Secret-Key PIR from One-Way Functions

Nir Bitansky* Noam Mazor*

May 8, 2026

Abstract

In secret-key private information retrieval (SK-PIR), the client in an offline phase processes the database using a short secret key. In the online phase the client could then use the secret key to make queries to the server, without revealing the entries accessed, and using only sublinear communication $o(N)$ in the database size N. While (non-SK) PIR requires public-key cryptography, recent work provides evidence that SK-PIR may not. In particular, Chen, Ishai, Mour, and Rosen (STOC 26) construct SK-PIR with communication N^ε, for any ε, from high-noise LPN, which is not known to imply public-key cryptography.

We construct SK-PIR with online communication $\tilde{O}(\sqrt{N})$, under the minimal assumption of one-way functions. More generally we can achieve client-to-server communication $\tilde{O}(N_c)$ and server-to-client communication $\tilde{O}(N_s)$ as long as $N_c \cdot N_s \geq N$.

Our construction is simple and is based on garbled circuits with an *uncorrelated input encoding* property, which is satisfied by schemes from the literature.

1 Introduction

Private information retrieval (PIR) enables a client to retrieve an item from a database $D \in \{0,1\}^N$ without revealing which item is accessed, and using only one message in each direction and sublinear communication $o(N)$ in total. Since its introduction, in the multi-server [CGKS98] and single-server settings [KO97], it has become an essential tool in the context of *sublinear cryptography.*

A central question in the theoretical study of PIR is the required computational assumptions. In the multi-server setting, unconditional security is possible [CGKS98] and tightly connected to the question of *locally decodable codes* [KT00]. Assuming one-way functions, two-server (computationally-private) PIR is possible with communication $O(\lambda \log N)$ [BGI15], for security parameter λ. In the single-server setting, PIR is known to imply two-message oblivious transfer [DMO00], which in turn implies public-key encryption, as well as collision-resistant hashing [IKO05], whereas constructions with communication $\mathrm{poly}(\lambda, \log N)$ are known from a variety of number-theoretic and lattice assumptions (essentially any assumption that suffices for linearly-homomorphic encryption [KO97]).

Secret-Key PIR. A setting where public-key cryptography is not known to be necessary is that of *secret-key PIR with preprocessing* (SK-PIR). Here in an offline phase the client processes the database D, using a short secret key $sk \in \{0,1\}^\lambda$, sends the processed database $\widehat{D}$ to the server,

*New York University. {nbitansky,noammaz}@gmail.com

and keeps sk. Then in the online phase, the client uses sk to generate its database queries. As in plain PIR, we require sublinear communication $o(N)$ per query, and require that security holds for an unbounded polynomial number of queries.

The notion of SK-PIR was originally considered for the sake of *sublinear server complexity* (aka *doubly-efficient PIR*) [BIM04] and was initially constructed under a non-standard assumption regarding *permuted Reed-Muller codes* [BIPW17; CHR17]. (In a later breakthrough, doubly-efficient PIR was constructed in the public-key setting from Ring LWE [LMW23].)

Forgoing sublinear server complexity, and focusing on communication complexity, Chen, Ishai, Mour, and Rosen [CIMR26] construct SK-PIR with communication N^ε, for any constant ε, from *high-noise* LPN, which is not known to imply any form of public-key cryptography. Indeed, in terms of lower bounds, the only limitation in constructing SK-PIR is one-way functions [BIPW17].

Are one-way functions sufficient for SK-PIR?

1.1 Our Result

We answer the above question in the affirmative, showing that one-way functions suffice for SK-PIR with communication $\sqrt{N} \cdot \lambda \cdot \mathrm{polylog}(N)$. More generally, we show:

Theorem 1.1. *Assuming one-way functions, for any N_c, N_s such that $N_c \cdot N_s = N$, there exists SK-PIR for length N databases, with server-to-client communication N_s and client-to-server communication $N_c \cdot \lambda \cdot \mathrm{polylog}(N)$.*

The size of the processed database $\widehat{D}$ stored by the server is $O(N \cdot \lambda)$ and its running time per-query is $N \cdot \mathrm{poly}(\lambda, \log N)$. Our construction is simple and is based on circuit garbling with an *uncorrelated input encoding* property, which we prove to be satisfied by existing garbling schemes from the literature.

Main Open Question: In plain (non-SK) PIR schemes compression can be amplified by recursive composition [KO97]. This also turns out to be the case for the SK-PIR scheme in [CIMR26] where the database processing is a linear function, and leads to communication N^ε for an arbitrarily small ε. In contrast, in our SK-PIR, recursive composition does not seem to be applicable, and hence our compression only goes as far as $\sqrt{N}$.

Whether compression could be pushed further or there is a barrier preventing further compression (assuming only one-way functions) is left as an open question.

Additional Background on PIR. PIR schemes have been extensively studied from both a theoretical and a practical perspectives. In this paper, we focus on the question of computational assumptions, and specifically on SK-PIR from one-way functions. We refer the reader to [CIMR26, Section 3.1] for a comprehensive account of relevant PIR literature.

1.2 Technical Overview

We now explain the basic ideas behind our PIR protocol.

Starting Point: Short Server Message Using Garbling. We start from explaining the simplest version of the protocol where the client's online message grows with the database size N, but the server's online message is independent of the database size. We observe that this can be easily done using garbled circuits [Yao86]. The construction is essentially the same as the construction of (non-compact) functional encryption [SS10], except that we use *oblivious* garbling [BHR12], where the server does not obtain the result in the clear.

In oblivious garbling, given a Boolean circuit $C : \{0,1\}^n \to \{0,1\}$, a garbler $\mathsf{Gar}(C)$ generates a garbled circuit $\widetilde{C}$, input labels $\ell = \{\ell_{i,b} : i \in [n], b \in \{0,1\}\}$, and a decoding key d. Given the garbled circuit $\widetilde{C}$ and the labels $\ell_x = \{\ell_{i,x_i}\}$ corresponding to any input $x \in \{0,1\}^n$, it is possible to evaluate the garbled circuit, so that the result $\widetilde{C}(\ell_x)$ can be decoded to $C(x)$ using the decoding key d. In terms of security, $(\widetilde{C}, \ell_x)$ reveal nothing about x, and can be simulated from C alone (here is where oblivious garbling differs from plain garbling, which also reveals $C(x)$). In terms of efficiency, $(\widetilde{C}, \ell_x)$ are essentially the size of the circuit C, and the result of evaluation $\widetilde{C}(\ell_x)$ is only a single bit.

