11, we describe the idea behind our approach. We take inspiration from the work of Ishai, Kushilevitz,

Ostrovsky, and Sahai [68], who construct "batch" PIR schemes, in which the client can issue a batch of Q queries at once, and the server can respond to all Q queries in time $\widetilde{O}(n)$. (In contrast, answering Q queries using a non-batch PIR scheme requires server time $\Omega(Qn)$.) The crucial difference between our PIR schemes and prior work on batch PIR is that our schemes allow the client to make its Q queries adaptively, rather than in a single batch all at once.

Our idea is to first permute the database according to a pseudorandom permutation and then partition the n database records into Q chunks, each of size n/Q. The key observation is that, if the client makes Q adaptive queries, it is extremely unlikely that the client will ever need to query any chunk more than λ times. In particular, by a balls-in-bins argument, the probability, taken over the random key of the pseudorandom permutation, that any chunk receives more than λ queries is negligible in λ.

Then, given a two-server offline/online PIR scheme Π for a *single query*, we construct a two-server offline/online PIR scheme for many queries as follows:

- *Offline phase.* The client and the offline server run the offline phase of Π on each of the Q database chunks λ times. For each of the Q database chunks, the client then holds λ client keys and hints.
- *Online phase.* Whenever the client wants to make a database query, it identifies the chunk in which its desired database record falls. The client finds an unused client key for that chunk and runs the online phase of Π for that chunk to produce a query. The client sends the query to the online server, who answers that query with respect to each of the Q database chunks. Using the online server's answers, the client can reconstruct its database record of interest. Crucially, the client's query does not reveal to the server the chunk in which its desired database record falls. Finally, the client then deletes the client key and hint that it used for this query.

The formal description of our protocol appears in Construction 15.

Remark 13. Construction 15 uses a pseudorandom permutation (PRP) to permute and partition the database. The client then reveals the PRP key it used for this partitioning to the server. Crucially, the security of our construction does *not* rely on the pseudorandomness of the PRP. The PRP security property only appears in the correctness argument of our scheme (which we give in the full version of this paper [38]). So, revealing the PRP key to the server in this way has no effect on the security of the scheme.

Remark 14 (Reducing online download). In the online phase of Construction 15, the online server's answer to the client consists of a vector of Q answers $a = ((a)_1, \ldots, (a)_Q)$. The client uses only one of these answers $(a)_{j^*}$. To reduce download cost, the client and server can run a single-server PIR protocol, where the server's input is the database a of Q answers and the client's input is the index $j^* \in [Q]$ of it's desired answer. This reduces the client's online download cost by a factor of Q, at the cost of requiring the server to perform $O_\lambda(Q)$ public-key operations in the online phase.

Construction 15 (Two-server offline/online PIR for Q adaptive queries with a single-server online phase). The scheme uses a single-query two-server offline/online PIR scheme Π and a pseudorandom permutation $\mathsf{PRP} : \mathcal{K}_\lambda \times [n] \to [n]$. The scheme is parameterized by a maximum number of queries $Q = Q(n) < n$.

I. Offline phase.

$\mathsf{HintQuery}(1^\lambda, n) \to (\mathsf{ck}, q)$.

1. For $j \in [Q]$ and $\ell \in [\lambda]$: $((\hat{\mathsf{ck}})_{j\ell}, (\hat{q})_{j\ell}) \leftarrow \Pi.\mathsf{HintQuery}(1^\lambda, n/Q)$.
2. Sample $k \xleftarrow{R} \mathcal{K}_\lambda$, set $\mathsf{ck} \leftarrow (k, \hat{\mathsf{ck}}, \emptyset)$, and set $q \leftarrow (k, \hat{q})$.
3. Return (ck, q).

$\mathsf{HintAnswer}(D, q) \to a$.

1. Parse $(k, \hat{q}) \leftarrow q$.
2. *// Permute the database according to* $\mathsf{PRP}(k, \cdot)$ *and divide it into* Q *chunks.*
   For $j \in [Q]$: $C_j \leftarrow (D_{\mathsf{PRP}(k,(j-1)(n/Q)+1)} \| \cdots \| D_{\mathsf{PRP}(k,(j+1)(n/Q))}) \in \{0,1\}^{n/Q}$.
3. For $j \in [Q]$ and $\ell \in [\lambda]$: $(a)_{j\ell} \leftarrow \Pi.\mathsf{HintAnswer}(C_j, (\hat{q})_{j\ell})$.
4. Return a.

$\mathsf{HintReconstruct}(\mathsf{ck}, a) \to h$.

1. Parse $(k, \hat{\mathsf{ck}}, \mathsf{queried}) \leftarrow \mathsf{ck}$.
2. For $j \in [Q]$ and $\ell \in [\lambda]$: $(\hat{h})_{j\ell} \leftarrow \Pi.\mathsf{HintReconstruct}((\hat{\mathsf{ck}})_{j\ell}, (a)_{j\ell})$.
3. Set $\mathsf{cache} \leftarrow \{\}$. *// An empty map (associative array) data structure.*
4. Return $h = (\hat{h}, \mathsf{cache})$.

II. Online phase.

$\mathsf{Query}(\mathsf{ck}, i) \to (\mathsf{ck}', \mathsf{st}, q)$.

1. Parse $(k, \hat{\mathsf{ck}}, \mathsf{queried}) \leftarrow \mathsf{ck}$.
2. Find (the unique) $i^* \in [n/Q]$ and $j^* \in [Q]$ so that $\mathsf{PRP}(k, i) = (j^* - 1)(n/Q) + i^*$.
3. Find $\ell^* \in [\lambda]$ such that $(\mathsf{ck})_{j^*\ell^*} \neq \perp$.
   - If no such ℓ^* exists or $i \in \mathsf{queried}$, sample $i^* \xleftarrow{R} [n/Q]$ and choose a random $j^* \in [Q]$ and $\ell^* \in [\lambda]$ out of those for which $(\mathsf{ck})_{j^*\ell^*} \neq \perp$.
4. Let $(\_, \mathsf{st}', q') \leftarrow \Pi.\mathsf{Query}((\hat{\mathsf{ck}})_{j^*\ell^*}, i^*)$.
5. Let $(\hat{\mathsf{ck}})_{j^*\ell^*} \leftarrow \perp$, let $\mathsf{st} \leftarrow (\mathsf{st}', i, j^*, \ell^*)$, let $q \leftarrow (k, q')$, and let $\mathsf{ck}' \leftarrow (k, \hat{\mathsf{ck}}, \mathsf{queried} \cup \{i\})$.
6. Return $(\mathsf{ck}', \mathsf{st}, q)$.