The protocol works as follows:

- The client secret key will consist of $2N$ symmetric encryption keys $k = \{k_{i,b} : i \in [N], b \in \{0,1\}\}$. In the actual scheme the keys k will be generated using a pseudorandom function (PRF), so that the effective secret key is a short PRF seed. For simplicity, through the rest of the introduction, we'll ignore this extra step, and address the keys k directly.
- To preprocess a database $D \in \{0,1\}^N$, we provide the server with the N keys $k_D = \left\{k_{i,D[i]}\right\}_i$ corresponding to the database entries, as well as the database D itself.
- To generate a query for an entry $i^* \in [N]$, the client garbles the circuit $C : \{0,1\}^N \times \{0,1\}^{\log N} \to \{0,1\}$ that given $(D, i) \in \{0,1\}^N \times [N]$ outputs $D[i]$. It then sends the server the garbled $\widetilde{C}$, encryptions $\{\mathsf{Enc}(k_{j,b}, \ell_{j,b}) : j \in [N], b \in \{0,1\}\}$ of the $2N$ labels corresponding to the database under the corresponding keys k, and the $\log N$ labels $\ell_{\langle i^* \rangle}$ corresponding to (the binary encoding of) its desired index i^*. The client keeps the decoding key d.
- To answer the query, the server uses its keys k_D to obtain the corresponding labels ℓ_D and then evaluates the garbled circuit $\widetilde{C}(\ell_D, \ell_{\langle i^* \rangle})$, sends the resulting bit to the client, who then decodes $D[i^*]$ using its decoding key d.

To argue security, note that the server is oblivious of the keys $k_{\overline{D}} = \left\{k_{i,1-D[i]}\right\}_i$ and accordingly we can simulate the encryptions to the corresponding labels $\ell_{\overline{D}}$. Once this is done, the server's view only includes the garbled circuit and the labels $(\ell_D, \ell_{\langle i^* \rangle})$ corresponding to a single input (D, i^*), and hence by obliviousness, can be simulated independently of i^*.

A Natural Approach Toward Compression. In the above solution, while the server sends only a single bit per query, the client's message is as large as the database. However, there seems to be a natural approach to balancing the communication between the client and server and making both $\sqrt{N}$ (or more generally N_c, N_s for $N_c \cdot N_s = N$). Specifically, we can view the database as a $\sqrt{N} \times \sqrt{N}$ matrix $D[i,j]$ and apply the same garbled circuit on each of its rows.

In more detail:

- The client secret key consists of $2N$ symmetric encryption keys $k = \left\{k_{i,j,b} : i,j \in [\sqrt{N}], b \in \{0,1\}\right\}$.

- To preprocess a database $D \in \{0,1\}^{\sqrt{N}\times\sqrt{N}}$, we provide the server with the N keys $k_D = \left\{k_{i,j,D[i,j]}\right\}_{i,j}$ corresponding to the database entries, as well as the database D itself.
- To generate a query for an entry (i^*, j^*), the client garbles the circuit $C : \{0,1\}^{\sqrt{N}} \times \{0,1\}^{\log\sqrt{N}} \to \{0,1\}$ that is only meant to take as input a single row $D[i,\cdot] \in \{0,1\}^{\sqrt{N}}$ and an index $j \in [\sqrt{N}]$, and output $D[i,j]$. The client then sends the server the garbled $\widetilde{C}$, encryptions $\{\mathsf{Enc}(k_{i^*,j,b}, \ell_{j,b}) : j \in [\sqrt{N}], b \in \{0,1\}\}$ of the $2\sqrt{N}$ labels corresponding to the database row $D[i^*,\cdot]$ under the keys $\{k_{i^*,j,b}\}_{j,b}$, corresponding to row i^*, and the $\log\sqrt{N}$ labels $\ell_{\langle j^*\rangle}$ corresponding to (the binary encoding of) column j^*. The client keeps the decoding key d.
- To answer the query, the server, for every row $i \in [\sqrt{N}]$, uses its keys $\left\{k_{i,j,D[i,j]}\right\}_j$ attempting to decrypt the corresponding labels $\ell_{D[i,\cdot]} = \left\{\ell_{i,j,D[i,j]}\right\}_j$ and then evaluates the garbled circuit $\widetilde{C}(\ell_{D[i,\cdot]}, \ell_{\langle j^*\rangle})$, and obtains a resulting bit b_i. Eventually, it sends $b_1, \ldots, b_{\sqrt{N}}$ to the client. The client then decodes the bit b_{i^*} using its decoding key d (discarding all other bits b_i).

In terms of functionality the above works perfectly. Security, however, is not as clear. While we can argue that j^* is hidden using a similar argument to the previous protocol, i^* the second part of the index may leak. Indeed, it may be that given different keys $k_0 \neq k_1$ and an encryption $\mathsf{Enc}(k_b, m)$ under one of them, it is possible to determine which one was used. Furthermore, even if this is not the case, it may be easy to identify an evaluation of the garbled circuit $\widetilde{C}$ on *incorrect labels* ℓ', resulting from decryption with non-matching keys:

$$\ell' = \mathsf{Dec}(k_{i,j,D[i,j]}, \mathsf{Enc}(k_{i^*,j,D[i^*,j]}, \ell_{i^*,j,D[i^*,j]})) \ .$$

Uncorrelated Input Encoding. To deal with the above, we consider a strengthening of oblivious security, which we call *uncorrelated input encoding*. Here we require that for any input x, given a garbled circuit $\widetilde{C}$ (but withholding the decoding key d), the labels ℓ_x corresponding to x are indistinguishable from uniformly random labels that have no correlation with the garbled circuit:

$$\widetilde{C}, \ell_x \approx_c \widetilde{C}, \mathbf{U} \ .$$

This notion can be derived from the notion of *blind* garbled circuits [BLSV18] and both are satisfied by *point and permute* garbling schemes [BMR90]. We include the construction and security proof for completeness.

In addition, we use a symmetric key encryption scheme that is *regular* in the sense that for any key k, encryptions of random messages $\mathsf{Enc}_k(\mathbf{U})$ are distributed according to a fixed distribution, independent of k. For instance, we can use a pseudorandom function F to define $\mathsf{Enc}_k(m; r) = r, F_k(r) \oplus m$ to guarantee that $\mathsf{Enc}_k(\mathbf{U})$ is always uniformly random.

Given these two adjustments, we can now recover the security argument by a hybrid argument. Analogously to the previous argument, we first invoke encryption security with respect to the keys $k_{\overline{D}[i^*,\cdot]}$, of which the server is oblivious. Specifically, instead of encrypting the labels $\ell_{\overline{D}[i^*,\cdot]}$, we now encrypt under these keys independent uniformly random strings.

At this point, the server's view only includes the labels corresponding to the input $(\ell_{D[i^*,\cdot]}, \ell_{\langle j^*\rangle})$, as well as the garbled $\widetilde{C}$. Invoking the uncorrelated input encoding property, we can now also

replace $(\ell_{D[i^*,\cdot]}, \ell_{\langle j^* \rangle})$ with uniformly random strings. This in particular applies to the encrypted labels $\ell_{D[i^*,\cdot]}$. Now, all of the $2\sqrt{N}$ encrypted labels are uniformly random. By the regularity of the symmetric encryption scheme, these encryptions are distributed independently of the keys $\{k_{i^*,j,b}\}_{j,b}$ that were used. In fact, the view of the server is now completely independent of (i^*, j^*).

Point and Permute Garbling and Uncorrelated Input Encoding. We now describe how point and permute garbling [BMR90] satisfy the uncorrelated input property. The proof is essentially the same as the proof of *blindness* in [BLSV18] and is given for the sake of completeness. It is in fact possible to generically derive uncorrelated input encoding from blindness (see further explanation below).

In point and permute garbling, every gate g in the circuit C is associated with two random secret keys s_0^g, s_1^g and a random permutation bit π^g. If g takes its input from two gates a, b and corresponds to a Boolean operation $G(\cdot,\cdot)$, we generate a garbled table:

$$T^g = \left(F_{s_\alpha^a}(L, g, \alpha, \beta) \oplus F_{s_\beta^b}(R, g, \alpha, \beta) \oplus (s_{\tilde{\gamma}}^g, \tilde{\gamma}) \right)_{\alpha,\beta \in \{0,1\}} ,$$

where F is a pseudorandom function, $\gamma = G(\alpha \oplus \pi^a, \beta \oplus \pi^b)$, and $\tilde{\gamma} = \gamma \oplus \pi^g$.

For a computation $C(x)$, we denote by $v(g) \in \{0,1\}$ the output value of gate g in the computation. Then the garbled circuit invariant is that the evaluator learns for every gate g, $\tilde{v}(g) = v(g) \oplus \pi^g$ and the key $s_{\tilde{v}(g)}^g$. In particular, denoting the input gates by $1, \ldots, n$, the corresponding input labels consist of $\tilde{x}_i = x_i \oplus \pi^i$ and $s_{\tilde{x}_i}^i$. The decoding key d is the permutation bit π^o corresponding to the output wire o.

If the inputs of gate g are the outputs of gates a, b, then $(s_{\tilde{v}(g)}^g, \tilde{v}(g))$ is decoded from the entry $(\tilde{v}(a), \tilde{v}(b))$ in the table T^g. All the other *incorrect* entries $(\alpha, \beta) \neq (\tilde{v}(a), \tilde{v}(b))$ in the table T^g are pseudorandom as the keys $s_{1-\tilde{v}(a)}^a, s_{1-\tilde{v}(b)}^b$ are unknown (formally this is shown by a hybrid argument going in topological order from inputs to the output).

We then observe that considering this indistinguishable garbled circuit where all incorrect entries are sampled uniformly at random, the remaining entries are also distributed uniformly at random. To see this, we examine the tables in topological order, but now from the output gate to the input gates. We first note that the output permutation bit and key $(s_{\tilde{v}(o)}^o, \tilde{v}(o))$ are uniformly random and not used anywhere else (the corresponding decoding key π^o is kept by the client, and $s_{\tilde{v}(o)}^o$ is not used at all). Accordingly, the correct entry in T^o

$$F_{\ldots}(L, o, \cdots) \oplus F_{\ldots}(R, o, \cdots) \oplus (s_{\tilde{v}(o)}^o, \tilde{v}(o))$$

is uniformly random. We can then proceed to the next gate g and argue that $(s_{\tilde{v}(g)}^g, \tilde{v}(g))$ is uniformly random and accordingly the correct row in T^g is also uniformly random, and so on for each and every gate.

At this point, the garbled circuit $\widetilde{C}$ is in fact completely uniformly random and independent of the labels ℓ_x. Moreover the marginal distribution of the labels ℓ_x is also uniformly random so overall the real garbled circuit and labels $(\widetilde{C}, \ell_x)$ are computationally indistinguishable from uniformly random and independent circuit and labels $(\mathbf{U}', \mathbf{U})$, and in particular indistinguishable from $(\widetilde{C}, \mathbf{U})$ as required.

A Word on Blindness. Blind garbling, considered in [BLSV18], is defined similarly to plain garbling where the evaluator obtains the output $y = C(x)$, and $\widetilde{C}, \ell_x$ are simulated from y, with the extra requirement that if y is uniformly random then simulation is indistinguishable from uniformly random strings $(\mathbf{U}, \mathbf{U}')$. Uncorrelated garbling can be obtained generically from blinding by randomizing the output using a one-time pad d, kept as the decoding key, and considering a circuit that outputs $C(x) \oplus d$.

2 Preliminaries

All logarithms are taken in base 2. For $n \in \mathbb{N}$, let $[n] := \{1, \ldots, n\}$. Given a vector s of length n, let s_i denote its i-th entry, and $s_{\leq i}$ denote its first i entries, and $s_{>i}$ denote its last $n - i$ entries. For a distribution $\mathcal{P}$, let $d \leftarrow \mathcal{P}$ denote that d was sampled according to $\mathcal{P}$.

We consider circuits with fan-in 2 NAND gates. The size of a circuit C, denoted by $|C|$, is the number of gates in C (both input and inner gates).

Two distribution ensembles $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$, $Y = \{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable, denoted by $X \approx_c Y$, if for every non-uniform PPT A there exists a negligible function δ, such that

$$|\Pr[A(X_\lambda) = 1] - \Pr[A(Y_\lambda) = 1]| \leq \delta(\lambda).$$

When the ensembles are indexed with additional input $z \in \{0,1\}^*$, we say that $\{X_{\lambda,z}\}_{\lambda,z} \approx_c \{Y_{\lambda,z}\}_{\lambda,z}$ if for any polynomial p, and any sequence $\{z_\lambda\}_\lambda$ with $|z_\lambda| \leq p(\lambda)$, it holds that $\{X_{\lambda,z_\lambda}\}_\lambda \approx_c \{Y_{\lambda,z_\lambda}\}_\lambda$. We say that $X \equiv Y$ if for every λ, X_λ distributed identically to Y_λ.

2.1 Pseudorandom Functions

We next define pseudorandom functions. It is well known [HILL99; GGM84] that pseudorandom functions exist if one-way functions exist.