$\mathsf{Answer}^D(q) \to a$.

1. Parse $(k, q') \leftarrow q$.
2. For $j \in [Q]$: $(a)_j \leftarrow \Pi.\mathsf{Answer}^{\mathcal{O}_j}(q')$, where $\mathcal{O}_j(x) := D_{\mathsf{PRP}(k,(j-1)(n/Q)+x)}$.
3. Return a.

$\mathsf{Reconstruct}(\mathsf{st}, h, a) \to (h', D_i)$.

1. Parse $(\mathsf{st}', i, j^*, \ell^*) \leftarrow \mathsf{st}$ and parse $(\hat{h}, \mathsf{cache}) \leftarrow h$.
2. If $\mathsf{cache}[i]$ is not set, let $\mathsf{cache}[i] \leftarrow \Pi.\mathsf{Reconstruct}(\mathsf{st}', (\hat{h})_{j^*\ell^*}, (a)_{j^*})$.
3. Set $D_i \leftarrow \mathsf{cache}[i]$. Set $h' \leftarrow (\hat{h}, \mathsf{cache})$.
4. Return (h', D_i).

4 Single-Server PIR with Sublinear Amortized Time from DCR, QR, DDH, or LWE

In this section, we use the general transformation of Sect. 3 to construct the first single-server PIR schemes with sublinear amortized total time and sublinear extra storage, allowing the client to make her queries adaptively.

These constructions work in two steps:

- First, we use the Compiler Lemma (Lemma 11) to convert a two-server offline/online PIR scheme for a single query into a two-server offline/online PIR scheme for *multiple adaptive queries*, in which the client only communicates with a single server in the online phase.
- Next, we use linearly homomorphic encryption and single-server PIR to allow the client and server to run the offline phase of the two-server scheme without leaking any information to the server. At this point, we can execute the functionality of both servers in the two-server scheme using just a single server. In other words, we have constructed a single-server offline/online PIR scheme that supports multiple adaptive queries.

The idea of using homomorphic encryption to run a two-server protocol on a single server arose first, to our knowledge, in the domain of multi-prover interactive proofs. Aiello, Bhatt, Ostrovsky, and Rajagopalan [2] formalized this general approach, which was initially proposed by Biehl, Meyer, and Wetzel [18]. Subsequent work demonstrated that compiling multi-prover proof systems to single-prover systems requires care [44,47,48,71,91] (in particular it requires the underlying proof system to be sound against "no-signaling" provers [91]). Corrigan-Gibbs and Kogan [39] used homomorphic encryption to convert a two-server PIR scheme to a single-server offline/online PIR scheme that supports a *single* query in sublinear online time. Our contribution is to construct a single-server PIR scheme that supports multiple, adaptive queries and that thus achieves *sublinear amortized total time*.

We now show that any one of a variety of cryptographic assumptions—the Decision Composite Residuosity assumption [77,82], the Quadratic Residuosity assumption [59], the Decision Diffie-Hellman assumption [20], or the Learning with Errors assumption [87]—suffices for constructing single-server PIR with sublinear amortized time:

Theorem 16 (Single-server PIR with sublinear amortized time). *Under the DCR, LWE, QR, or DDH assumptions, there exists a single-server offline/online PIR scheme that, on database size n, security parameter λ, and as long as the client makes at least $n^{1/4}$ adaptive queries, has*

- *amortized communication $\widetilde{O}_\lambda(n^{1/2})$,*
- *amortized server time $\widetilde{O}_\lambda(n^{3/4})$,*
- *amortized client time $\widetilde{O}_\lambda(n^{1/2})$, and*
- *client storage $\widetilde{O}_\lambda(n^{3/4})$.*

The proof of Theorem 16 will make use of the following two-server offline/online PIR scheme which is implicit in prior work.

Lemma 17 (Implicit in Theorem 20 of CK20 [39]). *There is a two-server offline/online PIR scheme (with information-theoretic security) that supports a single query on database size n such that, in the offline phase:*

- *the client uploads a vector $q \in \{0,1\}^n$ to the offline server,*
- *the offline server computes the inner product of the database with all n cyclic shifts of the query vector q (in $\widetilde{O}(n)$ time using a fast Fourier transform),*
- *the client downloads $\widetilde{O}(\sqrt{n})$ bits of the resulting matrix-vector product*

and, in the online phase:

- *the client uploads $\widetilde{O}(\sqrt{n})$ bits to the online server,*
- *the online server runs in time $\widetilde{O}(\sqrt{n})$, and*
- *the client downloads one bit.*

Proof of Theorem 16. The proof works in two main steps. First, we use Lemma 11 to "compile" the single-query two-server PIR scheme of Lemma 17 into a multi-query two-server PIR scheme. Second, we use linearly homomorphic encryption—following the work of Corrigan-Gibbs and Kogan [39] in the single-query setting—to allow a single server to implement the role of both servers.

Step 1: A stepping-stone two-server scheme. We first construct a two-server offline/online PIR scheme that: (a) supports multiple queries, (b) has sublinear online time, and (c) requires only one server in the online phase. To do so, we use the Compiler Lemma (Lemma 11) to convert the two-server PIR scheme of Lemma 17 into a two-server PIR scheme satisfying these three goals.

In particular, Lemma 11 and Lemma 17 together imply a two-server offline/online PIR scheme that supports any number of queries $Q < n$, and whose offline and online phases consist of running $O(\lambda Q)$ instances of the PIR scheme of Lemma 17 on databases of size n/Q. The resulting scheme then has the following structure in the offline phase:

- the client uploads $\widetilde{O}_\lambda(Q)$ bit vectors to the offline server, each of size n/Q,
- the offline server applies a length-preserving linear function to each vector (in quasi-linear time, as in the Lemma 17 scheme),
- the client downloads a total of $\widetilde{O}_\lambda(\sqrt{Qn})$ bits from the vectors that the server computes.

And in the online phase,

- the client uploads $\widetilde{O}_\lambda(\sqrt{Qn})$ bits to the online server,
- the online server runs in time $\widetilde{O}_\lambda(\sqrt{Qn})$, and
- the client downloads $\widetilde{O}_\lambda(Q)$ bits.

This scheme requires the existence of one-way functions.

As desired, this scheme supports multiple queries, has sublinear online time (whenever $Q \ll n$), and requires only one server in the online phase. The offline upload cost and the client time of the scheme are $\widetilde{\Omega}_\lambda(n)$—linear in the database size, but we remove this limitation later on.