Definition 2.1 (Pseudorandom function)**.** *An efficiently computable family of functions* $F = \left\{F_k \colon \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\right\}_{k \in \{0,1\}^\lambda}$ *is a* pseudorandom function family (PRF) *if for every oracle-aided non-uniform PPT* A*, there exists a negligible function* μ*, such that for any* λ*,*

$$\left|\Pr\left[\mathcal{A}^{F_k}(1^\lambda) = 1\right] - \Pr\left[\mathcal{A}^{\Pi}(1^\lambda) = 1\right]\right| \leq \mu(\lambda),$$

where $k \leftarrow \{0,1\}^\lambda$ *and* $\Pi \colon \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}$ *is a uniformly random function.*

2.2 Private Information Retrieval

Finally, we define secret-key PIR.

Definition 2.2 (Secret-key PIR [CHR17; BIPW17; CIMR26])**.** *A* secret-key private information retrieval (SK-PIR) *consists of PPT algorithms* (Gen, Preprocess, Query, Answer, Decode) *such that:*

- $\mathsf{Gen}(1^\lambda, 1^N)$*: The key generation algorithm. Takes a security parameter* 1^λ *and database size* 1^N *and outputs a secret key* sk*.*
- $\mathsf{Preprocess}(sk, D)$*: The database encoding algorithm. Takes a secret key* sk *and a database* $D \in \{0,1\}^N$ *and outputs a database encoding* $\widehat{D}$*.*

- $\mathsf{Query}(sk, i)$: *The query algorithm. Takes a secret key sk and an index $i \in [N]$, and outputs a client message q and a decoding key $dec \in \{0,1\}^*$.*
- $\mathsf{Answer}(\widehat{D}, q)$: *The server answer algorithm. Takes a database encoding $\widehat{D}$ and a PIR-query q and returns an answer a.*
- $\mathsf{Decode}(a, i, dec)$: *The decoding algorithm. Takes a PIR-answer a, an index i and a decoding key dec and outputs a bit $b \in \{0,1\}$.*

Correctness: *The client outputs the correct value. For any $sk \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$, $D \in \{0,1\}^N$, $\widehat{D} \leftarrow \mathsf{Preprocess}(sk, D)$ and $i \in [N]$,*

$$\Pr[\mathsf{Decode}(a, i, dec) = D[i]] = 1,$$

where $(q, dec) \leftarrow \mathsf{Query}(sk, i)$ and $a \leftarrow \mathsf{Answer}(\widehat{D}, q)$.

Security: *For any oracle-aided non-uniform PPT algorithm $\mathcal{A}$ and any polynomial $N = N(\lambda)$, there exists a negligible function μ such that for any $\lambda \in \mathbb{N}$ and $D \in \{0,1\}^N$, it holds that*

$$\left|\Pr\left[\mathcal{A}^{\mathsf{Query}_q(sk,\cdot)}(1^\lambda, \widehat{D}) = 1\right] - \Pr\left[\mathcal{A}^{\mathsf{Query}'_q(sk,\cdot)}(1^\lambda, \widehat{D}) = 1\right]\right| \leq \mu(\lambda),$$

where $sk \leftarrow \mathsf{Gen}(1^\lambda, 1^N)$, $\widehat{D} \leftarrow \mathsf{Preprocess}(sk, D)$, $\mathsf{Query}_q(sk, i)$ samples $(q, d) \leftarrow \mathsf{Query}(sk, i)$ and outputs q, and $\mathsf{Query}'_q(sk, \cdot)$ is an oracle that ignores its input and outputs $\mathsf{Query}_q(sk, 1)$.

3 Uncorrelated Garbling

In this section we define uncorrelated garbling. In Section 3.1 we show a construction based on one-way functions.

Definition 3.1 (Uncorrelated Garbling)**.** *An uncorrelated garbling scheme consists of PPT algorithms $(\mathsf{Gar}, \mathsf{Eval})$ such that:*

- $(\widetilde{C}, \boldsymbol{\ell}, m) \leftarrow \mathsf{Gar}(1^\lambda, C)$*, given a security parameter λ and circuit $C\colon \{0,1\}^n \to \{0,1\}$, outputs a garbled circuit $\widetilde{C}$, $2n$ input labels $\ell = \{\ell_b^i : i \in [n], b \in \{0,1\}\}$, and decoding key d.*
- $b \leftarrow \mathsf{Eval}(C, \tilde{C}, \ell_x)$*, given a circuit C, a garbled circuit $\tilde{C}$ and a set ℓ_x of n labels, outputs a bit b.*

The scheme should satisfy:

Correctness: *Evaluation is decoded to the circuit value. For any circuit $C\colon \{0,1\}^n \to \{0,1\}$ and input $x \in \{0,1\}^n$,*

$$\Pr\left[d \oplus \mathsf{Eval}(C, \widetilde{C}, \ell_x) = C(x)\right] = 1 \ ,$$

where $(\widetilde{C}, \ell = \{\ell_b^i\}_{i,b}, d) \leftarrow \mathsf{Gar}(1^\lambda, C)$ and $\ell_x = \{\ell_{x_i}^i\}_i$.

Uncorrelated Input Encoding: *For any circuit C and input x the labels ℓ_x are pseudorandom, given the garbled circuit $\widetilde{C}$.*

$$\left\{\widetilde{C}, \ell_x\right\}_{\lambda,C,x} \approx_c \left\{\widetilde{C}, \mathbf{U}\right\}_{\lambda,C,x} \ ,$$

where $C : \{0,1\}^n \to \{0,1\}$ is a circuit $x \in \{0,1\}^n$, $(\tilde{C}, \ell = \{\ell_b^i\}_{i,b}) \leftarrow \mathsf{Gar}(1^\lambda, C)$, $\ell_x = \{\ell_{x_i}^i\}_i$, and $\mathbf{U}$ is uniformly random (of the same length as ℓ_x).

3.1 OWF to Uncorrelated Garbling

In this section we construct an uncorrelated garbling scheme from point and permute garbling, which in turn can be based on one-way functions. As mentioned in the introduction, the proof is similar to the proof that point and permutation garbling is *blind* [BLSV18] and is given here for the sake of completeness.

Theorem 3.2. *Assuming OWF, there exists uncorrelated garbling schemes. Furthermore, the size of the garbled circuit and lables is* $|(\widetilde{C}, \ell)| = O(|C| \cdot \lambda)$.

We next describe the standard *point and permute* garbed circuit [BMR90].

The Garbling Algorithm. $\mathsf{Gar}(1^\lambda, C)$. The algorithm is given a circuit $C : \{0,1\}^n \to \{0,1\}$ with m topologically ordered gates $1, \ldots, m$, such that $1 \ldots n$ are the input gates and m is the output gate, and all (non-input) gates are NAND. In what follows, we use a PRF $F_s : \{L, R\} \times [m] \times \{0,1\}^2 \to \{0,1\}^{\lambda+1}$.

The algorithm operates as follows:

1. For every gate $g \in [m]$, we sample two random PRF seeds and a random permutation bit:

$$s_0^g, s_1^g \leftarrow \{0,1\}^\lambda, \pi^g \leftarrow \{0,1\} \ .$$

2. For each non-input gate $g \in \{n+1, \ldots, m\}$ with input wires corresponding to gates a, b, we compute:

$$T^g = \left(F_{s_\alpha^a}(L, g, \alpha, \beta) \oplus F_{s_\beta^b}(R, g, \alpha, \beta) \oplus (s_{\tilde{\gamma}}^g, \tilde{\gamma}) \right)_{\alpha,\beta \in \{0,1\}} \ ,$$

   where $\gamma = \mathsf{NAND}(\alpha \oplus \pi^a, \beta \oplus \pi^b)$ and $\tilde{\gamma} = \gamma \oplus \pi^g$.

3. Output $\widetilde{C} = \{T^g\}_{g>n}$, input labels $\ell_{i,b} := (s_{\tilde{b}}^i, \tilde{b})$, where $\tilde{b} = b \oplus \pi^i$, and the decoding key $d := \pi^m$.

The Evaluation Algorithm. $\mathsf{Eval}(C, \widetilde{C}, \ell_x)$ The algorithm is given a garbled circuit $\widetilde{C}$ and labels $\ell_x = \{\ell_i\}_i$. The algorithm operates as follows:

1. For every input gate $i \in [n]$, let $\ell_i = (s^i, \tilde{\gamma}^i)$.

2. For each non-input gate $g \in \{n+1, \ldots, m\}$ with input wires corresponding to gates $a, b \leq g$, let

$$(s^g, \tilde{\gamma}^g) = T^g[\tilde{\gamma}^a, \tilde{\gamma}^b] \oplus \left(F_{s^a}(L, g, \tilde{\gamma}^a, \tilde{\gamma}^b) \oplus F_{s^b}(R, g, \tilde{\gamma}^a, \tilde{\gamma}^b) \right) \ .$$

3. Output $\tilde{\gamma}^m$.

Correctness. We start with proving the correctness of the scheme.

Lemma 3.3. *The scheme is correct: For any circuit $C\colon \{0,1\}^n \to \{0,1\}$ and input $x \in \{0,1\}^n$,*

$$\Pr\left[d \oplus \mathsf{Eval}(C, \widetilde{C}, \ell_x) = C(x)\right] = 1 \ .$$

Proof. Fix C and x, and for each gate g of C let $v(g)$ be the value of the gate in the computation $C(x)$. Let $(\widetilde{C}, \ell_x, d) \leftarrow \mathsf{Gar}(1^\lambda, C)$, and for each gate g let s_0^g, s_1^g, π_g be the values chosen by Gar in the execution. Consider an execution of $\mathsf{Eval}(C, \widetilde{C}, \ell_x)$.

We claim that the following invariant regarding the computed values of $(s^g, \tilde{\gamma}^g)$ holds in any step of Eval: For each gate g, it holds that $\tilde{\gamma}^g = \pi^g \oplus v(g)$, where $v(g) \in \{0,1\}$ denote the output value of gate g in the computation $C(x)$. Moreover, we have that $s^g = s^g_{\tilde{\gamma}^g}$, where s^g_b is the value chosen by Gar.

The proof is by induction on g. Indeed, for the input gates the invariant holds by construction. For every internal gate g with input wires corresponding to gates $a, b \leq g$, assume the invariant holds with respect to the inputs of a, b. Then by construction of T^g it holds that

$$T^g[\tilde{\gamma}^a, \tilde{\gamma}^a] = \left(F_{s^a}(L, g, \tilde{\gamma}^a, \tilde{\gamma}^b) \oplus F_{s^b}(R, g, \tilde{\gamma}^a, \tilde{\gamma}^b)\right) \oplus (s^g_{\tilde{\gamma}^g}, \tilde{\gamma}^g) \ ,$$

where $\tilde{\gamma}^g = \mathsf{NAND}(\tilde{\gamma}^a \oplus \pi^a, \tilde{\gamma}^a \oplus \pi^b) \oplus \pi^g = \mathsf{NAND}(v(a), v(b)) \oplus \pi^g = v(g) \oplus \pi^g$.

The correctness now holds by the choice of d to be the permutation bit π^m of the output gate. □

Uncorrelated Input Encoding. We next prove the security of the scheme.

Lemma 3.4. *For any circuit C and input x,*

$$\left\{\widetilde{C}, \ell_x\right\}_{\lambda,C,x} \approx_c \left\{\widetilde{C}, \mathbf{U}\right\}_{\lambda,C,x} \ ,$$

where $(\tilde{C}, \ell = \{\ell_b^i\}_{i,b}) \leftarrow \mathsf{Gar}(1^\lambda, C)$ and $\ell_x = \{\ell^i_{x_i}\}_i$.

We prove the following claim, from which the proof of Lemma 3.4 is immediate.

Claim 3.5. *For any circuit C and input x,*

$$\left\{\widetilde{C}, \ell_x\right\}_{\lambda,C,x} \approx_c \left\{\mathbf{U}', \ell_x\right\}_{\lambda,C,x} \ ,$$

where $(\tilde{C}, \ell = \{\ell_b^i\}_{i,b}) \leftarrow \mathsf{Gar}(1^\lambda, C)$, $\ell_x = \{\ell^i_{x_i}\}_i$, and $\mathbf{U}'$ is uniformly random (of the same length as $\widetilde{C}$).

Proof of Lemma 3.4. Since the marginal distribution of the labels ℓ_x chosen by Gar is uniform, it follows that

$$\left\{\widetilde{C}, \ell_x\right\}_{\lambda,C,x} \approx_c \left\{\mathbf{U}', \mathbf{U}\right\}_{\lambda,C,x} \approx_c \left\{\widetilde{C}, \mathbf{U}\right\}_{\lambda,C,x} \ ,$$

where $\mathbf{U}', \mathbf{U}$ are two independent uniformly random strings. □

We next prove Claim 3.5.

Proof of Claim 3.5. The proof is by a hybrid argument. Fix the circuit C and input x, and let n be the input length of C, and m the total number of gates. For each gate g, let $v(g) \in \{0,1\}$ denote the output value of gate g in the computation $C(x)$.

Consider a random execution of $\mathsf{Gar}(1^\lambda, C)$. For each gate $g \in [m]$ let s_0^g, s_1^g, π_g be the values chosen by Gar in the execution, and let $\tilde{\gamma}^g = \pi^g \oplus v(g)$. Fix any gate g with input wires corresponding to gates gates $a, b \leq g$.