Step 2: Using homomorphic encryption to run the two-server scheme on one server. Next, we show that the client can fetch the information it needs to complete the offline phase of the Step-1 scheme without revealing any information to the server. In the Step-1 scheme, the offline server's work consists of evaluating a client-supplied linear function over the database and can thus be performed under linearly homomorphic encryption. For this step, we will need a linearly homomorphic encryption scheme with ciphertexts of size $\widetilde{O}_\lambda(1)$, along with a single-server PIR scheme with communication cost and client time $\widetilde{O}_\lambda(1)$. The existence of both primitives follows from the Decision Composite Residue (DCR) assumptions [77,82] and the Learning with Errors (LWE) assumption [87]. Recent work of Döttling, Garg, Ishai, Malavolta, Mour, and Ostrovsky [45] shows that the Quadratic Residuosity (QR) assumption [59] and decision Diffie-Hellman (DDH) assumption [20] also imply these primitives.

In particular, the client first samples a random encryption key for a linearly homomorphic encryption scheme. Then the client executes the offline phase as follows:

- The client encrypts each component of its $\widetilde{O}_\lambda(Q)$ bit vectors using the linearly homomorphic encryption scheme. The client sends these vectors to the server.
- Under encryption, the server applies the length-preserving linear function to each encrypted vector. As in the Step-1 scheme, this computation takes $\widetilde{O}_\lambda(n)$ time using an FFT on the encrypted values.
- The client uses a single-server PIR scheme [74], to fetch a total of $\widetilde{O}_\lambda(\sqrt{Qn})$ components of the ciphertext vectors that the server has computed. Since modern single-server PIR schemes have communication cost $\widetilde{O}_\lambda(1)$, this step requires communication and client time $\widetilde{O}_\lambda(\sqrt{Qn})$. Using batch PIR [7,64,68], the server can answer this set of queries in time $\widetilde{O}_\lambda(n)$.

Finally, the client decrypts the resulting ciphertexts to recover exactly the same information that it obtained at the end of the offline phase of the two-server scheme. At this point, the offline phase has upload $\widetilde{O}_\lambda(n)$, server time $\widetilde{O}_\lambda(n)$, client time $\widetilde{O}_\lambda(n)$, and download $\widetilde{O}_\lambda(\sqrt{Qn})$. The online phase has upload $\widetilde{O}_\lambda(\sqrt{Qn})$ bits, server time $\widetilde{O}(\sqrt{Qn})$, client time $\widetilde{O}_\lambda(\sqrt{Qn}+Q)$, and download $\widetilde{O}_\lambda(Q)$.

Final Rebalancing. We complete the proof by reducing the offline upload cost using the standard rebalancing idea [34, Section 4.3]. In particular, we divide the database into k chunks, of size $n' = n/k$, for a parameter k chosen later.

Now, the offline phase has upload $\widetilde{O}_\lambda(n/k)$, server time $\widetilde{O}_\lambda(n)$, client time $\widetilde{O}_\lambda(n/k+\sqrt{Qnk})$, and download $k \cdot \widetilde{O}_\lambda(\sqrt{Qn/k})$ and the online phase has upload $\widetilde{O}_\lambda(\sqrt{Qn/k})$ bits, server time $k \cdot \widetilde{O}(\sqrt{Qn/k})$, client time $\widetilde{O}_\lambda(\sqrt{Qn/k}+Qk)$ and

download $k \cdot \widetilde{O}_\lambda(Q)$. We choose Q and k to balance the following costs, ignoring $\mathrm{poly}(\lambda, \log n)$ factors:

- the amortized offline time: n/Q, and
- the online server time: $\sqrt{kQn}$.

To do so, we choose $k = \frac{n}{Q^3}$ and $Q \leq n^{1/3}$. This yields a PIR scheme with amortized server time $\widetilde{O}_\lambda(n/Q)$, amortized client time $\widetilde{O}_\lambda(Q^2 + n/Q^2)$ and amortized communication $\widetilde{O}_\lambda(Q^2 + n/Q^2)$. The client storage is equal to the (non-amortized) offline download cost, which is $\widetilde{O}_\lambda(n/Q)$.

Finally, to construct the scheme of Theorem 16, we chose $Q = n^{1/4}$ to minimize the offline upload. This causes the amortized server time and the client storage to become $\widetilde{O}_\lambda(n^{3/4})$, while the amortized client time and the amortized communication are both $\widetilde{O}_\lambda(n^{1/2})$.

Efficiency. The efficiency claims follow immediately from the construction.

Security. The security argument closely follows that of prior work on single-server offline/online PIR [39]. More formally, the server's view in an interaction with a client consists of (1) the client's encrypted bit vectors sent in the offline phase, (2) the client's standard single-server PIR queries sent in the offline phase, (3) the messages that the client sends in the online phase. To prove security, we can construct a sequence of hybrid distributions that move from the world in which the client queries a sequence of database indexes $I_0 = (i_{0,1}, i_{0,1}, \ldots, i_{0,Q})$ to the world in which the client queries a different sequence $I_1 = (i_{1,1}, i_{1,1}, \ldots, i_{1,Q})$. The steps of the argument are:

- replace the encrypted bit vectors with encryptions of zeros, using the semantic security of the encryption scheme,
- replace the client's standard single-server PIR query with a query to a fixed database row, using the security of the underlying single-server PIR scheme,
- swap query sequence I_0 with query sequence I_1, using the security of the underlying two-server offline/online PIR scheme,
- swap the client's standard single-server PIR query and encrypted bit vectors back again, using the security of these primitives. □

Remark 18 (Single-server PIR with $\widetilde{O}_\lambda(n^{2/3})$ amortized time and communication). With an alternate rebalancing (taking Q to be $n^{1/3}$), we can build a single-server offline/online PIR scheme that, as long as the client makes at least $n^{1/3}$ adaptive queries, has amortized communication $\widetilde{O}_\lambda(n^{2/3})$, amortized server time $\widetilde{O}_\lambda(n^{2/3})$, amortized client time $\widetilde{O}_\lambda(n^{2/3})$, and client storage $\widetilde{O}_\lambda(n^{2/3})$. This PIR scheme has better amortized server time than that of Theorem 16, at the cost of requiring a client upload linear in n in the offline phase. (However, the *amortized* communication of this scheme is still sublinear in n.)

5 Single-Server PIR with Optimal Amortized Time and Storage from Fully Homomorphic Encryption

In this section, we construct a single-server many-query offline/online PIR scheme directly, rather than through a generic transformation. Assuming fully homomorphic encryption (Definition 2), our scheme achieves the optimal tradeoff between amortized server time and client storage, up to polylogarithmic factors. This fills a gap left open by the protocols of Sect. 4 and demonstrates that the lower bound we give in Sect. 6 is tight. We prove the following result:

Theorem 19 (Single-server PIR with optimal amortized time and storage from fully homomorphic encryption). *Assuming gate-by-gate fully homomorphic encryption (Definition 2), there exists a single-server offline/online PIR scheme that, on security parameter $\lambda \in \mathbb{N}$, database size $n \in \mathbb{N}$, and maximum number of queries $Q < n$, supports Q adaptive queries with:*

- *amortized server time $\tilde{O}_\lambda(n/Q)$,*
- *client-side storage $\tilde{O}_\lambda(Q)$,*
- *amortized communication $\tilde{O}_\lambda(n/Q)$, and*
- *amortized client time $\tilde{O}_\lambda(Q + n/Q)$.*

This new scheme achieves amortized server time better than we could expect from any protocol derived from the generic compiler of Sect. 3, given current state-of-the-art offline/online PIR protocols. To answer each query, that compiler executes the online phase of a PIR scheme on Q database chunks, each of size n/Q. Similar to the compiler of Sect. 3, the PIR scheme here works by splitting the database into random chunks, so that the client's distinct adaptive queries fall into distinct chunks with high probability. However, the new PIR scheme in this section keeps the mapping of database rows to chunks *secret* from the server. (In contrast, in the scheme of Sect. 3, the client reveals to the server the mapping of database rows to chunks.) By keeping the mapping of database rows to chunks secret, in the online phase of this scheme, the server only has to compute over the contents a single chunk. In this way, we achieve lower computation than the schemes of Sect. 4, which execute an online phase for each database chunk.

In this section, we sketch the ideas behind the PIR scheme that proves Theorem 19; a complete proof appears in the full version of this paper [38].

Proof idea for Theorem 19. At a very high level, the PIR scheme that we construct works as follows:

1. In an offline phase, the client chooses small, random subsets $S_1, \ldots, S_m \subseteq [n]$. For each subset, the client privately fetches from the server the parity of the database bits indexed by the set.
2. When the client wants to fetch database record i in the online phase, it finds a subset $S \in \{S_1, \ldots, S_m\}$ such that $i \in S$. Then, the client *usually* asks the server for the parity of the database bits indexed by $S \setminus \{i\}$. The parity of the database bits indexed by S and $S \setminus \{i\}$ give the client enough

information to recover the value of the ith database record, D_i. Then, the client re-randomizes the set S it just used.

In more detail, our PIR scheme operates as follows: in the offline phase, the client samples $(\lambda+1)\cdot Q$ random subsets of $[n]$, each of size n/Q. We refer to the first λQ sets as the "primary" sets and to the remaining Q sets as the "backup" sets. For each set S, the client retrieves the parity of the database bits the set indexes, i.e., $\sum_{j\in S} D_j \mod 2$, from the server, while keeping the set contents hidden using encryption. For each backup set S, the client additionally chooses a random member of the set S and privately retrieves the database value indexed by that element, via a batch PIR protocol [7,64,68].

With high probability over the client's random choice of sets, whenever the client wants to fetch the i-th database record, the client holds a primary set that contains i. Again with good probability, the client then asks the server for the parity of the database bits indexed by the punctured set $S\setminus\{i\}$, with which she can reconstruct the desired database value D_i. Finally, the client must refresh her state, as using the same S to query for another index i' could leak (i, i') to the server and thus break security. To achieve this, the client discards S and promotes the next available backup set, S_b, to become a new primary set. If S_b does not already contain i, the client modifies S_b by deleting the set element whose database value she knows and inserting i; the client recomputes the parity of this new set using the value of D_i she has just retrieved. With this mechanism, the distribution of the client's primary sets remains random, ensuring that her online queries are independent.

There are two failure events in this scheme: it is possible that (a) none of the primary sets contain the index queried, i, or that (b) the client sends the server a set other than $S\setminus\{i\}$, as decided by a coin flip (to avoid always sending a query set that does not contain i). We drive down the probability of either failure event to $\mathrm{negl}(\lambda)$, by repeating the offline and online phases λ times. Then, by construction, this scheme satisfies correctness for Q queries. Intuitively, the scheme is secure because (a) the use of encryption and batch PIR in the offline phase prevents the server from learning the contents of the presampled sets, and (b) the client's online queries are indistinguishable from uniformly random subsets of $[n]$ of size $n/Q - 1$, as proved in the full version of this paper [38].

We now discuss the PIR scheme's efficiency.

Communication and Storage. The client can succinctly represent her presampled sets with only logarithmic-size keys by leveraging pseudorandomness. Then, in the offline phase, she exchanges only $\widetilde{O}_\lambda(Q)$ bits with the server to communicate the (encrypted) descriptions and parities of $O_\lambda(Q)$ randomly sampled sets. The client additionally retrieves the database values of Q indices—one from each backup set—in $\widetilde{O}_\lambda(Q)$ communication with batch PIR. The client stores her presampled sets and her state between queries in $\widetilde{O}_\lambda(Q)$ bits. In each online phase, the client must however hide whether she inserted an index into her query set (and, if so, which index she inserted). Therefore, the client explicitly lists all elements in the punctured set she is querying for (instead of using pseudorandomness) and thus exchanges $\widetilde{O}_\lambda(n/Q)$ bits with the server.

Computation. In the offline phase, the client retrieves the encrypted parities of the database bits indexed by each of $O_\lambda(Q)$ encrypted sets of size n/Q. In the full version of this paper [38], we present a Boolean circuit that computes the parities of the database bits of s subsets of $[n]$, each of size ℓ, in $\widetilde{O}(s \cdot \ell + n)$ gates. Our circuit is inspired by circuits for private set intersection [67,85,86] and makes use of sorting networks [10]. The server can execute the offline phase in $\widetilde{O}_\lambda(n)$ time by running the above circuit under a gate-by-fate fully homomorphic encryption scheme. Further, the offline server can respond to the client's batch PIR query in $\widetilde{O}_\lambda(n)$ time. In each online phase, the server must complete $O_\lambda(n/Q)$ work per query, as it computes the parity of a punctured set containing $n/Q - 1$ elements. Thus, each query requires $\widetilde{O}_\lambda(n/Q)$ amortized total server time.