Step 1: Incorrect Entries Are Pseudorandom. We first claim that for any *incorrect* entry $(\alpha, \beta) \neq (\tilde{\gamma}^a, \tilde{\gamma}^b)$, $T^g[\alpha, \beta]$ is pseudorandom. Specifically, consider the hybrid circuit $\widehat{C} = \left\{\widehat{T}^g\right\}_{g>n}$, where $\widehat{T}^g[\alpha, \beta] = T^g[\alpha, \beta]$ if $(\alpha, \beta) = (\tilde{\gamma}^a, \tilde{\gamma}^b)$, or chosen uniformly at random otherwise. Let $\widehat{\mathsf{Gar}}$ be the algorithm that given $1^\lambda, C$ and x, acts like Gar and computes $(\widetilde{C}, \ell)$, and additionally outputs $\widehat{C}$. We first claim that

$$\left\{\widetilde{C}, \ell_x\right\}_{\lambda,C,x} \approx_c \left\{\widehat{C}, \ell_x\right\}_{\lambda,C,x} , \tag{1}$$

where $(\widetilde{C}, \widehat{C}, \ell) \leftarrow \widehat{\mathsf{Gar}}(1^\lambda, C, x)$. Note that the entries in T^g that we replace with random value in $\widehat{C}$ are exactly the entries that are not used by $\mathsf{Eval}(C, \widetilde{C}, x)$. In particular, $\widehat{C}$ is independent of the incorrect seeds (i.e. s_α^g for $\alpha \neq \tilde{\gamma}^g$). To see that Equation (1) holds, consider the following hybrids G_i for every $n \leq i \leq m$.

$\mathbf{G_i}$. The distribution G_i is the output distribution of the following process. Sample $(\widetilde{C}, \widehat{C}, \ell) \leftarrow \widehat{\mathsf{Gar}}(1^\lambda, C, x)$ and output $\left(\left(\left\{\widehat{T}^g\right\}_{g\leq i}, \{T^g\}_{g>i}\right), \ell_x\right)$.

Then G_n is distributed as $(\widetilde{C}, \ell_x)$, while G_m is distributed as $(\widehat{C}, \ell_x)$. We next claim that G_i is indistinguishable from G_{i+1}. Toward this, let $g = i+1$ be a gate with input wires corresponding to gates a, b, and consider any incorrect entry $T^g[\alpha, \beta]$ with $(\alpha, \beta) \neq (\tilde{\gamma}^a, \tilde{\gamma}^b)$. Assume w.l.o.g that $\alpha \neq \tilde{\gamma}^a$. We first claim that G_i only depends on the truth-table of $F_{s_\alpha^a}$, and is otherwise independent of the key s_α^a. Indeed, the only part of $\widetilde{C}$ that depends on s_α^a is T^a, and specifically the two entries in T^a that correspond to the incorrect label (s_α^a, α). This means that in $\widehat{T}^a$ those entries are replaced with uniformly random value. Since $a \leq i$, this implies that G_i is independent of s_α^a given the truth table of $F_{s_\alpha^a}$. Now, noting that $F_{s_\alpha^a}(L, g, \alpha, \beta)$ is only used once in $\widetilde{C}$, the security of the PRF implies that $F_{s_\alpha^a}(L, g, \alpha, \beta)$ is indistinguishable from uniform given $\left(\left(\left\{\widehat{T}^g\right\}_{g\leq i}, \{T^g\}_{g>i+1}\right), \ell_x\right)$. The above holds for any incorrect entry $(\alpha, \beta) \neq (\tilde{\gamma}^a, \tilde{\gamma}^b)$ of T^g, implying that G_i is indistinguishable from G_{i+1}. Equation (1) follows.

Step 2: Correct Entries Are Random. Next, we replace the correct table entries. We claim that,

$$\left\{\widehat{C}, \ell_x\right\}_{\lambda,C,x} \equiv \{\mathbf{U}, \ell_x\}_{\lambda,C,x} . \tag{2}$$

This can be seen using a top-down hybrid strategy. For $i \in \{m, m-1, \ldots, n\}$, define

$$H_i = \left(\left(\left\{\widehat{T}^g\right\}_{n<g\leq i}, \left\{\mathbf{U}_{4(\lambda+1)}\right\}_{g>i}\right), \ell_x\right) .$$

Then H_m is distributed exactly as $(\widehat{C}, \ell_x)$, and H_n is distributed as $(\mathbf{U}, \ell_x)$. We show that H_i is distributed identically to H_{i-1} for every $i \in \{m, m-1, \ldots, n+1\}$.

Let i is the only gate whose table changes between H_i and H_{i-1}, and let a, b be the gates corresponding to its inputs. The only non-random entry of $\widehat{T}^i$ is $\widehat{T}^i[\tilde{\gamma}^a, \tilde{\gamma}^b]$, which equals to

$$\left(F_{s^a_{\tilde{\gamma}^a}}(L, i, \tilde{\gamma}^a, \tilde{\gamma}^b) \oplus F_{s^b_{\tilde{\gamma}^b}}(R, i, \tilde{\gamma}^a, \tilde{\gamma}^b) \oplus (s^i_{\tilde{\gamma}^i}, \tilde{\gamma}^i) \right).$$

We argue that this entry is distributed uniformly given the rest of the view in H_i. Specifically, the label $(s^i_{\tilde{\gamma}^i}, \tilde{\gamma}^i)$ is uniformly random, which implies that the entire value of $\widehat{T}^i$ is random. Indeed, the tables of all gates that take i as input are among $\widehat{T}^{i+1}, \ldots, \widehat{T}^m$, which have already been replaced with uniform random strings in H_i. (In the first transition from H_m to H_{m-1}, we use the fact that the decoding information π^m, as well as s^m, are not given as part of the garbled circuit.)

This completes the hybrid argument and the proof of Claim 3.5.

□

Proof of Theorem 3.2. The correctness and security hold by Lemmas 3.3 and 3.4. The overhead is $O(m\lambda)$ by construction. □

4 Uncorrelated Garbling to SK-PIR

We next construct the SK-PIR protocol. We prove the following theorem.

Theorem 4.1. *Assuming one-way functions, for any N_c, N_s such that $N_c \cdot N_s = N$, there exists SK-PIR for length N databases, with server-to-client communication N_s and client-to-server communication $N_c \cdot \lambda \cdot \mathrm{polylog}(N)$.*

By setting $N_s = N_c = \sqrt{N}$, we get the following corollary.