As for the client, in the offline phase, she generates $\mathcal{O}_\lambda(Q)$ random sets. Using pseudorandomness to represent each set, the time to generate these sets without expanding them is $\widetilde{O}_\lambda(Q)$. Also in the offline phase, the client runs a batch PIR protocol with the server to recover Q database values, requiring at most $\widetilde{O}_\lambda(Q)$ client time. In the online phase, the client first has to find a primary set that contains the index $i \in [n]$ she wants to read. By generating each set using a pseudorandom permutation, she can efficiently test whether each set contains i by inverting the permutation in time $\widetilde{O}_\lambda(1)$. Testing all $\mathcal{O}_\lambda(Q)$ primary sets takes the client time $\widetilde{O}_\lambda(Q)$. When she finds a succinctly-represented primary set that contains i, the client expands the set in time $\widetilde{O}_\lambda(n/Q)$ to build her online query. Finally, promoting a backup set to become a new primary set and, if necessary, replacing a set element by i takes time $\widetilde{O}_\lambda(1)$. We conclude that the client's amortized, per-query time is $\widetilde{O}_\lambda(Q + n/Q)$.

6 Lower Bounds

In this section, we present lower bounds for multi-query offline/online PIR schemes in which the server stores the database in its original form—that is, the server does not preprocess or encode the database. (If preprocessing is allowed, candidate single-server PIR schemes using program obfuscation can circumvent our lower bounds [25].)

Remark 20 (Generalization to multi-server PIR). While we present and prove these lower bounds in the single-server setting, both lower bounds hold for protocols with any constant number of servers. With multiple servers, T bounds the database bits probed per query by any online server.

6.1 Lower Bound for Adaptive Schemes

First, we give a new lower bound on the product of the (a) client storage and (b) online time of any single-server, offline/online PIR scheme for many adaptive queries. Specifically, we show that in any adaptive, multi-query, offline/online PIR scheme, where the client stores S bits between queries and the server responds to each query in amortized time T, it must hold that $ST = \widetilde{\Omega}(n)$.

This new lower bound matches the best adaptive multi-query scheme in the two-server setting [39, Section 4] and it matches our new scheme (Sect. 5) in the single-server setting, up to polylogarithmic factors.

In the following, we say that a single-server PIR scheme for Q adaptive queries probes T database bits per query *on average* if, for every sequence of Q indices and *every choice of the client's randomness*, the server makes at most QT total probes to the database in the process of answering all Q queries.

Theorem 21 (Lower bound for adaptive schemes). *Consider a computationally secure, single-server PIR scheme for many adaptive queries, such that, on security parameter $\lambda \in \mathbb{N}$ and database size $n \in \mathbb{N}$,*

- *the server stores the database in its original form,*
- *the client stores at most S bits between consecutive queries, and*
- *the server probes T database bits per query on average.*

Then, for polynomially bounded $n = n(\lambda)$ and large enough λ, it holds that $(S+1) \cdot (T+1) \geq \widetilde{\Omega}(n)$.

We give a complete proof of Theorem 21 in the full version of this paper [38]. Our proof invokes the following lower bound from prior work, which shows that for any *single-query* offline/online PIR scheme, either the offline communication or the online server time must be large:

Theorem 22 ([39, Section 6]). *Consider a computationally secure, single-query, offline/online PIR scheme such that, on security parameter $\lambda \in \mathbb{N}$ and database size $n \in \mathbb{N}$,*

- *the server stores the database in its original form,*
- *the client downloads C bits in the offline phase,*
- *the server probes T bits of the database to process each online query, and*
- *the client recovers its index of interest with probability at least $\epsilon \geq 1/2 + \Omega(1)$.*

Then, for polynomially bounded $n = n(\lambda)$, it holds that $(C+1) \cdot (T+1) \geq \widetilde{\Omega}(n)$.

Proof idea for Theorem 21. We show that any *multi-query* PIR scheme with small client storage implies a *single-query* offline/online PIR scheme with small offline communication. In more detail, the reduction works as follows:

1. First, we show that, for any *many-query* PIR scheme Π as in the theorem statement, there must exist a query sequence that satisfies the following condition: if the client makes PIR queries to each of the indices in this sequence one at a time, and then makes any subsequent PIR query, the server answers this last query with at most T database probes in expectation. We call such a query sequence a *good* query sequence.
2. Then, we build a *single-query* PIR scheme using Π and any fixed, good query sequence for Π. In an offline phase, we first let the PIR server run Π's offline phase, and then run as many iterations of Π's online phase as needed to query

for each index in the good query sequence. At this point, the server sends its intermediate state from running Π to the client. In an online phase, the client then runs one iteration of Π's online phase, using the intermediate state it received from the server, to query for its index of interest.
By construction, this single-query scheme requires S bits of offline download, and at most T database probes in expectation in the online phase. Correctness and security follow from the correctness and security of Π.

3. Finally, we modify the above single-query scheme to make $O(T)$ online database probes in the worst case, rather than in expectation.

Applying Theorem 22 to this single-query scheme then gives the bound on the client storage S and the running time T of the PIR scheme. □

6.2 Lower Bound for Batch PIR with Advice

The lower bound of section Sect. 6.1 rules out PIR schemes with small client storage and small amortized server online time in the adaptive setting. In this section, we ask whether it is possible to do better if the client makes all of its queries in a *single non-adaptive batch*. In particular, we consider schemes for "batch PIR with advice," in which a client obtains—via out-of-band means or via an offline phase—S bits of preprocessed advice about the database contents (before she knows which indices she wants to query). Then, the client makes a batch of Q non-adaptive queries, and the server makes at most T database probes per query (i.e., at most QT probes per batch). We show that $ST + QT = \widetilde{\Omega}(n)$.

For simplicity, we state the theorem in terms of batch PIR with advice, which we formally define in the full version of this paper [38]. This PIR model is in fact identical to single-server, multi-query, offline/online PIR, in which the client makes its queries non-adaptively, up to some syntactic differences.

Theorem 23 (Lower bound for batch PIR with advice). *Consider a computationally secure, single-server batch-PIR-with-advice scheme such that, on security parameter $\lambda \in \mathbb{N}$, database size $n \in \mathbb{N}$, and batch size $Q \in [n]$,*

- *the server stores the database in its original form,*
- *the client downloads S bits of advice, and*
- *the server probes at most QT database bits to answer a batch of Q queries.*

Then, for polynomially bounded $n = n(\lambda)$ and large enough λ, it holds that $ST + QT \geq \widetilde{\Omega}(n)$.

Theorem 23 shows that it is impossible to get additional speedups from PIR schemes that both (a) have the client store information, as in the offline/online PIR model, and (b) jointly process a batch of queries, as in the batch PIR model. An alternative interpretation of Theorem 23 suggests that adaptivity can "come for free" in the single-server setting: it requires no more online server time than standard batch PIR, as long as the client has at least $O(Q)$ storage.

We formally prove Theorem 23 in the full version of this paper [38].

Proof idea for Theorem 23. We prove this theorem via an incompressibility argument [3,41–43,50,95], by demonstrating that any batch PIR with advice scheme that defies this lower bound could be used to compress the database it is run on—thus, such a PIR scheme cannot exist. We make this argument in steps:

1. First, we define the multi-query Box Problem, an extension of Yao's Box Problem [95] in which the players iteratively open many boxes. Informally, the multi-query Box Problem encodes a game involving two players and an n-bit string $D = (D_1, \ldots, D_n)$:
   - Initially, the first player examines D and produces an S-bit advice string to be passed to the second player.
   - Then, the second player is given the S-bit advice string and a set of Q indices $\{i_1, \ldots, i_Q\}$. The player's goal is to output $(D_{i_1}, \ldots, D_{i_Q}) \in \{0,1\}^Q$. To solve this task, the second player may read at most QT bits of D. When the player reads a bit of D whose index lies in the challenge set $\{i_1, \ldots, i_Q\}$, we say that the player's query is a "violation" and we require that the player make at most V violations.

   The two players win the game if the second player recovers $(D_{i_1}, \ldots, D_{i_Q})$. We say that a strategy ϵ-solves the multi-query Box Problem if it allows the players to win with probability at least ϵ.
2. With an incompressibility argument, we prove that any strategy that ϵ-solves the multi-query Box Problem with a large enough Q and a small enough V must satisfy that $ST + QT = \tilde{\Omega}(\epsilon n)$.
3. We show that an efficient batch-PIR-with-advice scheme for Q queries, with advice length S and per-query online time T, gives a good solution to the multi-query Box Problem. More specifically, given any such PIR scheme, we devise the following strategy for the multi-query Box Problem:
   - Both players treat the n-bit input string D as a database, that the first player examines and that the second player must recover at Q points.
   - Initially, the first player computes and outputs the S-bit advice string that the batch-PIR-with-advice scheme would have produced on this database.
   - Then, the second player takes in the S-bit advice string and Q database indices to retrieve. The second player retrieves these Q database values by executing the batch PIR scheme with the advice—probing at most QT database indices in total, across all Q queries.

   In this construction, the second player probes each index in the challenge set with low probability. (Otherwise, the PIR scheme would leak which values the player is reading from what indices are probed.) We show that this strategy $(1/2 - \mathrm{negl}(\lambda))$-solves the multi-query Box Problem with at most $2Q^2T/n$ violations. The bounds on any algorithm that solves the multi-query Box Problem must also apply to the PIR scheme, giving that $ST + QT = \tilde{\Omega}(n)$.□

7 Conclusion

We construct new single-server PIR schemes that have sublinear amortized total server time. A number of related problems remain open:

– Is it possible to match the performance of our PIR scheme based on fully homomorphic encryption (Sect. 5) while using simpler assumptions?
– Can we construct single-server PIR schemes for many adaptive queries that achieve optimal $\widetilde{O}_\lambda(1)$ communication, $\widetilde{O}_\lambda(n^{1/2})$ amortized server time, and $\widetilde{O}_\lambda(n^{1/2})$ client storage? Our scheme from Sect. 5 has larger communication $\widetilde{O}_\lambda(n^{1/2})$. One approach would be to design puncturable pseudorandom sets [39,89] with short descriptions that support both insertions and deletions.
– Our lower bounds in Sect. 6 only apply to PIR schemes in which the server stores the database in unencoded form. Can we beat these bounds by having the server store the database in some encoded form [14]?

Acknowledgements. We thank David Wu and Yuval Ishai for reading an early draft of this work and for their helpful suggestions on how to improve it. We thank Yevgeniy Dodis, Siyao Guo, and Sandro Coretti for answering questions about presampling. We deeply appreciate the support and technical advice that Dan Boneh gave on this project from the very start. Part of this work was done when the third author was a student at Stanford University. This work was supported in part by the National Science Foundation (Award CNS-2054869), a gift from Google, a Facebook Research Award, and the Fintech@CSAIL Initiative, as well as the National Science Foundation Graduate Research Fellowship under Grant No. 1745302 and an EECS Great Educators Fellowship.

References

1. Aguilar-Melchor, C., Barrier, J., Fousse, L., Killijian, M.O.: XPIR: private information retrieval for everyone. PoPETs **2**, 155–174 (2016)
2. Aiello, W., Bhatt, S., Ostrovsky, R., Rajagopalan, S.R.: Fast verification of any remote procedure call: short witness-indistinguishable one-round proofs for NP. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 463–474. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-45022-X_39
3. Akshima, Cash, D., Drucker, A., Wee, H.: Time-space tradeoffs and short collisions in Merkle-Damgård hash functions. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020. LNCS, vol. 12170, pp. 157–186. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-56784-2_6
4. Ali, A., et al.: Communication–computation trade-offs in PIR. In: USENIX Security, pp. 1811–1828. USENIX Association (2021)
5. Ambainis, A.: Upper bound on the communication complexity of private information retrieval. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) ICALP 1997. LNCS, vol. 1256, pp. 401–407. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-63165-8_196
6. Angel, S., Chen, H., Laine, K., Setty, S.T.V.: PIR with compressed queries and amortized query processing. In: S&P (2018)
7. Angel, S., Setty, S.: Unobservable communication over fully untrusted infrastructure. In: SOSP, pp. 551–569 (2016)
8. Backes, M., Kate, A., Maffei, M., Pecina, K.: ObliviAd: provably secure and practical online behavioral advertising. In: S&P (2012)
9. Barak, B., et al.: On the (im)possibility of obfuscating programs. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 1–18. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_1