Corollary 4.2. *Assume the existence of a one-way function. Then there exists a SK-PIR for length N database, with communication $O(\sqrt{N} \cdot \lambda \cdot \mathrm{polylog}N)$.*

We start with describing the protocol. In the following, let $(\mathsf{Gar}, \mathsf{Eval})$ be the uncorrelated garbling scheme promised by Theorem 3.2, with labels of length $(\lambda + 1)$. Let $F_k \colon [N_s] \times [N_c] \times \{0,1\} \to \{0,1\}^\lambda$ be a PRF and let $(\mathsf{Enc}, \mathsf{Dec})$ be the encryption scheme defined by

$$\mathsf{Enc}(k, m) \equiv (r, E_k(r) \oplus m) \ ,$$

where $r \leftarrow \{0,1\}^\lambda$, and $E_k \colon \{0,1\}^\lambda \to \{0,1\}^{\lambda+1}$ is also a PRF.

Let $C \colon \{0,1\}^{N_c} \times \{0,1\}^{\log N_c} \to \{0,1\}$ be the circuit that given $z \in \{0,1\}^{N_c}$ and $j \in [N_c]$ (represented as a $\log N_c$-bits string) outputs z_j. For simplicity, we assume throughout that N_s and N_c are powers of 2.

The PIR Protocol:

- **Key Generation:** $\mathsf{Gen}(1^\lambda, 1^N)$ outputs a random PRF key $sk \leftarrow \{0,1\}^\lambda$.
- **Preprocessing:** $\mathsf{Preprocess}(sk, D)$ given a secret key sk and a database $D \in \{0,1\}^N$, interpreted as $D \in \{0,1\}^{N_s \times N_c}$:

1. For any $i \in [N_s], j \in [N_c], b \in \{0,1\}$, computes $k_{i,j,b} = F_{sk}(i,j,b)$
2. Outputs $\widehat{D} = \{(k_{i,j,b}, b) : i \in [N_s], j \in [N_c], b = D[i,j]\}$

- **Query:** $\mathsf{Query}(sk, i^*, j^*)$ given a secret key sk and indices (i^*, j^*):
  1. Samples $(\widetilde{C}, \ell, d) \leftarrow \mathsf{Gar}(1^\lambda, C)$.
  2. For every $j \in [N_c]$, $b \in \{0,1\}$, samples $\ell'_{j,b} \leftarrow \mathsf{Enc}(k_{i^*,j,b}, \ell_{j,b})$.
  3. Computes the labels $\ell_{\langle j^* \rangle} = \{\ell_{N_c+h,b} \colon h \in [\log N_c], b = \langle j^* \rangle_h\}$ corresponding to the input j^* (where $\langle j^* \rangle$ is interpreted as the binary representation of j^*).
  4. Outputs a client message $q = \left(\widetilde{C}, \left\{\ell'_{j,b}\right\}_{j,b}, \ell_{\langle j^* \rangle}\right)$ and a decoding key $dec = d$.

- **Answer:** $\mathsf{Answer}(\widehat{D}, q)$ given a database encoding $\widehat{D}$ and a query $q = \left(\widetilde{C}, \left\{\ell'_{j,b}\right\}_{j,b}, \ell_{\langle j^* \rangle}\right)$:
  1. For every i, j and $b = D[i,j]$, computes $\ell^i_j = \mathsf{Dec}(k_{i,j,b}, \ell'_{j,b})$.
  2. For any i, computes $r_i = \mathsf{Eval}\left(C, \widetilde{C}, \left(\left\{\ell^i_j\right\}_{j \in [N_c]}, \ell_{\langle j^* \rangle}\right)\right)$.
  3. Outputs $a = (r_1, \ldots, r_{N_s})$.

- **Decoding:** $\mathsf{Decode}(a, i^*, j^*, dec)$ given an answer $a \in \{0,1\}^{N_s}$, an index (i^*, j^*) and a decoding key $dec \in \{0,1\}$, outputs $dec \oplus a_{i^*}$.

To prove Theorem 4.1 we show that the protocol is correct and secure. The claimed communication complexity follows readily.

Claim 4.3. *The protocol is correct.*

Proof. Fix D, i^*, j^*. Observe that for every $j \in [N_c]$ the value $\ell^{i^*}_j$ computed in Step 1 of Answer satisfies by definition

$$\ell^{i^*}_j = \mathsf{Dec}(k_{i^*,j,b_j}, \ell'_{j,b_j}) = \mathsf{Dec}(k_{i^*,j,b_j}, \mathsf{Enc}(k_{i^*,j,b_j}, \ell_{j,b_j})) = \ell_{j,b_j} \ ,$$

where $b_j = D[i^*, j]$ and ℓ_{j,b_j} is the label produced by Gar. Therefore,

$$r_{i^*} = \mathsf{Eval}\left(C, \widetilde{C}, \left(\left\{\ell^{i^*}_j\right\}_{j \in [N_c]}, \ell_{\langle j^* \rangle}\right)\right) = \mathsf{Eval}\left(C, \widetilde{C}, \left(\{\ell_{j,b_j}\}_{j \in [N_c]}, \ell_{\langle j^* \rangle}\right)\right) \ .$$

By the correctness of $(\mathsf{Gar}, \mathsf{Eval})$,

$$r_{i^*} \oplus dec = \mathsf{Eval}\left(C, \widetilde{C}, \left(\{\ell_{j,b_j}\}_{j \in [N_c]}, \ell_{\langle j^* \rangle}\right)\right) \oplus d = D[i^*, j^*].$$

□

Claim 4.4. *The protocol is secure.*

Proof. Fix an oracle-aided efficient adversary $\mathcal{A}$. Recall that Query_q is the algorithm that given input (sk, i, j) executes $\mathsf{Query}(sk, i, j)$ to get (q, dec) and outputs only q. We prove that $\mathcal{A}$, given $\widehat{D} \leftarrow \mathsf{Preprocess}(sk, D)$, cannot distinguish a (real) query oracle $\mathsf{Query}_q(sk, \cdot)$ from a (simulated) query oracle that for any input (i^*, j^*) returns a fresh sample from a fixed distribution independent of (i^*, j^*). We show this by a hybrid argument.

Hybrid 0: Real. This is the real experiment where the adversary $\mathcal{A}$ has access to $\mathsf{Query}_q(sk, \cdot)$.

Hybrid 1: Random Symmetric Encryption Keys. In this hybrid, all keys $k_{i,j,b}$ are chosen truly at random instead of being derived using the PRF F_{sk}. Since $\mathcal{A}$ does not obtain sk, this hybrid is indistinguishable from the previous one by the security of the PRF.