10. Batcher, K.E.: Sorting networks and their applications. In: AFIPS, p. 307–314. Association for Computing Machinery (1968)
11. Beimel, A., Ishai, Y.: Information-theoretic private information retrieval: a unified construction. In: Orejas, F., Spirakis, P.G., van Leeuwen, J. (eds.) ICALP 2001. LNCS, vol. 2076, pp. 912–926. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-48224-5_74
12. Beimel, A., Ishai, Y., Kushilevitz, E., Raymond, J.: Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval. In: FOCS, pp. 261–270. IEEE Computer Society (2002)
13. Beimel, A., Ishai, Y., Malkin, T.: Reducing the servers computation in private information retrieval: PIR with preprocessing. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 55–73. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_4
14. Beimel, A., Ishai, Y., Malkin, T.: Reducing the servers' computation in private information retrieval: PIR with preprocessing. J. Cryptol. **17**, 125–151 (2004)
15. Bell, J.H., Bonawitz, K.A., Gascón, A., Lepoint, T., Raykova, M.: Secure single-server aggregation with (poly) logarithmic overhead. In: CCS (2020)
16. Bell, S., Komisarczuk, P.: An analysis of phishing blacklists: Google Safe Browsing, OpenPhish, and PhishTank. In: ACSW (2020)
17. Bentley, J.L., Saxe, J.B.: Decomposable searching problems I: static-to-dynamic transformation. J. Algorithms **1**, 301–358 (1980)
18. Biehl, I., Meyer, B., Wetzel, S.: Ensuring the integrity of agent-based computations by short proofs. In: Rothermel, K., Hohl, F. (eds.) MA 1998. LNCS, vol. 1477, pp. 183–194. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0057658
19. Blackwell, K., Wootters, M.: A note on the permuted puzzles toy conjecture. arXiv preprint arXiv:2108.07885 (2021)
20. Boneh, D.: The decision Diffie-Hellman problem. In: Buhler, J.P. (ed.) ANTS 1998. LNCS, vol. 1423, pp. 48–63. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0054851
21. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015. LNCS, vol. 9057, pp. 337–367. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-46803-6_12
22. Boyle, E., Gilboa, N., Ishai, Y.: Function secret sharing: improvements and extensions. In: CCS, pp. 1292–1303. ACM (2016)
23. Boyle, E., Holmgren, J., Ma, F., Weiss, M.: On the security of doubly efficient PIR. Cryptology ePrint Archive, Report 2021/1113 (2021)
24. Boyle, E., Holmgren, J., Weiss, M.: Permuted puzzles and cryptographic hardness. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11892, pp. 465–493. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36033-7_18
25. Boyle, E., Ishai, Y., Pass, R., Wootters, M.: Can we access a database both locally and privately? In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10678, pp. 662–693. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70503-3_22
26. Boyle, E., Naor, M.: Is there an oblivious RAM lower bound? In: ITCS (2016)
27. Brakerski, Z., Vaikuntanathan, V.: Fully homomorphic encryption from ring-LWE and security for key dependent messages. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 505–524. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_29
28. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_28

29. Canetti, R., Holmgren, J., Richelson, S.: Towards doubly efficient private information retrieval. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017. LNCS, vol. 10678, pp. 694–726. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70503-3_23
30. Chang, Y.-C.: Single database private information retrieval with logarithmic communication. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 50–61. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27800-9_5
31. Chen, H., Huang, Z., Laine, K., Rindal, P.: Labeled PSI from fully homomorphic encryption with malicious security. In: CCS, pp. 1223–1237 (2018)
32. Cheng, R., et al.: Talek: private group messaging with hidden access patterns. In: ACSAC, pp. 84–99. ACM (2020)
33. Chor, B., Gilboa, N.: Computationally private information retrieval (extended abstract). In: STOC, pp. 304–313. ACM (1997)
34. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. In: FOCS, pp. 41–50. IEEE Computer Society (1995)
35. Chor, B., Goldreich, O., Kushilevitz, E., Sudan, M.: Private information retrieval. J. ACM **45**, 965–981 (1998)
36. Coretti, S., Dodis, Y., Guo, S.: Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10991, pp. 693–721. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96884-1_23
37. Coretti, S., Dodis, Y., Guo, S., Steinberger, J.: Random oracles and non-uniformity. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018. LNCS, vol. 10820, pp. 227–258. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78381-9_9
38. Corrigan-Gibbs, H., Henzinger, A., Kogan, D.: Single-server private information retrieval with sublinear amortized time. Cryptology ePrint Archive, Report 2022/081 (2022)
39. Corrigan-Gibbs, H., Kogan, D.: Private information retrieval with sublinear online time. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. LNCS, vol. 12105, pp. 44–75. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-45721-1_3
40. Dauterman, E., Feng, E., Luo, E., Popa, R.A., Stoica, I.: DORY: an encrypted search system with distributed trust. In: OSDI, pp. 1101–1119. USENIX Association (2020)
41. De, A., Trevisan, L., Tulsiani, M.: Time space tradeoffs for attacks against one-way functions and PRGs. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 649–665. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14623-7_35
42. Dodis, Y., Guo, S., Katz, J.: Fixing cracks in the concrete: random oracles with auxiliary input, revisited. In: Coron, J.-S., Nielsen, J.B. (eds.) EUROCRYPT 2017. LNCS, vol. 10211, pp. 473–495. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-56614-6_16
43. Dodis, Y., Haitner, I., Tentes, A.: On the instantiability of hash-and-sign RSA signatures. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 112–132. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28914-9_7
44. Dodis, Y., Halevi, S., Rothblum, R.D., Wichs, D.: Spooky encryption and its applications. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 93–122. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_4
45. Döttling, N., Garg, S., Ishai, Y., Malavolta, G., Mour, T., Ostrovsky, R.: Trapdoor hash functions and their applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019. LNCS, vol. 11694, pp. 3–32. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26954-8_1

46. Dvir, Z., Gopi, S.: 2-server PIR with subpolynomial communication. J. ACM **63**, 1–15 (2016)
47. Dwork, C., Langberg, M., Naor, M., Nissim, K., Reingold, O.: Succinct proofs for NP and Spooky interactions (2004)
48. Dwork, C., Naor, M., Rothblum, G.N.: Spooky interaction and its discontents: compilers for succinct two-message argument systems. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016. LNCS, vol. 9816, pp. 123–145. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53015-3_5
49. Efremenko, K.: 3-query locally decodable codes of subexponential length. SIAM J. Comput. **41**, 1694–1703 (2012)
50. Gennaro, R., Trevisan, L.: Lower bounds on the efficiency of generic cryptographic constructions. In: FOCS, pp. 305–313 (2000)
51. Gentry, C.: A fully homomorphic encryption scheme. Ph.D. thesis, Stanford University (2009)
52. Gentry, C., Halevi, S.: Compressible FHE with applications to PIR. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019. LNCS, vol. 11892, pp. 438–464. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36033-7_17
53. Gentry, C., Ramzan, Z.: Single-database private information retrieval with constant communication rate. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 803–815. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_65
54. Gentry, C., Sahai, A., Waters, B.: Homomorphic encryption from learning with errors: conceptually-simpler, asymptotically-faster, attribute-based. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013. LNCS, vol. 8042, pp. 75–92. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40041-4_5
55. Gilboa, N., Ishai, Y.: Distributed point functions and their applications. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 640–658. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_35
56. Goldreich, O., Karloff, H., Schulman, L., Trevisan, L.: Lower bounds for linear locally decodable codes and private information retrieval. In: CCC (2002)
57. Goldreich, O.: Foundations of Cryptography. Cambridge University Press, Cambridge (2001)
58. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious RAMs. J. ACM **43**, 431–473 (1996)
59. Goldwasser, S., Micali, S.: Probabilistic encryption. J. Comput. Syst. Sci. **28**, 270–299 (1984)
60. Green, M., Ladd, W., Miers, I.: A protocol for privately reporting ad impressions at scale. In: CCS, pp. 1591–1601. ACM (2016)
61. Groth, J., Kiayias, A., Lipmaa, H.: Multi-query computationally-private information retrieval with constant communication rate. In: Nguyen, P.Q., Pointcheval, D. (eds.) PKC 2010. LNCS, vol. 6056, pp. 107–123. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13013-7_7
62. Gupta, T., Crooks, N., Mulhern, W., Setty, S., Alvisi, L., Walfish, M.: Scalable and private media consumption with Popcorn. In: NSDI, pp. 91–107 (2016)
63. Hamlin, A., Ostrovsky, R., Weiss, M., Wichs, D.: Private anonymous data access. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019. LNCS, vol. 11477, pp. 244–273. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17656-3_9
64. Henry, R.: Polynomial batch codes for efficient IT-PIR. PoPETs (2016)
65. Henry, R., Huang, Y., Goldberg, I.: One (block) size fits all: PIR and SPIR with variable-length records via multi-block queries. In: NDSS. The Internet Society (2013)

66. Henry, R., Olumofin, F.G., Goldberg, I.: Practical PIR for electronic commerce. In: CCS, pp. 677–690. ACM (2011)
67. Huang, Y., Evans, D., Katz, J.: Private set intersection: are garbled circuits better than custom protocols? In: NDSS. The Internet Society (2012)
68. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Batch codes and their applications. In: STOC, pp. 262–271. ACM (2004)
69. Jacob, R., Larsen, K.G., Nielsen, J.B.: Lower bounds for oblivious data structures. In: SODA, pp. 2439–2447. SIAM (2019)
70. Juels, A.: Targeted advertising ... and privacy too. In: Naccache, D. (ed.) CT-RSA 2001. LNCS, vol. 2020, pp. 408–424. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45353-9_30
71. Kalai, Y.T., Raz, R., Rothblum, R.D.: How to delegate computations: the power of no-signaling proofs. In: STOC, pp. 485–494 (2014)
72. Kogan, D., Corrigan-Gibbs, H.: Private blocklist lookups with Checklist. In: USENIX Security (2021)
73. Komargodski, I., Lin, W.-K.: A logarithmic lower bound for oblivious RAM (for all parameters). In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12828, pp. 579–609. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84259-8_20
74. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: single database, computationally-private information retrieval. In: FOCS, pp. 364–373. IEEE (1997)
75. Larsen, K.G., Nielsen, J.B.: Yes, there is an oblivious RAM lower bound! In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018. LNCS, vol. 10992, pp. 523–542. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_18
76. Larsen, K.G., Simkin, M., Yeo, K.: Lower bounds for multi-server oblivious RAMs. In: Pass, R., Pietrzak, K. (eds.) TCC 2020. LNCS, vol. 12550, pp. 486–503. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64375-1_17
77. Lipmaa, H.: An oblivious transfer protocol with log-squared communication. In: Zhou, J., Lopez, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005). https://doi.org/10.1007/11556992_23
78. Lipmaa, H.: First CPIR protocol with data-dependent computation. In: Lee, D., Hong, S. (eds.) ICISC 2009. LNCS, vol. 5984, pp. 193–210. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14423-3_14
79. Lueks, W., Goldberg, I.: Sublinear scaling for multi-client private information retrieval. In: Böhme, R., Okamoto, T. (eds.) FC 2015. LNCS, vol. 8975, pp. 168–186. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-47854-7_10
80. Mockapetris, P.: Domain names - concepts and facilities. RFC 1034 (1987). http://www.rfc-editor.org/rfc/rfc1034.txt
81. Mughees, M.H., Chen, H., Ren, L.: OnionPIR: response efficient single-server PIR. In: CCS (2021)
82. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
83. Patel, S., Persiano, G., Yeo, K.: Private stateful information retrieval. In: CCS, pp. 1002–1019 (2018)
84. Persiano, G., Yeo, K.: Limits of preprocessing for single-server PIR. In: SODA (2022)
85. Pinkas, B., Schneider, T., Zohner, M.: Faster private set intersection based on OT extension. In: USENIX Security, pp. 797–812. USENIX Association, San Diego (2014)

86. Pinkas, B., Schneider, T., Zohner, M.: Scalable private set intersection based on OT extension. ACM Trans. Priv. Secur. **21**, 1–35 (2018)
87. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM **56**, 1–40 (2009)
88. Servan-Schreiber, S., Hogan, K., Devadas, S.: AdVeil: a private targeted-advertising ecosystem. Cryptology ePrint Archive, Report 2021/1032 (2021)
89. Shi, E., Aqeel, W., Chandrasekaran, B., Maggs, B.: Puncturable pseudorandom sets and private information retrieval with near-optimal online bandwidth and time. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021. LNCS, vol. 12828, pp. 641–669. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-84259-8_22
90. Stark, E.M.: Splitting up trust, 14 September 2021. https://emilymstark.com/2021/09/14/splitting-up-trust.html
91. Tauman Kalai, Y., Raz, R., Rothblum, R.D.: Delegation for bounded space. In: STOC, pp. 565–574 (2013)
92. Unruh, D.: Random oracles and auxiliary input. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 205–223. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74143-5_12
93. Wehner, S., de Wolf, R.: Improved lower bounds for locally decodable codes and private information retrieval. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1424–1436. Springer, Heidelberg (2005). https://doi.org/10.1007/11523468_115
94. Woodruff, D., Yekhanin, S.: A geometric approach to information-theoretic private information retrieval. In: CCC. IEEE (2005)
95. Yao, A.: Coherent functions and program checkers. In: STOC (1990)
96. Yekhanin, S.: Towards 3-query locally decodable codes of subexponential length. J. ACM **55**, 1–16 (2008)