Hybrid 2: Uniformly Random Encrypted Labels $\ell'_{j,1-D[i^*,j]}$. In this hybrid, whenever $\mathsf{Query}_q(\cdot)$ is invoked with i^*, j^*, for any j and $b \neq D[i^*, j]$, instead of returning $\ell'_{j,b} = \mathsf{Enc}(k_{i^*,j,b}, \ell_{j,b})$, we return a fresh uniformly random string. To argue that this hybrid is indistinguishable from the previous one, first recall that $\mathsf{Enc}(k_{i^*,j,b}, \ell_{j,b}) \equiv (r, E_{k_{i^*,j,b}}(r) \oplus \ell_{j,b})$ for $r \leftarrow \{0,1\}^\lambda$ and a PRF E. Noting that in the previous hybrid, the corresponding keys $k_{i^*,j,b}$ are uniform and independent of $\mathcal{A}$'s view, we can invoke the security of the PRF E, replacing any $E_{k_{i^*,j,b}}(\cdot)$ with a truly random function $\Pi_{i^*,j,b}(\cdot)$. At this point, any corresponding encryption $\mathsf{Enc}(k_{i^*,j,b}, \cdot)$ is statistically close to uniform. Indeed, it is perfectly uniform provided we do not sample the same encryption randomness r twice, which occurs with overwhelming probability $1 - 2^{-\Omega(\lambda)}$.

Hybrid 3: Uniformly Random Labels $\ell_{j,D[i^*,j]}, \ell_{\langle j^* \rangle}$. In this hybrid, the labels $\ell_{j,D[i^*,j]}, \ell_{\langle j^* \rangle}$ sampled by the garbler Gar are replaced with uniformly random labels. Indistinguishability from the previous hybrid follows the uncorrelated input encoding of the garbling scheme.

Hybrid 4: Uniformly Random Encrypted Labels $\ell'_{j,D[i^*,j]}$. In this hybrid, whenever $\mathsf{Query}_q(\cdot)$ is invoked with i^*, j^*, for any j and $b = D[i^*, j]$, instead of returning $\ell'_{j,b} = \mathsf{Enc}(k_{i^*,j,b}, \ell_{j,b})$, we return a fresh uniformly random string. We argue that this hybrid is identically distributed as the previous one. Indeed, in the previous hybrid $\mathsf{Enc}(k_{i^*,j,b}, \ell_{j,b}) \equiv (r, E_{k_{i^*,j,b}}(r) \oplus \ell_{j,b})$, where now $\ell_{j,b}$ is uniformly random.

Conclusion. We argue that in the last hybrid any input (i^*, j^*) is answered with a fresh sample from a fixed distribution independent of (i^*, j^*). Indeed, in this hybrid, the oracle returns $\left(\widetilde{C}, \left\{\ell'_{j,b}\right\}_{j,b}, \ell_{\langle j^* \rangle}\right)$. Where $\widetilde{C}$ is a garbled circuit and $\left(\left\{\ell'_{j,b} : j \in [N_c], b \in \{0,1\}\right\}, \ell_{\langle j^* \rangle}\right)$ are uniformly random, all independent of (i^*, j^*).

□

Acknowledgment

We thank Tamer Mour and Brent Waters for pointing out the notion of blind garbled circuits in [BLSV18].

References

[BGI15] Elette Boyle, Niv Gilboa, and Yuval Ishai. "Function Secret Sharing". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2015, pp. 337–367 (cit. on p. 1).

[BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. "Foundations of garbled circuits". In: *Proceedings of the 2012 ACM conference on Computer and communications security*. 2012, pp. 784–796 (cit. on p. 3).

[BIM04] Amos Beimel, Yuval Ishai, and Tal Malkin. "Reducing the Servers' Computation in Private Information Retrieval: PIR with Preprocessing." In: *Journal of cryptology* 17.2 (2004) (cit. on p. 2).

[BIPW17] Elette Boyle, Yuval Ishai, Rafael Pass, and Mary Wootters. "Can we access a database both locally and privately?" In: *Theory of Cryptography Conference*. Springer. 2017, pp. 662–693 (cit. on pp. 2, 6).

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. "Anonymous IBE, leakage resilience and circular security from new assumptions". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2018, pp. 535–564 (cit. on pp. 4–6, 8, 13).

[BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. "The round complexity of secure protocols". In: *Proceedings of the twenty-second annual ACM symposium on Theory of computing*. 1990, pp. 503–513 (cit. on pp. 4, 5, 8).

[CGKS98] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. "Private information retrieval". In: *Journal of the ACM (JACM)* 45.6 (1998), pp. 965–981 (cit. on p. 1).

[CHR17] Ran Canetti, Justin Holmgren, and Silas Richelson. "Towards doubly efficient private information retrieval". In: *Theory of Cryptography Conference*. Springer. 2017, pp. 694–726 (cit. on pp. 2, 6).

[CIMR26] Caicai Chen, Yuval Ishai, Tamer Mour, and Alon Rosen. "Secret-key PIR from random linear codes". In: *Proceedings of the 58th Annual ACM Symposium on Theory of Computing*. 2026 (cit. on pp. 2, 6).

[DMO00] Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. "Single Database Private Information Retrieval Implies Oblivious Transfer". In: *International Conference on the Theory and Applications of Cryptographic Techniques*. 2000 (cit. on p. 1).

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "On the Cryptographic Applications of Random Functions". In: *Advances in Cryptology: Proceedings of CRYPTO 84*. 1984, pp. 276–288 (cit. on p. 6).

[HILL99] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. "A pseudorandom generator from any one-way function". In: *SIAM Journal on Computing* (1999), pp. 1364–1396 (cit. on p. 6).

[IKO05] Yuval Ishai, Eyal Kushilevitz, and Rafail Ostrovsky. "Sufficient conditions for collision-resistant hashing". In: *Theory of Cryptography Conference*. Springer. 2005, pp. 445–456 (cit. on p. 1).

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. "Replication is NOT Needed: SINGLE Database, Computationally-Private Information Retrieval". In: *Proceedings 38th annual symposium on foundations of computer science*. 1997, pp. 364–373 (cit. on pp. 1, 2).

[KT00] Jonathan Katz and Luca Trevisan. "On the efficiency of local decoding procedures for error-correcting codes". In: *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. 2000, pp. 80–86 (cit. on p. 1).

[LMW23] Wei-Kai Lin, Ethan Mook, and Daniel Wichs. "Doubly efficient private information retrieval and fully homomorphic RAM computation from ring LWE". In: *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*. 2023, pp. 595–608 (cit. on p. 2).

[SS10] Amit Sahai and Hakan Seyalioglu. "Worry-free encryption: functional encryption with public keys". In: *Proceedings of the 17th ACM conference on Computer and communications security*. 2010, pp. 463–472 (cit. on p. 3).

[Yao86] Andrew Chi-Chih Yao. "How to Generate and Exchange Secrets". In: *focs27*. 1986, pp. 162–167 (cit. on p. 3).